Analysis Tools

A brief introduction to AIDA

Anton Lechner[1]

[1]CERN, Geneva, Switzerland

ORNL, May 22nd 2008

Geant4 and Analysis

- **Geant4 does not contain any analysis tools**
  - Geant4 is a particle transport simulation package: Data analysis is not its primary objective
  - A user must introduce his/her own analysis functionalities
  - Many advanced tools exist, which may be used for analysis of the simulation output
- **Basic strategy**
  - Store simulation output in an appropriate format
    - Adopt your own approach (plain ascii file, csv,...)
    - Use advanced packages: AIDA-compliant tools, ROOT, ...
  - Process the data after the simulation using analysis tools
    - Gnuplot, Matlab, Octave, ...
    - AIDA-compliant tools (JAS, OpenScientist, ...), ROOT, PAW...

Choose an analysis tool according to your needs

Geant4 Analysis Example

- See: examples/extended/analysis/AnaEx01

HEP Tools

- **Geant4 was originally developed for high energy physics (HEP) applications**
  - Natural choice for many users to adopt analysis tools commonly used in the HEP community
- **Nowadays, Geant4 users work in many different domains (Medical physics, ...)**
  - The requirements concerning analysis have changed
  - Still, in many cases people use HEP tools
  - **However, you are free to use any tool you want**

A few HEP analysis tools are presented in the following

AIDA -- Abstract Interfaces for Data Analysis

This project is part of the Academic Software Organization. This organisation has been founded by an international group of computing scientists, engineers, physicists, in 1998 to help the development of software tools for academic scientific research in an international and collaborative v

The SLAC AIDA home page is now available

Categories

So far the following categories have been ider

- Histograms
- Vectors
- Ntuples
- Functions
- Fitter
- Plotter
- Analyzer
- EventDisplay
- Controller (UI)
- Messages

AIDA

http://wwwasd.web.cern.ch/wwwasd/lhc++/AIDA/

- **AIDA only defines interfaces**
- **There are several packages which are AIDA-compliant (see second part of the presentation)**

Introduction

OpenScientist is an integration of open source products working together to do scientific visualization and data analysis, in particular for high energy physics (HEP).

OpenScientist is definitely NOT one million lines of intricated and unnecessary complicated home made code reinventing everything.

Motivations

Experiments in hig
technologies but af
going to be a matte
that permits to inclu
everything at each

OpenScientist

http://openscientist.lal.in2p3.fr/

- **AIDA-compliant**

For data analysis, the HEP community had used the CERN/PAW tool for years. Whilst this program had been used and is probably still used by hundred of physicists in the world, it is not any more maintained by the lab that created and promoted it : CERN. This lab had not been able to establish long term collaborative software engineering plans in order to have a technical follow up of this program and analysis tools in general. (What is astounding, is that the same people that were behind CERN/PAW come now with another tool (ROOT) which has exactly the same engineering defects).

OpenScientist is first of all an architecture trying to handle the problem differently to avoid upsetting huge software phase transitions in the future. The key points of the system are the

JAS3

Documentation - Download - Source - Bug Database - Discussion

News

- **February 2, 2007** - JAS3, release 0.8.3 available
- **March 16, 2005** - JAS3, release 0.8.2 available
- **New Discussion Forums now availabl**

JAS3 is a general purpose, open-source, dat

- Plotting of 1d, 2d and 3d histograms, XY
- High quality print output.
- Fitting (binned or unbinned) using an extensible set of optimizers including Minuit.
- Export of plots in a variety of formats including PS, EPS, PDF, SVG, GIF, PNG.
- Easy to learn GUI for performing common analysis tasks (plotting, ntuple-analysis, fitting *etc.*).
- More complex analysis can be performed using a variety of scripting languages (pnuts, jython, *etc.*), or by writing Java analysis modules.
- Able to read data in a variety of formats including: SQL databases, text-files (see also HEP specific features).
- AIDA compliant analysis system.
- Built-in editor and compiler.
- Simple spreadsheet capabilities.
- Logbook for recording analysis tasks, and optionally publishing results to the web

JAS3

http://jas.freehep.org/jas3/

- **AIDA-compliant**

LHC Computing Grid Project

Physicist Interface (PI) Project

LCG Applications Area

NEW

2005/12/20: PI_1_3_12 released (release notes)
PI_1.3.12-UltraLite+HBook and

2005/09/24

2005/07/28

2005/05/19: PI_1_3_3 released (release notes)
PI_1.3.3-UltraLite+HBook and PI_1.3.3-Lite are available.

2005/04/15: PI_1_3_2 released (release notes)
PI_1.3.2-UltraLite+HBook is available.

2005/04/11: PI_1_3_1 released (release notes)

2005/02/23: PI_1_3_0 released (release notes)
PI_1.3.0-UltraLite+HBook is available.

old

PI

http://lcg-pi.web.cern.ch/lcg-pi

- **AIDA-compliant**
- **Not supported anymore**

ROOT

An Object-Oriented
Data Analysis Framework

Root

http://root.cern.ch/

Development release 5.19/04 NEW **14/05/2008**

The development release of ROOT 5.19/04 is now available.
In case you are upgrading from version 5.14,5.16 or 5.18, please read the releases notes of version 5.16 and 5.18 in addition to these notes.

The SVN tag for this version is **v5-19-04**.

Tar files for the source, documentation and binaries are available at:

Version 5.19/04 Release Notes
Full development notes (SVN logs)

CERN – European Laboratory for Particle Physics

PAW

Physics Analysis Workstation

PAW is conceived as an instrument to assist physicists in the analysis and presentation of their data. It provides interactive graphical presentation and statistic
histograms, event files (Ntuples), vectors, etc
Like *CERN Program Library* **PAW** usage an
License. For any questions about **PAW** conta
informations in the cern.heplib news group.

PAW Release Notes *(Last update: September 16th 2002)*	Release notes d
Known bugs *(Last update: October 27th 2004)*	A list of known bugs with workarounds. You can send a bug report to the **PAW** support person.
PAW Frequently Asked Questions	Frequently asked questions are listed here as well as answers to them. A list with some usage statistics is also available but the access is slower. A file containing all the FAQs is also available (useful for a printing purpose). You can try also the CERNLIB FAQs. The **PAW** FAQs are regularly reordered according to the number of accesses each one gets.

PAW

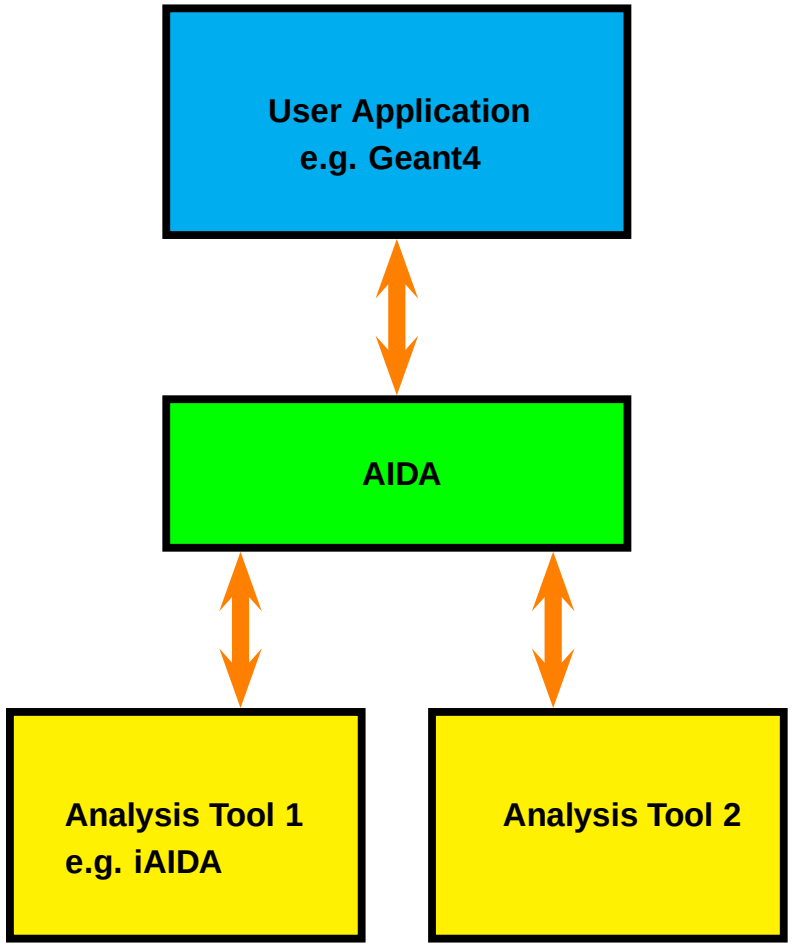
http://cern.ch/paw

- **Not supported anymore**
- **Listed for completeness**

Analysis based on AIDA (Abstract Interfaces for Data Analysis)

Powerful interfaces

- **AIDA contains a set of interfaces, that can be used regardless of the actual analysis system adopted for your application**
- **To use them with AIDA, analysis tools must be AIDA-compliant**
- **Interfaces available in C++ and Java**



Analysis based on AIDA (Abstract Interfaces for Data Analysis)

Advantages

- **Users need to get familiar only with one set of interfaces, even if they use different tools.**
- **Interoperability between (AIDA-compliant) tools is improved**
  - E.g. Data exchange in common storage format

Web

http://aida.freehep.org

XML: AIDA DTD

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE aida SYSTEM "http://aida.freehep.org/schemas/3.2.1/aida.dtd" >
<aida version="3.2.1">
<implementation package="pi" version="1"/>
<histogram1d name="energydeposit" path="/">
<annotation>
<item key="Title" value=""/>
<item key="Name" value="energydeposit"/>
<item key="Entries" value="2954521"/>
<item key="Mean" value="4.6017"/>
<item key="Rms" value="2.6180"/>
<item key="Extra Entries" value="0"/>
<item key="Overflow" value="0"/>
<item key="Underflow" value="0"/>
</annotation>
<axis direction="x" max="2.00000000e+01" min="0.00000000e+00" numberOfBins="200"
<statistics entries="2954521">
<statistic direction="x" mean="4.60173566e+00" rms="2.61799420e+00"/>
</statistics>
<data1d>
<bin1d binNum="0" entries="19667" error="3.41068562e-01" height="3.84858581e+01"
<bin1d binNum="1" entries="19135" error="3.39136174e-01" height="3.74355503e+01"
<bin1d binNum="2" entries="19709" error="3.43845025e-01" height="3.84493101e+01"
<bin1d binNum="3" entries="20275" error="3.52557299e-01" height="3.96591651e+01"
<bin1d binNum="4" entries="21377" error="3.59726359e-01" height="4.14686505e+01"
<bin1d binNum="5" entries="21425" error="3.64613301e-01" height="4.19952274e+01"
<bin1d binNum="6" entries="22184" error="3.75522504e-01" height="4.35834339e+01"
<bin1d binNum="7" entries="22876" error="3.74513676e-01" height="4.45023235e+01"
<bin1d binNum="8" entries="22145" error="3.60289572e-01" height="4.22770398e+01"
<bin1d binNum="9" entries="23054" error="3.75734685e-01" height="4.49317355e+01"
<bin1d binNum="10" entries="24452" error="3.87577540e-01" height="4.76144613e+01
<bin1d binNum="11" entries="24485" error="3.88205609e-01" height="4.75251403e+01
<bin1d binNum="12" entries="24180" error="3.81226123e-01" height="4.63081301e+01
<bin1d binNum="13" entries="24376" error="3.78446604e-01" height="4.64015217e+01
<bin1d binNum="14" entries="24741" error="3.80860429e-01" height="4.67680669e+01
<bin1d binNum="15" entries="24438" error="3.76234474e-01" height="4.58722091e+01
<bin1d binNum="16" entries="25820" error="3.86528588e-01" height="4.86974589e+01
<bin1d binNum="17" entries="27396" error="3.98164539e-01" height="5.07906874e+01
<bin1d binNum="18" entries="26675" error="3.96480823e-01" height="5.05151413e+01
<bin1d binNum="19" entries="26957" error="3.98990795e-01" height="5.12116324e+01
<bin1d binNum="20" entries="26738" error="3.94318970e-01" height="5.01963412e+01
<bin1d binNum="21" entries="27026" error="3.98725326e-01" height="5.08035462e+01
<bin1d binNum="22" entries="28155" error="4.16730052e-01" height="5.39968085e+01
<bin1d binNum="23" entries="28927" error="4.08211037e-01" height="5.40531261e+01
<bin1d binNum="24" entries="29643" error="4.21247755e-01" height="5.60905091e+01
<bin1d binNum="25" entries="28954" error="4.14181221e-01" height="5.44179285e+01
<bin1d binNum="26" entries="30132" error="4.15380624e-01" height="5.58361680e+01
```

AIDA-compliant analysis systems

Tools providing an implementation of AIDA interfaces

- **PI (Physics Interface)** - not maintained anymore
  - C++ and Python
  - http://lcg-pi.web.cern.ch/lcg-pi
- **iAIDA** - PI refactored
  - C++
  - http://iaida.dynalias.net/iAIDA.html
- **JAS**
  - Java
  - http://jas.freehep.org/jas3/
- **OpenScientist**
  - C++
  - http://openscientist.lal.in2p3.fr/

AIDA Interfaces for Analysis Objects (*not all listed*)

What does AIDA support?

- **1D, 2D and 3D histograms (IHistogram1D, ...)**
  - Filling and extracting data
  - Projections
  - Histogram arimethics (bin by bin: +,-,*,/)
- **1D, 2D and 3D clouds (ICloud1D, ...)**
  - unbinned histograms (can be transformed into binned ones)
- **N-Tuples (ITuple)**
  - Storing and retrieving N-tuple data
  - Ten different types of columns: the eight primitive types (int, short, long, float, double, char, boolean, byte), String and Object.

More AIDA interface definitions

What does AIDA support?

- **IO and Trees (ITree)**
  - Creating hierarchical structures of analysis objects (histograms, clouds, ...)
  - Saving and restoring analysis objects from files and databases
- **Functions and fitting (IFunction, IFitter)**
  - Definiting functions and applying fitting algorithms
  - Unbinned and binned fits
- **Plotting (IPlotter)**
  - Creating plots

Factories

Using factories to create AIDA objects

- **New AIDA objects are instantiated by factories**
  - A user is not required to instantiate analysis objects (histograms,..), plotters, trees, ... by himself
  - Different AIDA implementations might return different objects, but they all respect the defined interfaces

Example: Instantiating a histogram

- **1D Histogram (100 bins, lower/upper boundary = 0.0/1.0)**
  - `AIDA::IHistogramFactory* histogramFactory = ...` // see later how to create a histogram factory
  - `AIDA::IHistogram1D* hist1D = histogramFactory->createHistogram1D("MyHistogram",100,0,1.0);`

Histograms

AIDA supports 1D, 2D and 3D histograms. Creating histograms is quite straightforward using the methods in IHistogramFactory.

Histograms support arithmetic operations, in particular add, subtract, multiply and divide. In all cases the operation is applied bin-by-bin, and the histograms involved in the operation must have the same binning. The input histograms are unchanged, and a new histogram is created as the result of the operation. All of the arithmetic operations are methods of IHistogramFactory to reflect the fact that a new histogram is always created as a result of the operation. For multi-dimensional histograms it is also possible to create slices and projections.

The following example illustrates using histograms:

Histogram Arithmetic

AIDA User Manual

http://aida.freehep.org/doc/v3.3.0/UsersGuide

```
import hep.aida.*;
import java.util.Random;

public class HistogramArithmetic
{
   public static void main(String[] argv)
   {
      IAnalysisFactory af = IAnalysisFactory.create();
      IHistogramFactory hf = af.createHistogramFactory(af.createTreeFactory().create());

      IHistogram1D h1 = hf.createHistogram1D("test 1d",50,-3,6);
      IHistogram1D h2 = hf.createHistogram1D("test 2d",50,-3,6);

      Random r = new Random();
      for (int i=0; i<10000; i++)
      {
         h1.fill(r.nextGaussian());
         h2.fill(3+r.nextGaussian());
      }
```

Overview Package Class Tree Deprecated Index Help

AIDA API Version 3.3

PREV CLASS NEXT CLASS FRAMES NO FRAMES
SUMMARY: NESTED | FIELD | CONSTR | METHOD DETAIL: FIELD | CONSTR | METHOD

hep.aida

Interface IHistogram1D

All Superinterfaces:

IBaseHistogram, IHistogram

public interfac
extends IHist

User level inte

API documentation

http://aida.freehep.org/doc/v3.3.0/api/index.html

Author:

The AIDA team (http://aida.freehep.org/)

Method Summary

void	**add**(IHistogram1D hist) Add to this IHistogram1D the contents of another IHistogram1D.
IAxis	**axis**() Get the x axis of the IHistogram1D.

AIDA: A simple example

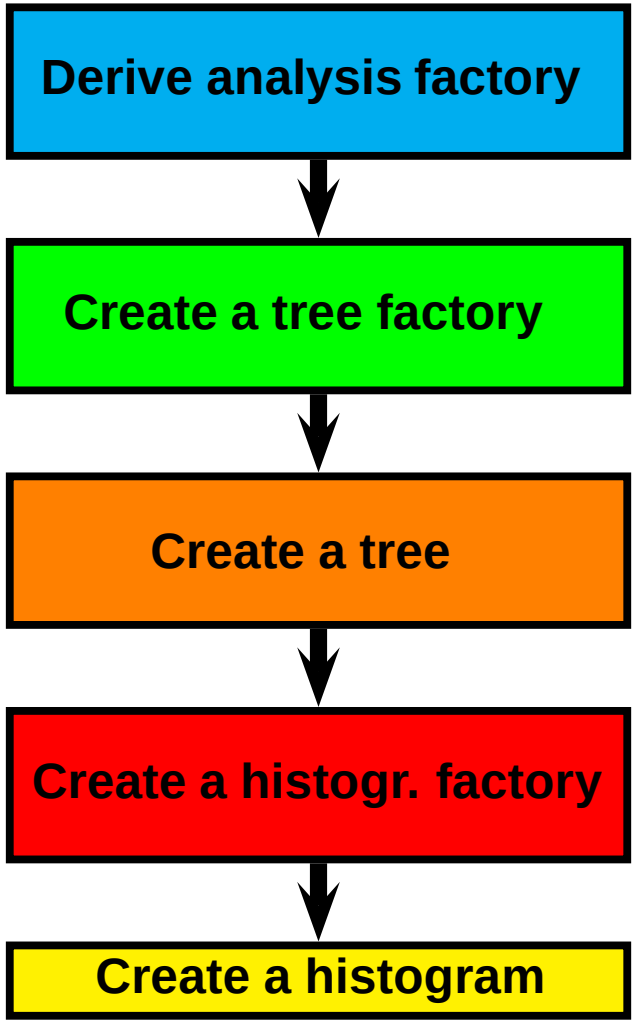
Creating, filling and storing a 1D histogram (XML)

```
AIDA::IAnalysisFactory* analysisFactory  = AIDA_createAnalysisFactory();

AIDA::ITreeFactory* treeFactory = analysisFactory->createTreeFactory();

AIDA::ITree* tree = treeFactory->create("output.xml",
                                        "xml",false, true,"uncompressed");
delete treeFactory;

AIDA::IHistogramFactory* histogramFactory =
                          analysisFactory->createHistogramFactory(*tree);

AIDA::IHistogram1D* aidaObject = histogramFactory->createHistogram1D("H",100,-0.5,0.5);
delete histogramFactory;

for(int i = 0; i < 100000; i++) {
   double val = CLHEP::RandGauss::shoot(0.0, 0.2);
   aidaObject -> fill(val);
}

tree -> commit();
tree -> close();
delete tree;
delete analysisFactory;
```

AIDA: A simple example

Basic strategy

- **Deriving a histogram**
  1. Use AIDA_createAnalysisFactory to get the analysis factory (AF) instance (singleton)
  2. Use the AF to create a tree factory (TF)
  3. Use the TF to create a tree
  4. Use the AF to create a histogram factory (HF) using the tree as argument
  5. Use the HF to create a histogram



AIDA: A simple example

Creating several histograms and clouds in a tree structure

```
AIDA::IHistogramFactory* histogramFactory =
                          analysisFactory->createHistogramFactory(*tree);

tree -> mkdir("/histograms");
tree -> cd("/histograms");
AIDA::IHistogram1D* hist1 = histogramFactory->createHistogram1D("H1",100,-0.5,0.5);
AIDA::IHistogram2D* hist2 = histogramFactory->createHistogram2D("H2",100,-0.5,0.5,
                                                                60,-0.2,0.2);

tree -> mkdir("/clouds");
tree -> cd("/clouds");
AIDA::ICloud1D* cloud1 = histogramFactory->createCloud1D("C1");
AIDA::ICloud2D* cloud2 = histogramFactory->createCloud2D("C2");

delete histogramFactory;

//     /
//     |-- clouds          Illustration of
//     |   |-- C1          the object hierarchy
//     |   `-- C2          in the current example
//     `-- histograms
//        |-- H1
//        `-- H2
```

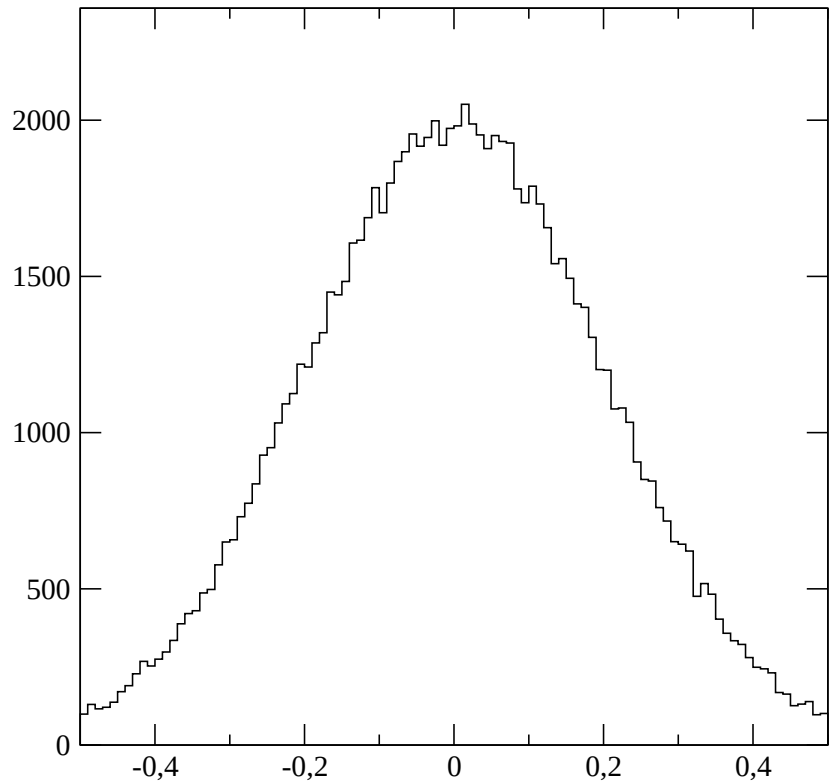
AIDA: A simple plotting example

Plotting a 1D histogram

- After filling a histogram it can be plotted using IPlotter:

```
AIDA::IPlotterFactory* plotterFactory =
  analysisFactory -> createPlotterFactory();

AIDA::IPlotter* plotter =
       plotterFactory -> create("Plot");

plotter -> region(0) -> plot(*aidaObject);
plotter -> writeToFile("histogram.ps","PS");

delete plotter;
delete plotterFactory;
```

- This example plot was created with iAIDA and Grace (plotting tool)



AIDA: A simple example

Retrieving a 1D histogram from a XML file

```
AIDA::IAnalysisFactory* analysisFactory  = AIDA_createAnalysisFactory();

AIDA::ITreeFactory* treeFactory = analysisFactory->createTreeFactory();

AIDA::ITree* tree = treeFactory->create("output.xml",
                                        "xml",true, false);
delete treeFactory;

AIDA:IManagedObject* obj = tree -> find("H1");

if(obj) AIDA::IHistogram1D* histogram = dynamic_cast<AIDA::IHistogram1D*>(obj);

delete tree;
delete analysisFactory;
```