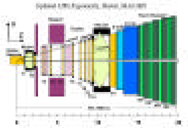
GAUDI Histograms

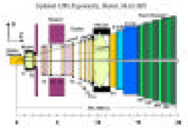
Pavel Binko

LHCb / CERN



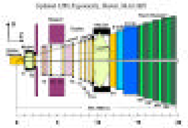
AIDA and LIZARD

- AIDA -- Abstract Interfaces for Data Analysis
  - Defines categories or packages (like in PAW)
    - Histograms, Vectors, Ntuples, Functions, Fitter, Plotter, Analyzer, Event display
  - There will be three sub-packages per category
    - Class definitions (e.g. all different histogram types, as 1D, 2D, etc.)
    - Factory - to allow creation of objects (of all classes defined above)
    - Manager - manipulates the objects above, steers a persistency
  - All classes in all categories will have a common messaging system
- All categories will have an abstract interface(s)
- LIZARD -- an AIDA compliant Interactive Analysis Environment
  - Should provide all the basic features of PAW (and more)
  - Based on AIDA specifications



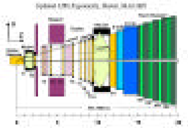
Basic principles

- Only interfaces, basic types, and types from foundation libraries are allowed
  - STL is currently the only one foundation library
  - Uses only int and double as basic types
- Most functions accept index as a parameter
  - The pre-defined values IHistogram::UNDERFLOW and IHistogram::OVERFLOW are also accepted
  - Conversion function from coordinate into index provided
- HTL internal classes or others do not appear in the interface



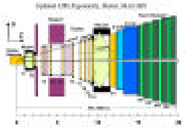
Class hierarchy

- IHistogram
  - Contains functions identical for both 1D and 2D histograms
  - User for histogram management (not visible to the users)
- IHistogram1D and IHistogram2D - “the” interfaces
  - Inherit from IHistogram
  - Contain 1D and 2D specific functions
- IAxis
  - Contains information about axis and its bins
    - Not the bin contents



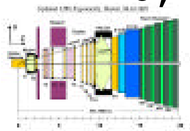
IHistogram

```
/// Constant specifying the underflow and overflow bin
enum { UNDERFLOW_BIN = -2, OVERFLOW_BIN = -1 };
/// Title of the histogram (will be set only in the constructor)
virtual std::string title() const                                         = 0;
/// Number of dimensions (1 for 1D histogram, 2 for 2D histogram, etc.)
virtual int dimensions() const                                            = 0;
/// Reset contents
virtual void reset()                                                      = 0;
/// Number of entries
virtual int entries() const                                               = 0;
virtual int allEntries() const                                            = 0;
virtual int extraEntries() const                                          = 0;
virtual double equivalentBinEntries() const                               = 0;
/// Sum of bin heights
virtual double sumBinHeights() const                                      = 0;
virtual double sumAllBinHeights() const                                   = 0;
virtual double sumExtraBinHeights() const                                 = 0;
```



IHistogram1D

```
/// Fill histogram
virtual void fill( double x, double weight = 1 )                          = 0;
/// Number of entries, bin height and bin error
virtual int binEntries( int index ) const                                 = 0;
virtual double binHeight( int index ) const                               = 0;
virtual double binError( int index ) const                                = 0;
/// mean and rms (calculated on filling-time)
virtual double mean() const                                               = 0;
virtual double rms() const                                                = 0;
/// Min height of in-range bins and index of the bin containing the minBinHeight()
virtual double minBinHeight() const                                       = 0;
virtual int minBin() const                                                = 0;
/// Max height of in-range bins and index of the bin containing the maxBinHeight()
virtual double maxBinHeight() const                                       = 0;
virtual int maxBin() const                                                = 0;
/// Get the X axis
virtual IAxis* xAxis() const                                              = 0;
/// Conversion from coordinate to index
virtual int coordToIndex( double coord ) const                            = 0;
```

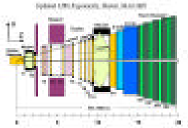


IHistogram2D (1)

```
/// Fill histogram
virtual void fill( double x, double y, double weight = 1 )                = 0;
/// Number of entries in a bin and projections on the axis X/Y
virtual int binEntries( int indexX, int indexY ) const                    = 0;
virtual int binEntriesX( int indexX ) const                               = 0;
virtual int binEntriesY( int indexY ) const                               = 0;
/// Height of a bin and projections on the axis X/Y
virtual double binHeight( int indexX, int indexY ) const                  = 0;
virtual double binHeightX( int indexX ) const                             = 0;
virtual double binHeightY( int indexY ) const                             = 0;
/// Bin contents error
virtual double binError( int indexX, int indexY ) const                   = 0;

/// mean and rms (calculated on filling-time) projected on the axis X/Y
virtual double meanX() const                                              = 0;
virtual double meanY() const                                              = 0;
virtual double rmsX() const                                               = 0;
virtual double rmsY() const                                               = 0;
```

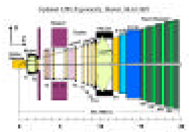
IHistogram2D (2)

```
/// Min height of in-range bins and index of the bin containing the minBinHeight()
virtual double minBinHeight() const                                       = 0;
virtual int minBinX() const                                               = 0;
virtual int minBinY() const                                               = 0;

/// Max height of in-range bins and index of the bin containing the maxBinHeight()
virtual double maxBinHeight() const                                       = 0;
virtual int maxBinX() const                                               = 0;
virtual int maxBinY() const                                               = 0;

/// Get the X/Y axis
virtual IAxis* xAxis() const                                              = 0;
virtual IAxis* yAxis() const                                              = 0;

/// Conversions between coordinates and bin indices
virtual int coordToIndexX( double coordX ) const                          = 0;
virtual int coordToIndexY( double coordY ) const                          = 0;
```



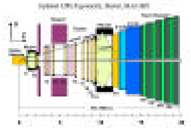
IHistogram2D (3)

```
/// Projection on axis X/Y
virtual IHistogram1D* projectionX() const                                 = 0;
virtual IHistogram1D* projectionY() const                                 = 0;

/// Slice parallel with the axis X, identified by bin indexY
virtual IHistogram1D* sliceX( int indexY ) const                          = 0;

/// Slice parallel with the axis Y, identified by bin indexX
virtual IHistogram1D* sliceY( int indexX ) const                          = 0;

/// Slice parallel with the axis X, between indexY1 and indexY2
virtual IHistogram1D* sliceX( int indexY1, int indexY2 ) const            = 0;

/// Slice parallel with the axis Y, between indexX1 and indexX2
virtual IHistogram1D* sliceY( int indexX1, int indexX2 ) const            = 0;
```



IAxis

```
/// Lower and upper axis edge
virtual double lowerEdge() const                                          = 0;
virtual double upperEdge() const                                          = 0;

/// Number of in-range bins in the axis
virtual int bins() const                                                  = 0;

/// Lower and upper edge of the in-range bin identified by index
virtual double binLowerEdge( int index ) const                            = 0;
virtual double binUpperEdge( int index ) const                            = 0;

/// Width of the in-range bin identified by index
virtual double binWidth( int index ) const                               = 0;

/// Centre of the bin located by index
virtual double binCentre( int index ) const                               = 0;

/// Conversions between coordinates and bin indices
virtual int coordToIndex( double coord ) const                            = 0;
```



Conclusions

- Designed AIDA histogram interfaces in C++ and Java
  - “Agreed” by large community of developers
    - via the HepVis and LHC++ mailing lists
  - LHC++, LIZARD, OpenScientist, JAS, etc.
  - LHC experiments interested in it: ATLAS, CMS, LHCb
- All AIDA histogram interfaces implemented in GAUDI together with the HistogramSvc
  - Using the Histogram Template Library (HTL) by LHC++
  - Switch to an other histogram package rather simple
- Implemented HBOOK convertors
  - Create HBOOK histograms, fill them and propagate statistics