Analysis with AIDA and Anaphe

Geant 4

User Workshop
CERN, 11-15 November 2002

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

- AIDA (Abstract Interfaces for Data Analysis)
  - Concept and design
- AIDA implementations (Anaphe)
- Description of the interfaces
- User examples
- Conclusions
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/JAIDA (SLAC) in Java
  - OpenScientist (Orsay) in C++
**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
AIDA

*User code sees only the abstract interfaces*

*yImplementation can be selected at run time without any change to the code (loading shared libraries)*

User program (E.g. Geant4) → AIDA → Anaphe → OpenScientist → JAS → Other Tools?

11/13/2002
Lorenzo Moneta, CERN IT/API
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 and JAS releases implementing AIDA 3.0
  - OpenScientist release is 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)
AIDA implementations

- **JAS (Java Analysis Studio)**
  - [jas.freehep.org/](jas.freehep.org/)
  - Analysis tools developed at SLAC written in Java
  - Easy to use and robust, multi platform, flexible and easy extendable
  - **JAIDA**: Java packages implementing AIDA interfaces

- **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
  - Follow up from LHC++ project (1997-2000) which provided OO/C++ libraries alternative to the Cernlib
- First production release in Summer 2001
- Implementation of the AIDA 3.0 interfaces in version 5 (October 2002)
- Provides component C++ libraries implementing AIDA interfaces
- **Lizard**: Interactive analysis tool in Python to use AIDA
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

AIDA interfaces
- IHistogram
- IPlotter
- IFitter

Developer abstract interfaces
- IDevHistogram
- IDevPlotter
- IDevFitter

Wrapper layer
- AIDA Plotter
- AIDA Fitter

Basic components
- Histo library
- Grace Plotter
- FML
AIDA Interfaces Summary

- **Histograms**
  - Binned 1-,2-,3- dimensional histograms
  - Unbinned 1-,2-,3- dimensional histograms (Clouds)
  - 1-,2- dimensional profile histograms

- **Tuples**

- **DataPointSet** (Vector of Points)

- **Functions**

- **Fitting interfaces**

- **Plotter interfaces**

- **Management of analysis objects:**
  - Tree
  - Factories
Histgrams

Example: IHistogram interfaces (binned histograms)

IHistogram:
Common functionality for all histograms (like entries, label, dimension,)

IHistogram1D

IHistogram2D

IHistogram3D

IHistogram1D interface

```java
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);
}
```
Tuples

**Tuple** - interface

- Support for basic C++ types (float/double/int/bool)
- Support for nested tuples (tuple in tuple)
  - E.g. Track/events/hits or Hbook column wise tuples
- Projection into histograms, clouds and profiles using evaluator and filters (weight and cuts)
  - IEvaluator and IFilter interfaces defined in AIDA and use C++ compiles expressions
- Support for chaining of tuples

**Implemented C++ library with**

- read/write of Hbook tuples (raw and column wise)
- Library is completed decoupled from the specific store in use
Data Point Set

Data Point Set (Vector of Points)

- Simple container for n-dimensional measurement points (values and positive/negative errors)
  - IDataPointSet
  - IDataPoint
  - IMeasurement

- Used for arithmetic operations, plotting and fitting

- Support conversion from histograms to DataPointSets
Functions and Fitting

- Function interface
  - Generic interface to n-dimensional function
  - Allows to set/retrieve parameters and get function value
  - Can provide gradient

- Fitting interfaces
  - IFitData: generic interface used to connect to data sources (Histograms, Clouds, DataPointSets, Tuples)
  - IFitter: interface allows to perform the fit and to configure it
    - E.g. setting different fitting methods (Chi2/ maximum likelihood)
  - IFitResult: to retrieve results (fitted parameters, errors,...)
  - IFitParameterSettings: to set bounds or fix the parameters
  - IFitRange: to set ranges on the source data
Fitting library

- Fitting and minimization library (FML)
  - Flexible OO library implementing AIDA interfaces
  - Using minimization engine based currently on NAGC/MINUIT but easy extendable to others (GSL in the future?)
  - Support for $\chi^2$, binned and unbinned maximum likelihood fits
  - Plug-in mechanism to load user functions
Plotting

- Plotter and Region interfaces
- Style interfaces
  - To control the way objects are drawn
  - Styles for markers, lines, text, axes, fill area, etc...

- Library based on GRACE implementing AIDA
  - 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
Management and persistency

- Hide implementations from user
  - Use factories to create analysis objects (Histograms, Ntuples,....)

- Objects are managed in a tree-directory structure (ITree interface)
  - 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
AIDA XML data format

- Defined an XML format to store all analysis objects of AIDA
  - Histograms, Functions, Tuples, etc…
- Allows transfer between different implementations
  - Anaphe files can be read from JAS and vice versa
- Support for compression (zipped) format
- Format (schema) is defined in:
  - http://aida.freehep.org/schemas/3.0/aida.dtd
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

C1 -> Lizard
Lizard -> C2
C3
C4
C5
Example of Lizard code (Python)

Creating an Histogram, filling, fitting and saving the result in an XML store

```python
# 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("MyHisto", "Gaussian Distribution", 100, 0, 100)
# filling with random gaussian points
for i in range(0,10000):
    h1.fill(random.gauss(45, 10), 1)

#fitting - create first function
funf = af.createFunctionFactory(tree)
fun = funf.createFunctionByName("MyFunction","G")
# set function’s initial parameters (optional)
p = [50,10,10]
p.setParameters(p)
# create fitter and fit the histogram
fitter = fitterFactory.createFitter("Chi2","")
fitResult = fitter.fit(h1,f)

#save all in XML file
tree.commit()
```
AIDA/Anaphe Users

- Users from HEP and non-HEP community
- Interest in AIDA also from LHC Computing Grid project (LCG)
- Geant 4 has adopted AIDA as a tool-independent analysis standard

- Anaphe starts being used in GEANT4
  - E.g. analysis of underground, astroparticle experiments and even in medical applications (radiotherapy)
  - Being adopted for GEANT4 test and validation process
Summary

- **AIDA interfaces** define a protocol for the analysis objects
  - Remove dependency (compile time) of user code from analysis library
  - User code needs no change if changing implementations
  - Allow interoperability between different frameworks

- **Anaphe** is a layered set of loosely coupled C++ components for data analysis and an interactive Python framework (**Lizard**)
  - Easy to use
  - Applicable to different environment
  - Committed to AIDA compliance

- Open to new requirements and feedbacks from users
References

❖ For documentation, downloads and more information
❖ AIDA:
  ➡️ http://aida.freehep.org/
❖ AIDA User Guide
❖ ANAPHE:
  ➡️ http://cern.ch/anaphe

❖ or send mail to
  ➡️ anaphe-editors@cern.ch