New Developments in Data Analysis Tools: The Anaphe project

8th Topical Seminar on
Innovative Particle and Radiation Detectors
Siena, 21-24 October 2002

Lorenzo Moneta
CERN IT/API
Lorenzo.Moneta@cern.ch

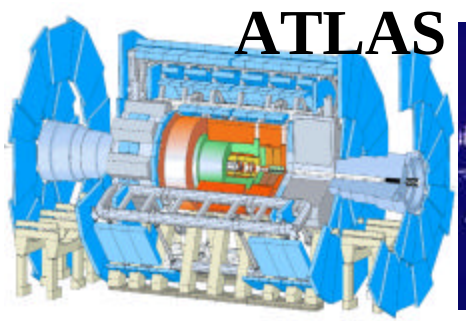
Outline

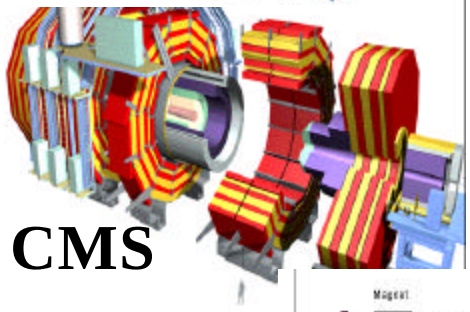
- Introduction
- AIDA (Abstract Interfaces for Data Analysis)
  - concept and design
- Anaphe
  - architecture
  - components
  - user examples
- Summary and Conclusions

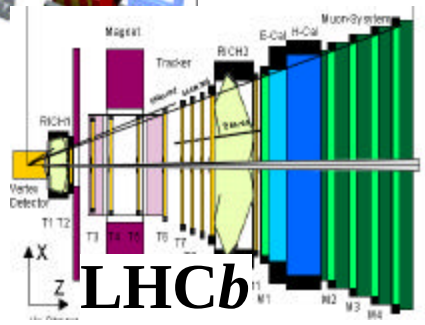
Introduction

- Complexity of detectors and huge amount of data produced
  - Impose strong requirements on computing systems and their software to reconstruct and analyze the data
- Need a long term vision
  - Technology changes
  - Maintenance
- Use of modern techniques:
  - Rigorous software engineering
  - OO programming
  - Importance of modular software
    - see I. Papadopoulos 's talk on Monday
  - Role of abstract interfaces

    (component based programming)
- Example: success of Geant4

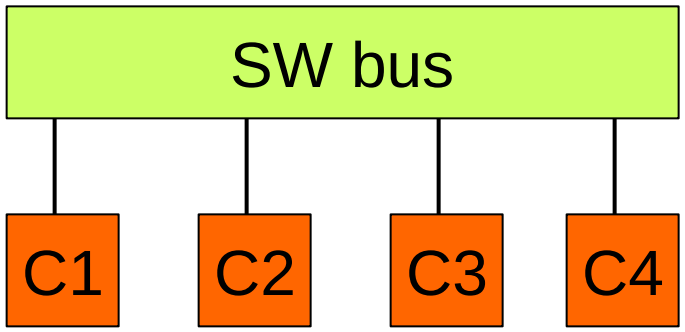






Abstract Interfaces

- An Abstract Interface (Class) specifies a protocol how clients may access and manipulate a component
- Defines no implementation but only functionality
- Essential element of OO to achieve a modular design:
  - Clean separation of specification and implementation
  - Clean separation of components
  - Components can be upgraded or replaced without effecting usage (plug in /out model)



Interfaces are the communication protocol of the bus

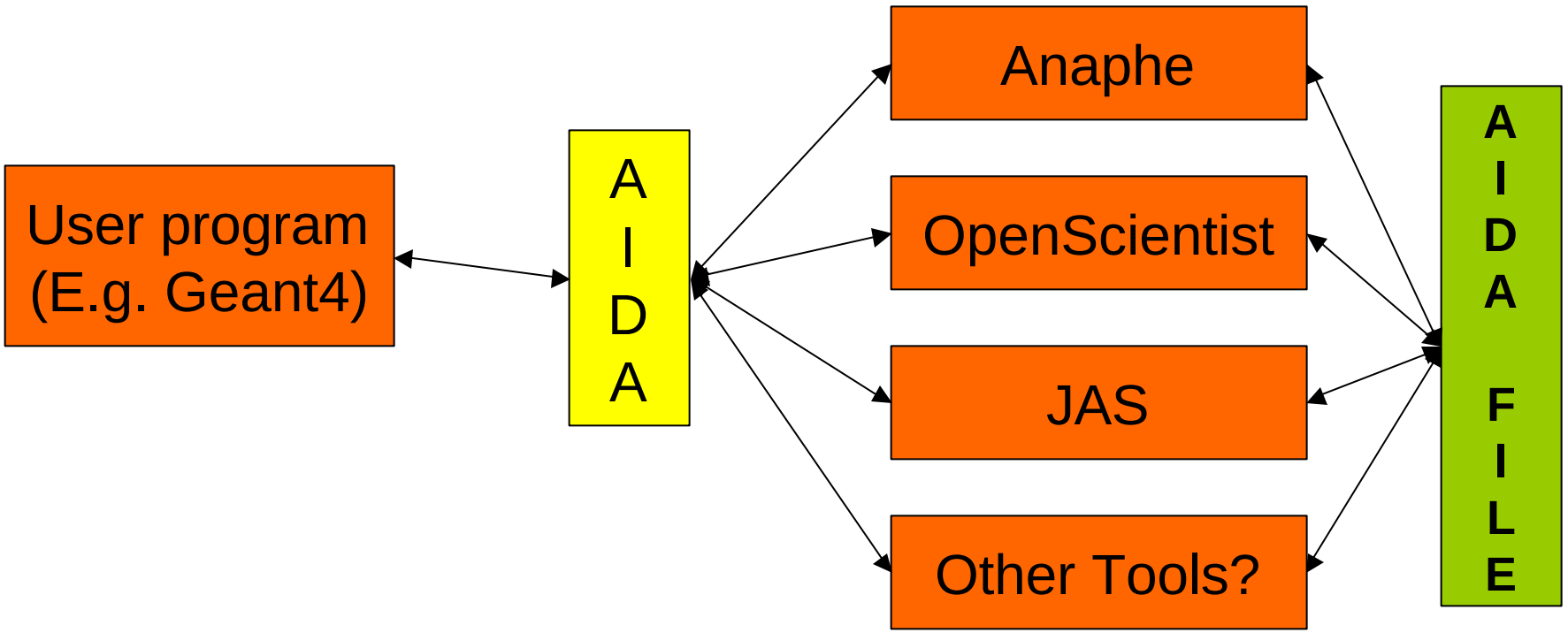
components

What is AIDA ?

- AIDA : Abstract Interfaces for Data Analysis
- Open source project with the goal to define abstract interfaces for common physics analysis objects
  - Histograms, ntuples, functions, fitter, plotter, tree and data storage
- Defines a common XML format for data exchange
- Allows multiple implementations and multiple languages
  - C++, Java and Python
- Exist three AIDA implementations:
  - Anaphe (CERN) in C++
  - JAS (SLAC) in Java
  - OpenScientist (Orsay) in C++

AIDA

- User code sees only the abstract interfaces
  - Implementation can be selected at run time without any change to the code (loading shared libraries)

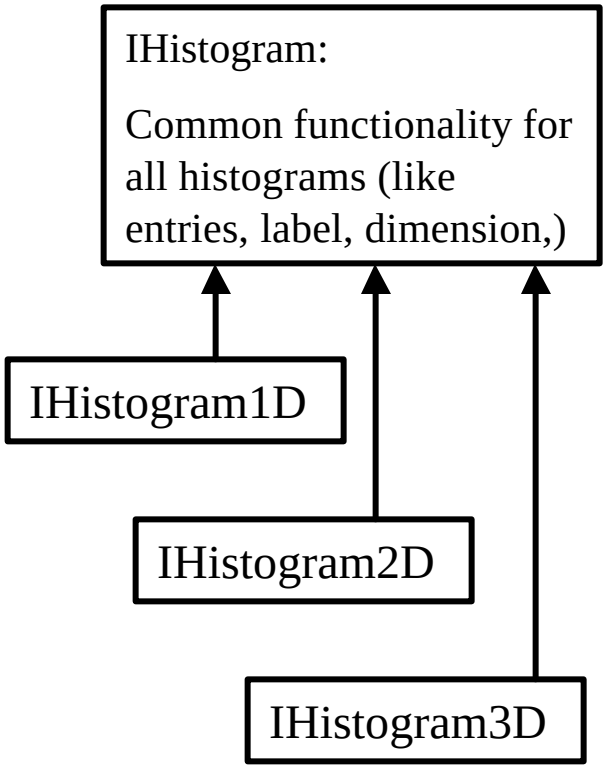


AIDA History

- AIDA started in 2000 by defining a common interfaces for histograms
- First end-user release (v. 2.2) end of 2001
- New AIDA release 3.0 in October 2002
  - large improvement in functionality (fitter and plotter)
  - New Anaphe release 5.0.0 implementing AIDA 3.0
  - JAS and OpenScientist releases expected soon
- Geant4 adopted AIDA for analysis
- AIDA is used also within Gaudi (SW framework used by LHCb, ATLAS and HARP)
- Recommended for adoption by LHC Computing Grid project (LCG)

Example of AIDA

Histogram interfaces



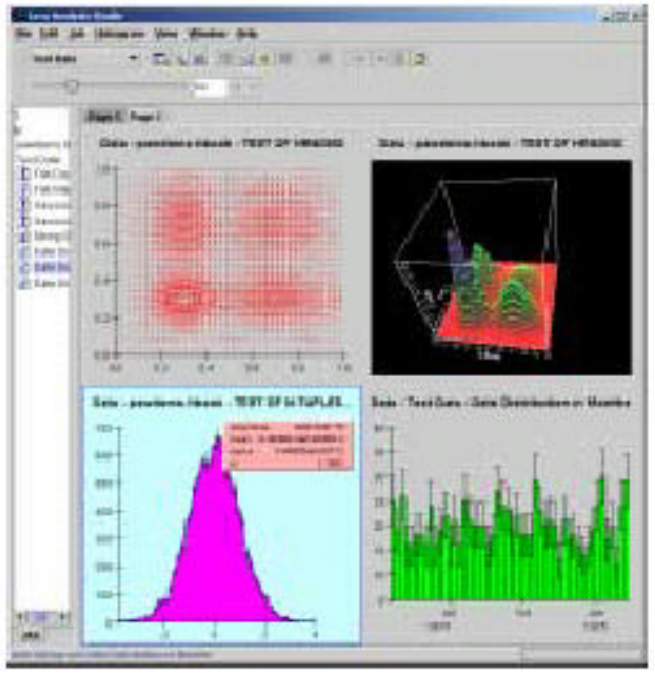
IHistogram1D interface

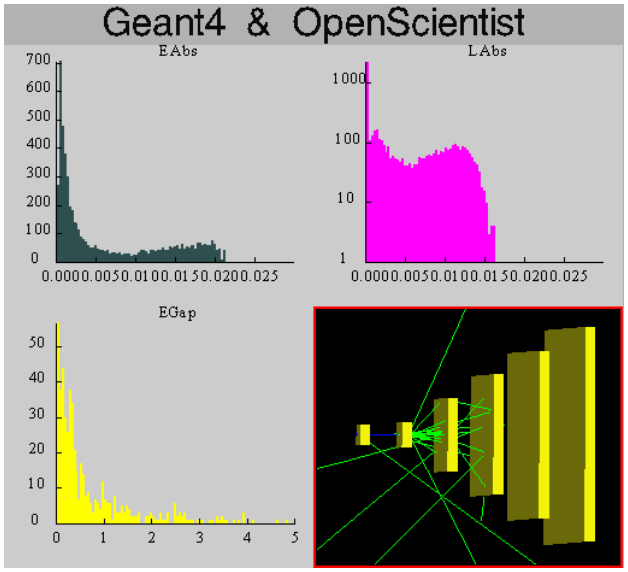
```
public interface IHistogram1D extends IHistogram {

    public void fill(double x);
    public void fill(double x, double weight);

    public double binCentre ( int index );
    public int binEntries(int index);
    public double binHeight(int index);
    public double binError(int index );

    public double mean();
    public double rms();

    public IAxis axis();
    public int coordToIndex(double coord);
}
```

AIDA implementations

⌘ JAS (Java Analysis Studio)

- jas.freehep.org/
- Analysis tools developed a SLAC written in Java
- Easy to use and robust, multi platform, flexible and easy extendable
- Large user community (BaBar, GEANT4 through AIDA)





⌘ OpenScientist

- http://www.lal.in2p3.fr/OpenScientist
- Modular tool developed by G. Barrand (Orsay)
- Collections of various C++ packages (histogramming, visualisation, storage)

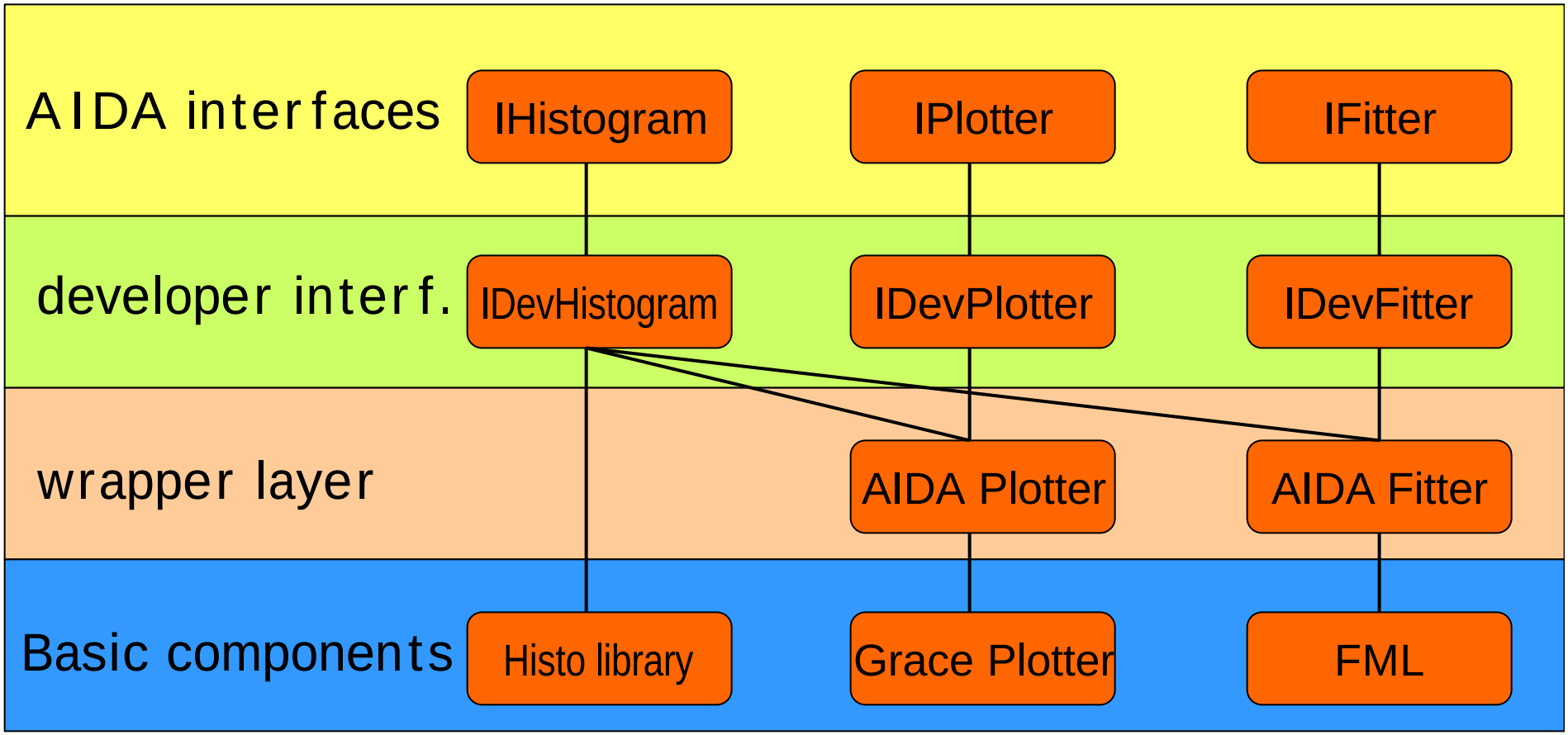
Anaphe

- Anaphe : Analysis for Physics Experiments
- An project in CERN IT division to provide a modular OO/C++ alternative to CERNLIB
- Provides libraries for
  - Histograms and Ntuples
  - Plotting and visualisation
  - Fitting and Minimisation
  - Management and storage
  - Interactive analysis using Python (Lizard)
- Try to use standards wherever possible
- Try to re-use existing class libraries

Layered Architecture of Anaphe

- Basic functionalities (histograms, fitting, etc.) are available as individual C++ class libraries (components)
- A thin wrapper layer implementing AIDA using the component libraries
  - ➔ Easy to adapt to changes in interfaces due to user request (e.g. adding functionality)
- A developer interfaces level extending the AIDA interfaces
  - More efficient (extra functionality is needed internally)
  - Maintain insulation
  - ➔ Easy to replace a component without affecting usage
- User sees only top level (AIDA)

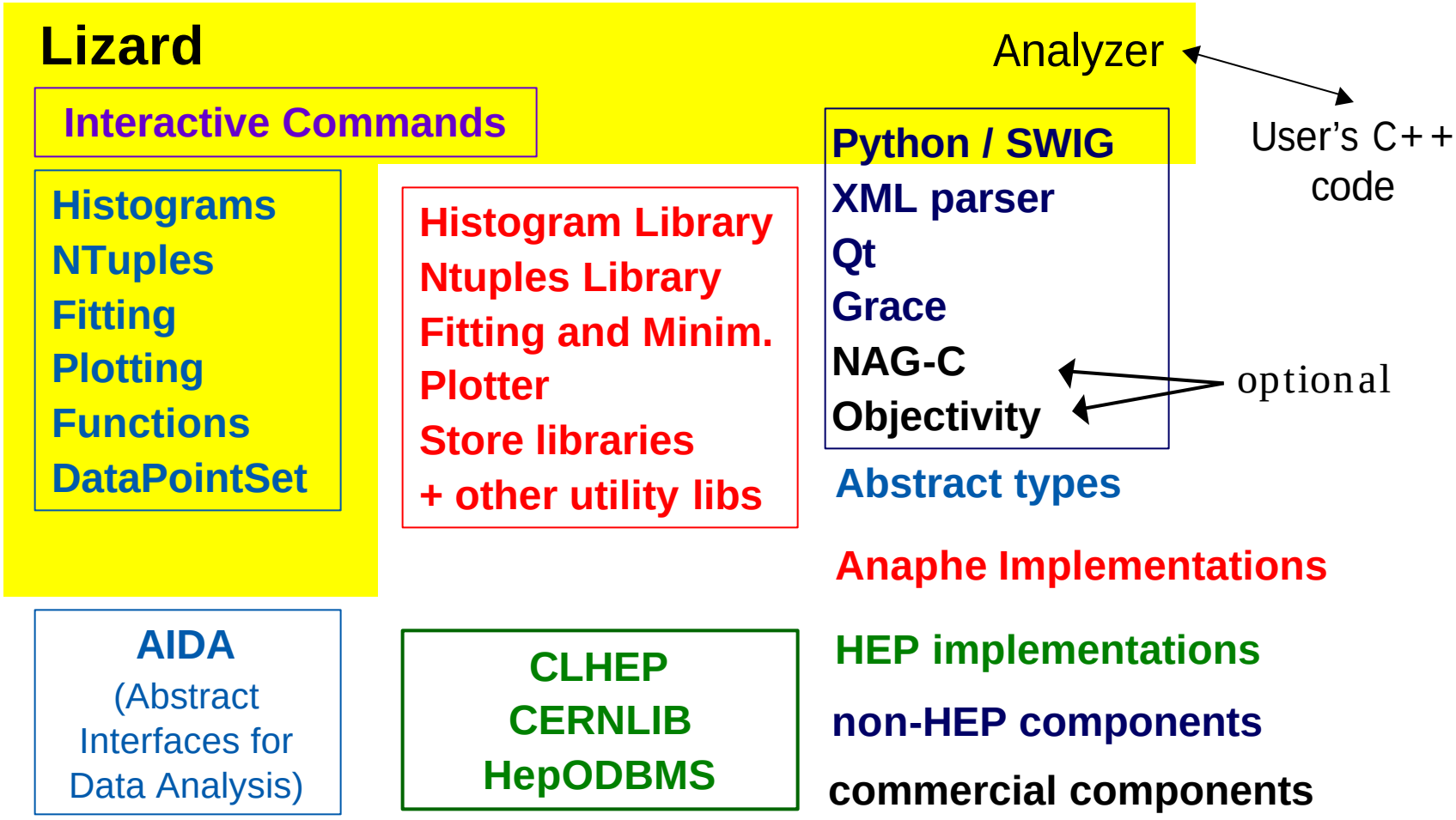
Anaphe Architecture



Architecture: developer interfaces

- Developer interfaces allow complete decoupling between different components
- Examples:
  1. Persistency store libraries are decoupled from histograms
     - Store uses developer interface to copy contents from the store in the histograms
     - No direct coupling between store library and histogram library
  2. Plotter library not coupled to data objects libraries (histograms)
- Converge between different AIDA implementations to use same developer interfaces:
  - mixed use of implementations (Code sharing)
    - Anaphe fitter with JAS histograms

Anaphe Components

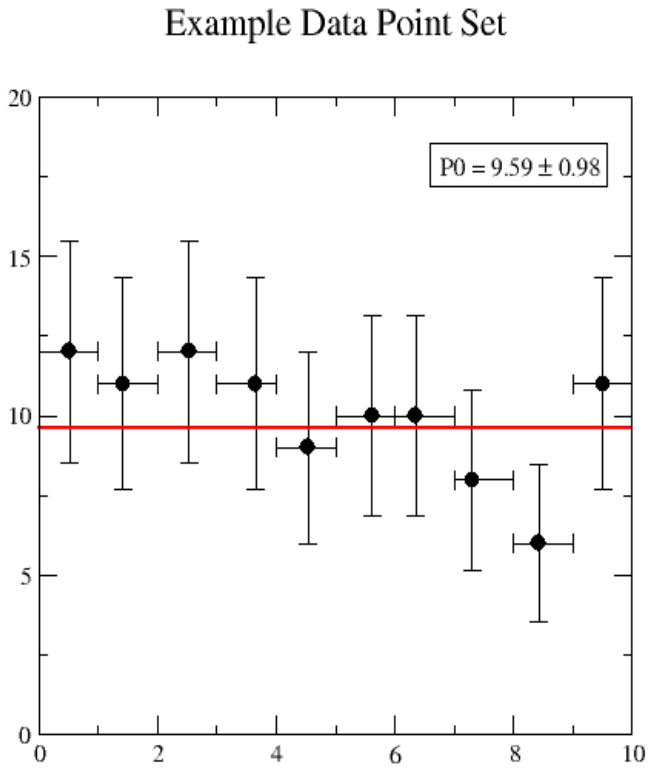


Anaphe History

- LHC++ project started in 1997
- HEP foundation libraries developed 1997-2000
- Anaphe started in 2000 with first version of Lizard (interactive python component)
- Production version Summer 2001
- Major re-design in 2002 to integrate with AIDA
  - AIDA 2.2 compliant version Summer 2002
- New version October 2002 implementing AIDA v 3.0
  - New Wrappers for AIDA
  - Improved Histograms and Ntuples libraries
  - New Plotter library based on Grace
  - Introduction of XML store

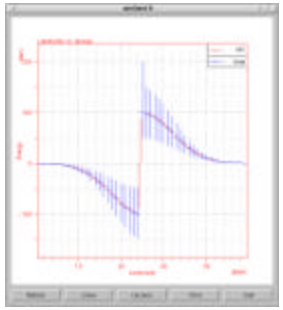
Histograms and Tuples libraries

- Histogram Library
  - Based on ideas of previous library (HTL) developed for LHC++
  - High performance
  - Histograms (up to 3D) and profiles
  - Unbinned histograms (clouds)
- NTuple Library
  - Raw and column wise (Hbook type)
  - Nested ntuples (ntuple in ntuple)
    - Event/track/hits
- Data Point Set (Vector of Points)
  - Simple container for n-dimensional measurement points (values and positive/negative errors)
  - Functionality to have operations (add/subtract) on different sets

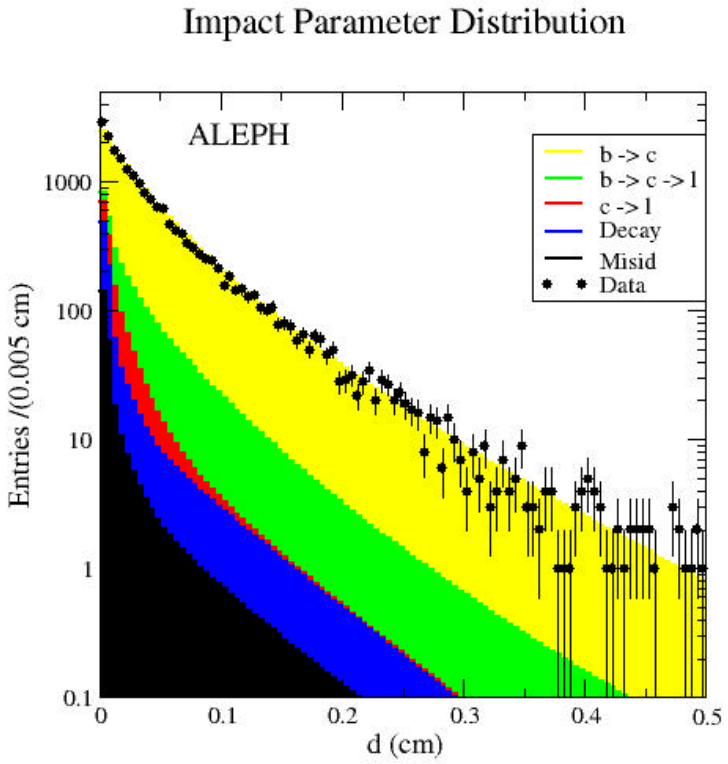


Plotter libraries

- Qplotter
  - used by old Anaphe versions
  - library based on Qt Free 3 (C++ & GUI open source library)



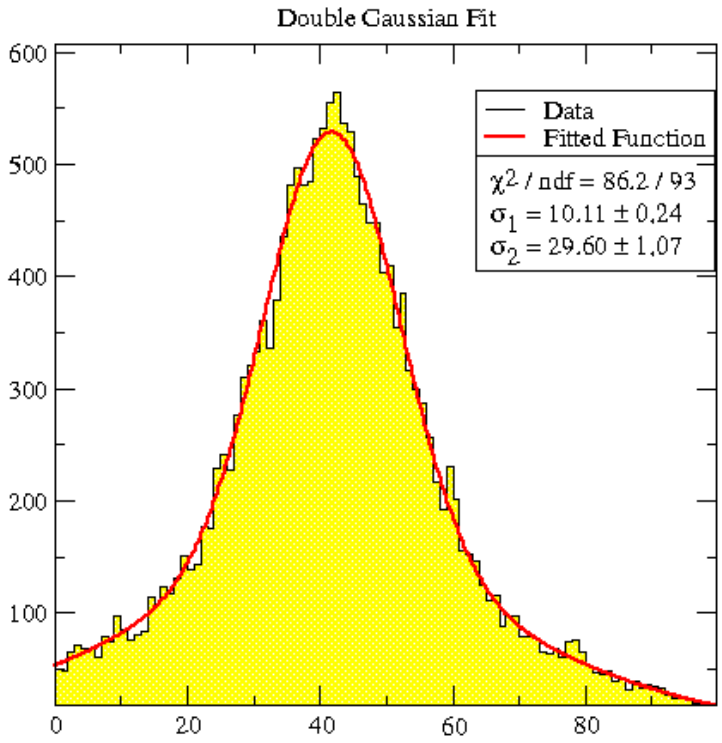
- GRACE Plotter
  - introduced in latest release
- Based on GRACE
  - a open source graphics package under GPL license
  - Very high quality graphics and powerful (publication quality plots)
  - Convenient point and click user interface
  - Flexible and easily extendable
  - Easy integration in Anaphe



Fitting library

Fitting and minimization library (FML)

- Flexible OO library.
- Using minimization engine based on NAGC/MINUIT but easy extendable to others
- Powerful: $\chi 2$, binned and unbinned maximum likelihood fits
- Plug-in mechanism to load user functions
- Implement new AIDA 3.0 interfaces (lots of new functionality)
- Integrated with all data sets (histograms, data points, ntuples)

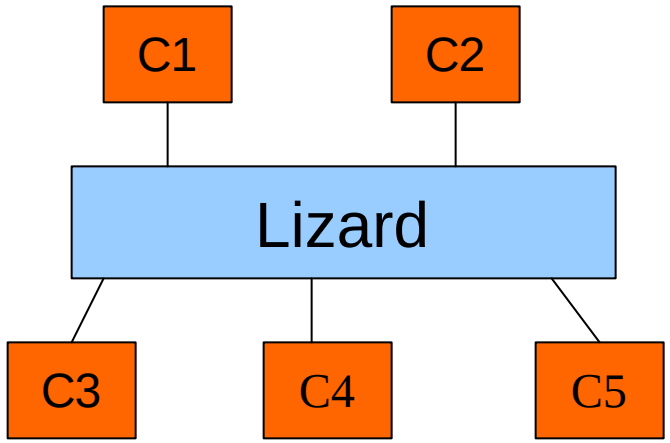


Management and persistency

- Hide implementations from user
  - Use factories to create objects (Histograms, Ntuples,….)
- Objects are managed in a tree-directory structure
  - Support for Unix-like directory and commands (ls, cp, mv, …)
- Tree hides store details from the user
  - User chooses store type at run time (when creating the tree)
- Multi store types functionality
  - can run with two different store type at the same time !
- Support in Anaphe for three store types:
  - XML (compress and uncompress) defined within AIDA
    - Possible to exchange files with other AIDA implementations (JAS)
  - Hbook (only histograms and Tuples)
  - Objectivity using HEPOBDMS
- Easy extendable to new types

Interactivity: Lizard

- Lizard : Python environment for interactive analysis
  - Unified user interface at top level
  - AIDA types and methods mapped into Python commands
    - use SWIG to generate the mapping from the C++ classes
  - User modules can be plugged in as required
  - Analyzer module provides on-the-fly compilation and running of user code
- Python as scripting language:
  - Easy to use
  - Object Oriented language
  - Maps well to C++ and Java
  - Huge user base with lots of free software (networking, GUI, OS, scientific etc)

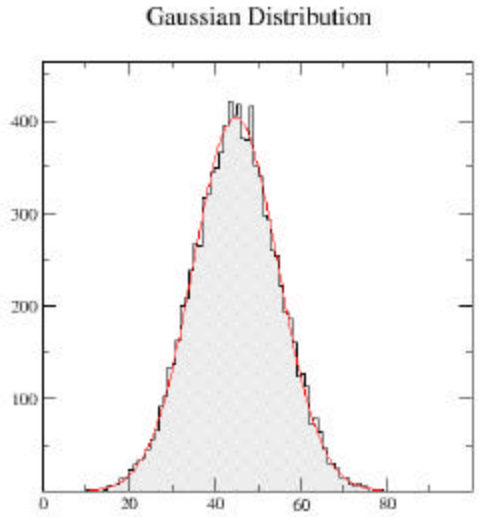


C++ component libraries

Example of Lizard

- Lizard code example in Python:
  - Creating an Histogram, filling, fitting and saving the result in an XML store

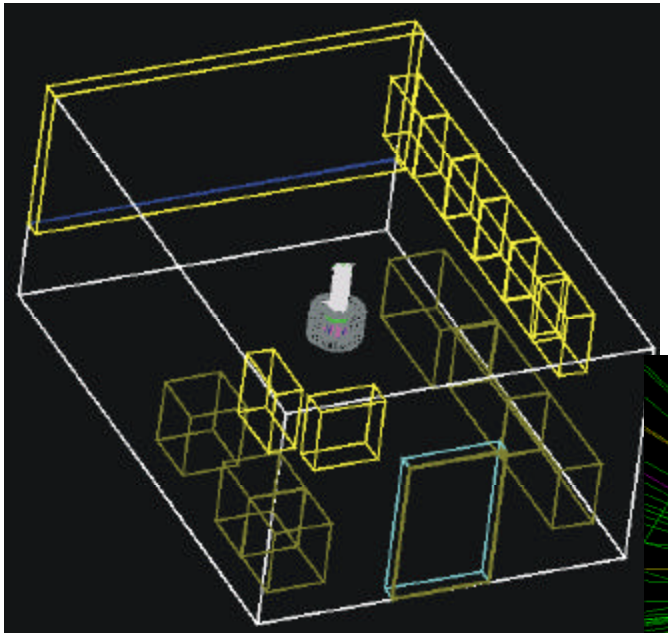
```
# create the tree with an XML store
tree=tf.create ("myExample.xml","XML",0,1)
# create histogram (first factory)
hf = af.createHistogramFactory(tree)
h1 = hf.createHistogram1D("1", "Gaussian Distribution",
100, 0., 100.)
# filling
for i in range(0,10000) :
   xval = 1.*random.gauss(45., 10.)
   h1.fill(xval,1.)

#fitting – create first fitter from Factory
fitter = fitterFactory.createFitter("Chi2","")
fitResult = fitter.fit(h1,"G")

#save all in XML file
tree.commit()
```



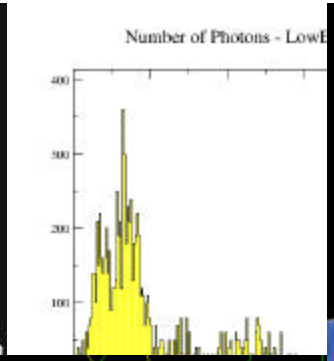
Anaphe Users

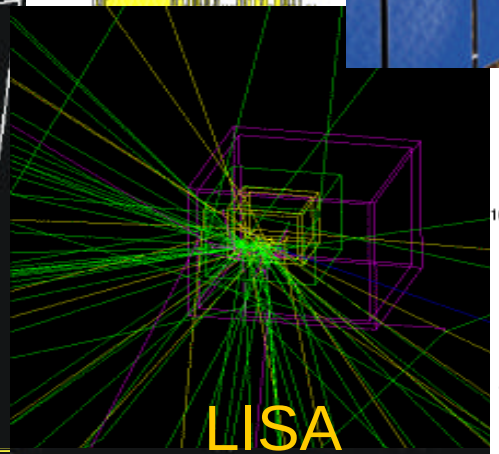
- Users from HEP and non HEP community
- Geant 4 has adopted AIDA as a tool-independent analysis standard
- Anaphe is used in GEANT4
  - In the advanced examples (ATLAS and CMS calorimeter test beam simulations)
  - And in analysis of underground, astroparticle experiments and even in medical applications (radiotherapy)
  - Adopted for GEANT4 test and validation process
- Running of Anaphe in a distributed environment (GRID)
  - ➔ See next talk of J. Moscicki
- Interest in AIDA also from LHC Computing Grid project (LCG)

Example of Anaphe users

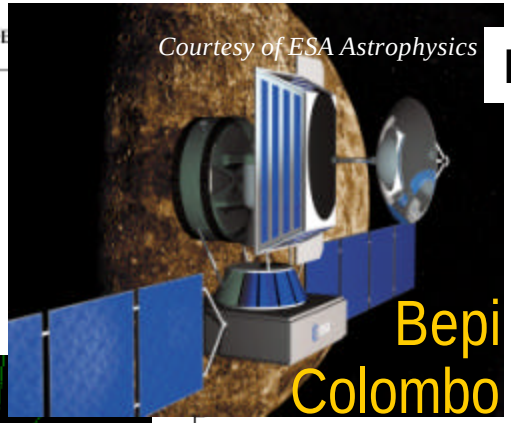


UK Dark Matter



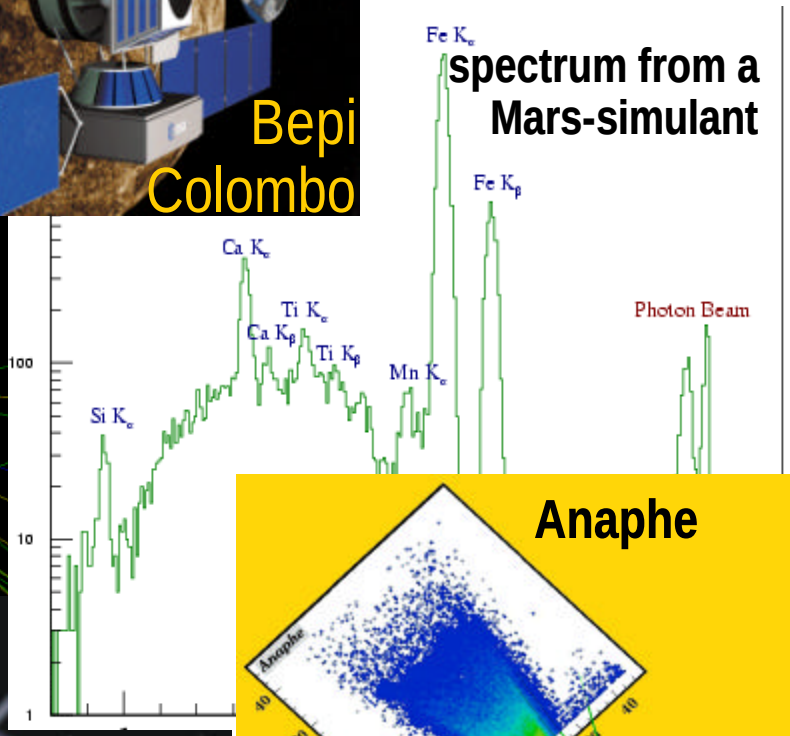


LISA

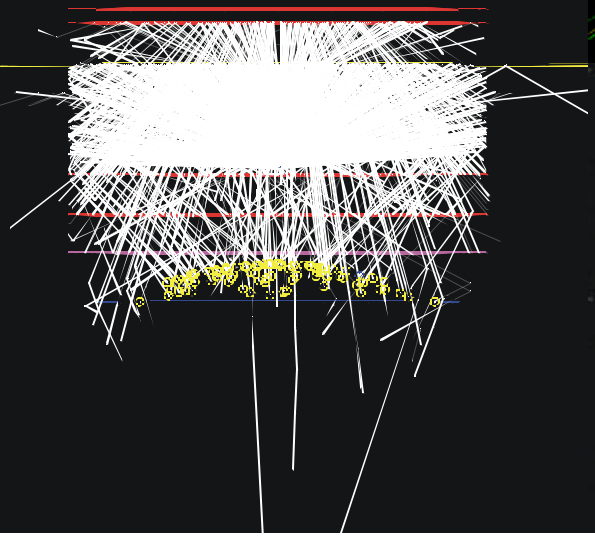


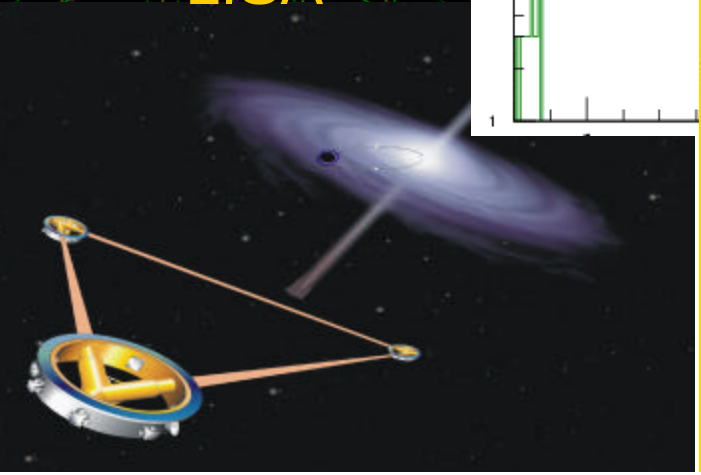
Bepi Colombo

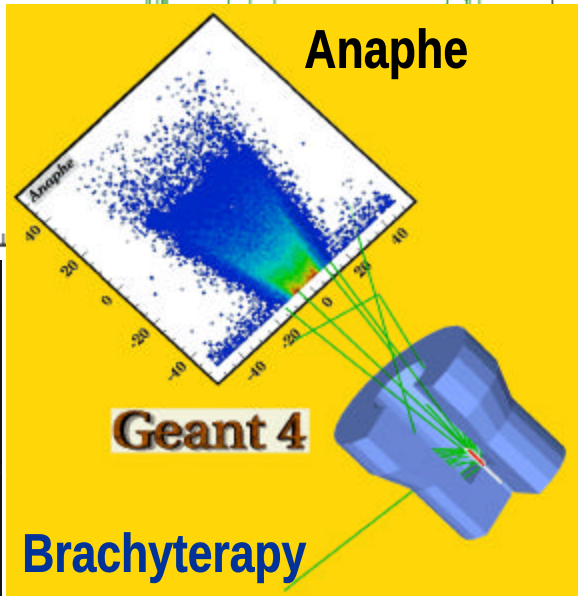
ESA mission to Mercury



spectrum from a Mars-simulant







Summary

- Anaphe is a layered set of loosely coupled C++ components for data analysis, plus an interactive Python framework (Lizard)
  - Easy to use
  - Applicable to different environment
- HEP-specific parts written in-house
- Developed and maintained by CERN IT
- Committed to AIDA compliance
  - Following LCG recommendation
- Open to new requirements from experiments

References

- For documentation, downloads and more information
  - **AIDA:**
    - **http://aida.freehep.org/**
  - **ANAPHE:**
    - **http://cern.ch/anaphe**
- or send mail to
    - **anaphe-editors@cern.ch**