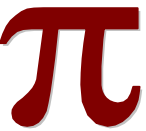
Status of PI Analysis Services

Lorenzo Moneta

CERN

AIDA Workshop

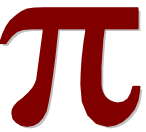
1/7/2003



http://lcgapp.cern.ch/project/pi/

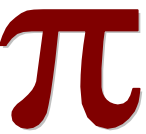
Analysis Services

- AIDA
  - Review Interface to Data Analysis
  - Adapt and extend them
    - Proxy layer for user convenience
- Root implementation of AIDA
  - Provide an implementation of the Interfaces to Data Analysis, as defined by the previous work package, based on Root.
- AIDA interface to SEAL and POOL services
  - Use SEAL and POOL to provide AIDA with object management and persistency services.
- Blueprint compliant Analysis tool set
  - Integration of various component in a consistent analysis tool set



AIDA Proxy layer

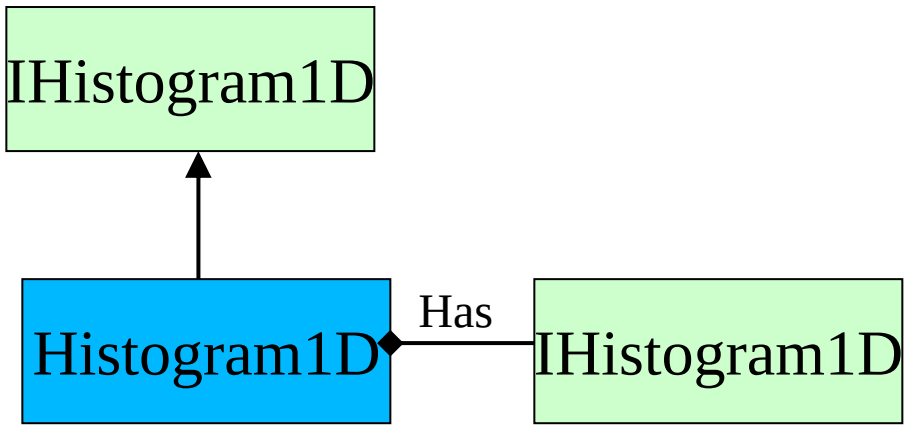
- C++ proxy classes to AIDA interfaces
  - **“Value semantics” for AIDA objects**
  - Implemented using the “Proxy” pattern, very easy !
    - 80% done using a script
  - Based only on AIDA Interfaces
    ➔ ***no dependency on a given implementation***
- Initially “hiding” of AIDA object management
  - AIDA tree is not exposed to users but hided in the Proxy implementation
- Keeping the functionality and signatures of AIDA
  - “re-shuffling” of factory methods to object constructors
- Examples on how to use with web-docs
  - Exist since March. Latest release : 0.2.1
  - Started integration with CMS SW
- Will be basis for a user-review and further evaluation
  - Any feedback will be propagated to AIDA team



AIDA_Proxy in more detail

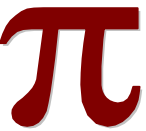
❖Histogram1D

```
namespace pi_aida {
class Histogram1D : public AIDA::IHistogram1D {
  public:
     // Constructor following the factory-create method (example)
    Histogram1D(std::string title, int nBins, double xMin, double xMax);
    // as an example the fill method:
    bool  fill ( double x, double weight = 1. )
          { if (rep == 0) return 0;
            else return rep->fill ( x , weight ); }
    // other methods are also mostly inlined …
  private:
    AIDA::IHistogram1D * rep;
}; }
```



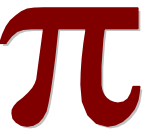
AIDA_Proxy classes

- **Generated Proxies for all AIDA data objects**
  - Histograms, Profiles, Clouds, DataPointSets, Tuples
- **Proxies also for Functions and Fitter**
  - Plotter is not yet done
- **AIDA_ProxyManager class**
  - Not exposed to users
  - Use AIDA factories to create objects
- **Proxy_Store**
  - Simple way of storing objects in a XML and/or a Root file
    - Only *open( ), write( ) and close( )* methods
  - Requested by users for evaluation of interfaces
- **HistoProjector**
  - Helper class for projections
  - Avoid using factories



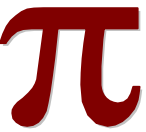
AIDA_ProxyManager

- **Implemented as a Loki singleton**
- **Objects are created using AIDA Factories**
- **No dependency on a particular implementation**
- **Factories (and relative implementations) are loaded using a plugin manager (from SEAL)**
  - Possible to choose implementation a run-time
  - Plugins exist now for all Anaphe implementations and for ROOT implementation of AIDA Histograms
- **Objects are managed by a memory tree**
  - Tree implementation can also be chosen using the plugin manager
- **Store objects using AIDA tree types (XML based or Root)**
  - Users interact only with the Proxy_Store



Example: Histogram

```
// Creating a histogram
  pi_aida::Histogram1D h1( "Example histogram.", 50, 0, 50 );
// Filling the histogram with random data
  std::srand( 0 );
  for ( int i = 0; i < 1000; ++i )
    h1.fill( 50 * static_cast<double>( std::rand() ) / RAND_MAX );

 // Printing some statistical values of the histogram
  std::cout << "Mean:" << h1.mean() << std::endl;
  std::cout << "RMS:" << h1.rms() << std::endl;

// Printing the contents of the histogram
const AIDA::IAxis& xAxis = h1.axis();
for ( int iBin = 0; iBin < xAxis.bins(); ++iBin )
    std::cout << h1.binMean( iBin )  << "       "
        << h1.binEntries( iBin)  << "       "
        << h1.binHeight( iBin ) << std::endl;
```

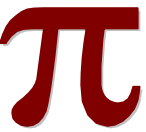


Example: Fitting a histogram

```
// create and fill the histogram …
// Creating the function which is going to be fitted with the histogram data
pi_aida::Function gaussFun("G");
// set parameters to starting values
gaussFun.setParameter("mean" , 50.);
gaussFun.setParameter("sigma", 10.);
gaussFun.setParameter("amp"  , 10.);
// Creating the fitter (ChiSquare by default)
pi_aida::Fitter fitter;   // or:   fitter("UnbinnedML")
// Perform the fit
AIDA::IFitResult& fitResult = *( fitter.fit( h1, gaussFun ) );
// Print the fit results
std::cout << "Fit result : chi2 / ndf : " << fitResult.quality() << " / " << fitResult.ndf() << std::endl;
for ( unsigned int i = 0; i < par.size(); ++i ) {
  std::cout << fitResult.fittedParameterNames()[i]
            << " = " << fitResult.fittedParameters()[i]
            << " +/- " << fitResult.errors()[i]
            << std::endl;
}
```

Example: Operations on Histograms

```
// Creating a histogram in Anaphe impl.
pi_aida::Histogram1D h1( "Example h1", 50, 0, 50, "AIDA_Histogram_Native" );
// fill h1
std::srand( 0 );
  for ( int i = 0; i < 1000; ++i )
    h1.fill( 50 * static_cast<double>( std::rand() ) / RAND_MAX );

// Creating a histogram in Root
 pi_aida::Histogram1D h2( "Example h2", 50, 0, 50, "AIDA_Root_Native" );
//Copying
h2 = h1;
//adding
pi_aida::Histogram1D h3 = h1 + h2;
```



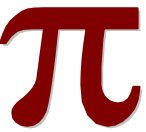
Example: Histogram Projections

```
// Creating a 2D histogram
  pi_aida::Histogram2D h( "Example 2D  hist.", 50, 0, 50, 50, 0, 50 );
// Filling the histogram…..
  …..

 // projections
pi_aida::HistoProjector hp;
pi_aida::Histogram1D hX = hp.projectionX(h);
pi_aida::Histogram1D hY= hp.projectionY(h);
```

- Implement projections on Histograms ?
  - **hX = h.projectionX()**

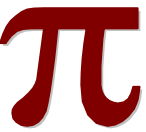
Example: Storing Histograms

```
// after created and filled the histograms
  ……..

// create a ROOT  Proxy_Store
 pi_aida::Proxy_Store s1("hist.root","Root",true);
s1.write(h1);
s1.close();
// create a XML Proxy_Store
pi_aida::Proxy_Store s2("hist.xml",“XML",true);
s2.write(h1);
s2.close();
```



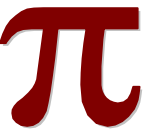
Features of AIDA_Proxy

- **All AIDA functionality is available** (excluding ITree)
- **Easy to use**
  - Hide factories to users
- **Value semantics**
  - Implemented operator “+” and “=“
  - Conversion (with ctor and operator “=“) from AIDA interfaces
- **Copy between implementations**
  - Anaphe -> Root and vice versa
- **Choose implementation at runtime**
  - User decides implementation when constructing the object
    - Use option in the constructor :
      - hA = Pi_aida::Histogram1D(title,nbin,min,max,”AIDA_Histogram_Native”)
      - hR = Pi_aida::Histogram1D(title,nbin,min,max,”AIDA_Histogram_Root”)



AIDA ROOT Implementation

- **AIDA_ROOT provides an implementation of AIDA Histograms**
  - Support now for 1D Histograms and Profiles
- **AIDA_Root::Histogram1D is a wrapper around a TH1D**
  - Use a developer interface layer
    - Creation by generic template factories
- **Histograms management:**
  - user managed
  - managed by an AIDA_ROOT Tree
    - implementation of ITree based on a Root File

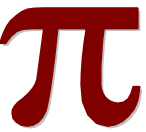


Example of AIDA_ROOT

- **Histogram creation using developer interfaces in PI AIDA_ROOT**

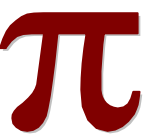
```
// create a factory
AIDA_CPP::GenFactory  * factory = new AIDA_ROOT::HistogramFactory;
// Create a histogram
 AIDA_CPP::IHistogram1D h1p = factory->create<AIDA_CPP::IHistogram1D>();
 h1p->initialize( "Example histogram.", 50, 0, 50 );
```

- **Advantage:**
  - Clean up of histogram factory
  - No need to create stubs for a partial implementation



Future evolution

- **Incorporate evaluation and feedback from users**
  - Propagate that to AIDA
- **Use AIDA developer interfaces**
  - Develop common utilities based only on developer interfaces
    - Copying between implementations, manipulators (projectors),…..
- **Integration with plotter**
  - Use OpenScientist and/or HippoDraw
- **Integration with experiment frameworks**
  - using SEAL component model (build an histogram service)
- **Integration with persistency services from POOL**
- **Use minimization library from Minuit C++ (from SEAL)**
- **Python binding to AIDA_Proxies**



PI releases

- PI latest release : 0.2.1
  - **Available on afs at**
    - **`/afs/cern.ch/sw/lcg/app/releases/PI/PI_0_2_1`**
  - **configuration**
    - **based on SEAL_0_3_2.**

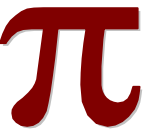
- Examples available on the Web

  http://lcgapp.cern.ch/project/pi/Examples/PI_0_2_1

- More information at the PI homepage:

  http://lcgapp.cern.ch/project/pi

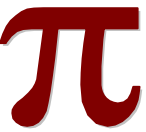
  - Reference documentation and code browser
  - talks...



Integration with External Tools (1)

Integration of AIDA and HippoDraw

- Prototype integration has been performed at the Python layer level
  - AIDA Histograms are created using Lizard
  - Simple Python program to copy the AIDA objects in HippoDraw compatible objects
    - Create an HippoDraw tuple with histogram/cloud/DPS information
  - use the Boost-Python interface to copy in and plot objects in HippoDraw
  - *Thanks to Paul Kunz for helping*



Integration with External Tools (2)

❖ Integration with ROOT using PyRoot

- PyROOT (former RootPython) from SEAL :
  - Python bindings for Root
    - Done using the ROOT dictionary
- AIDA objects are copied in Root objects at the Python level
- Example:
  - display an AIDA Histogram in a Root canvas from Lizard