Anaphe Developer Interfaces

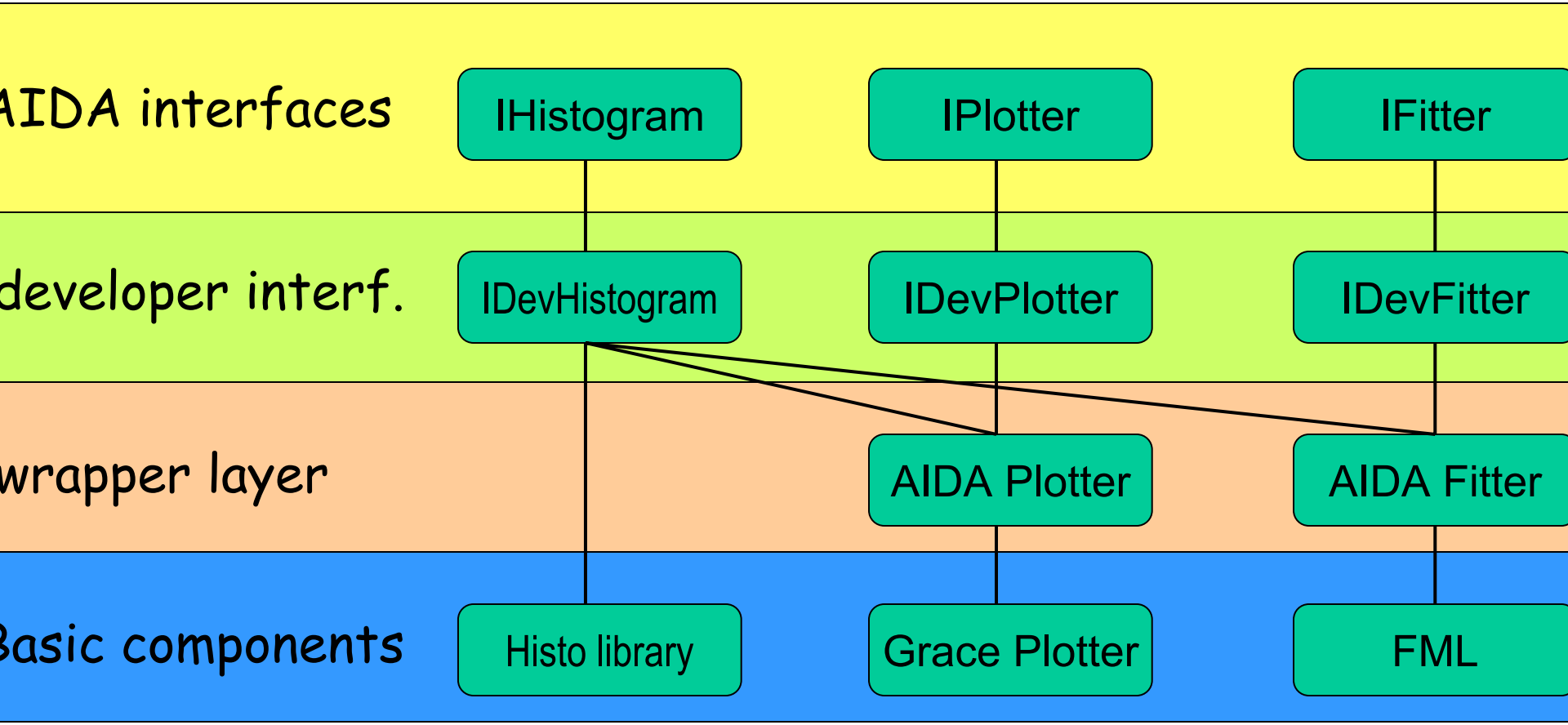
Lorenzo Moneta

CERN

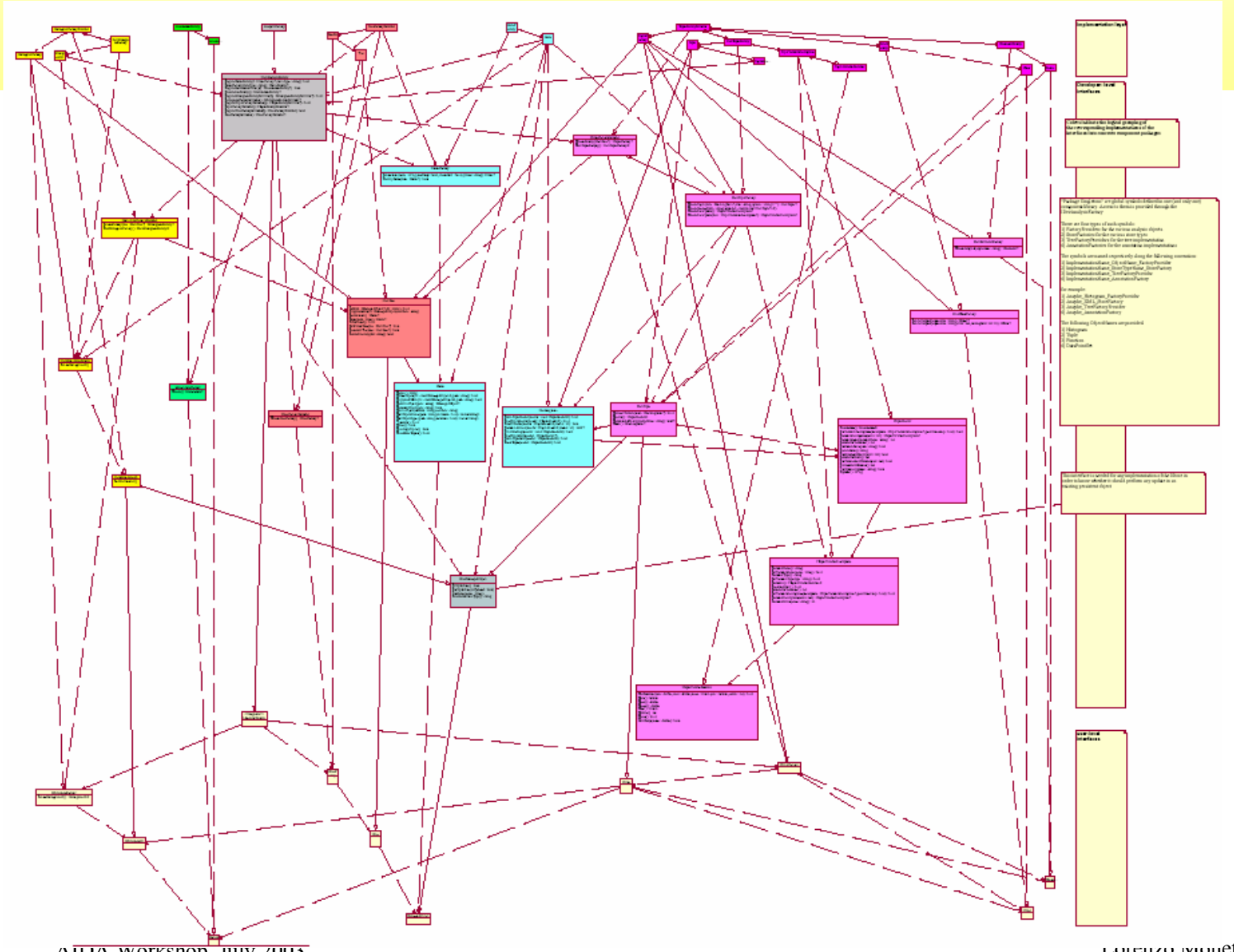
AIDA Workshop

3/7/2003

Anaphe Architecture



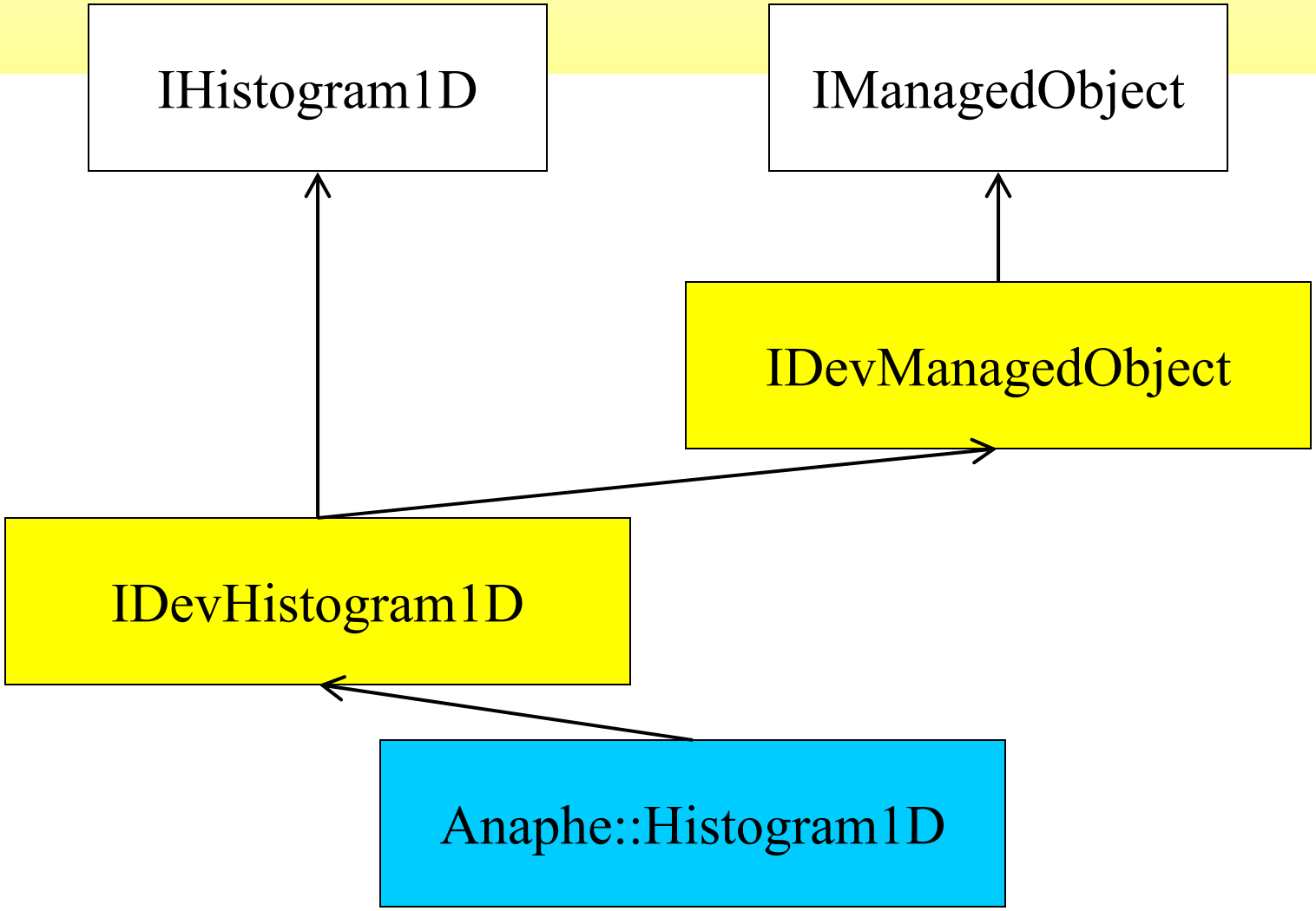
Design



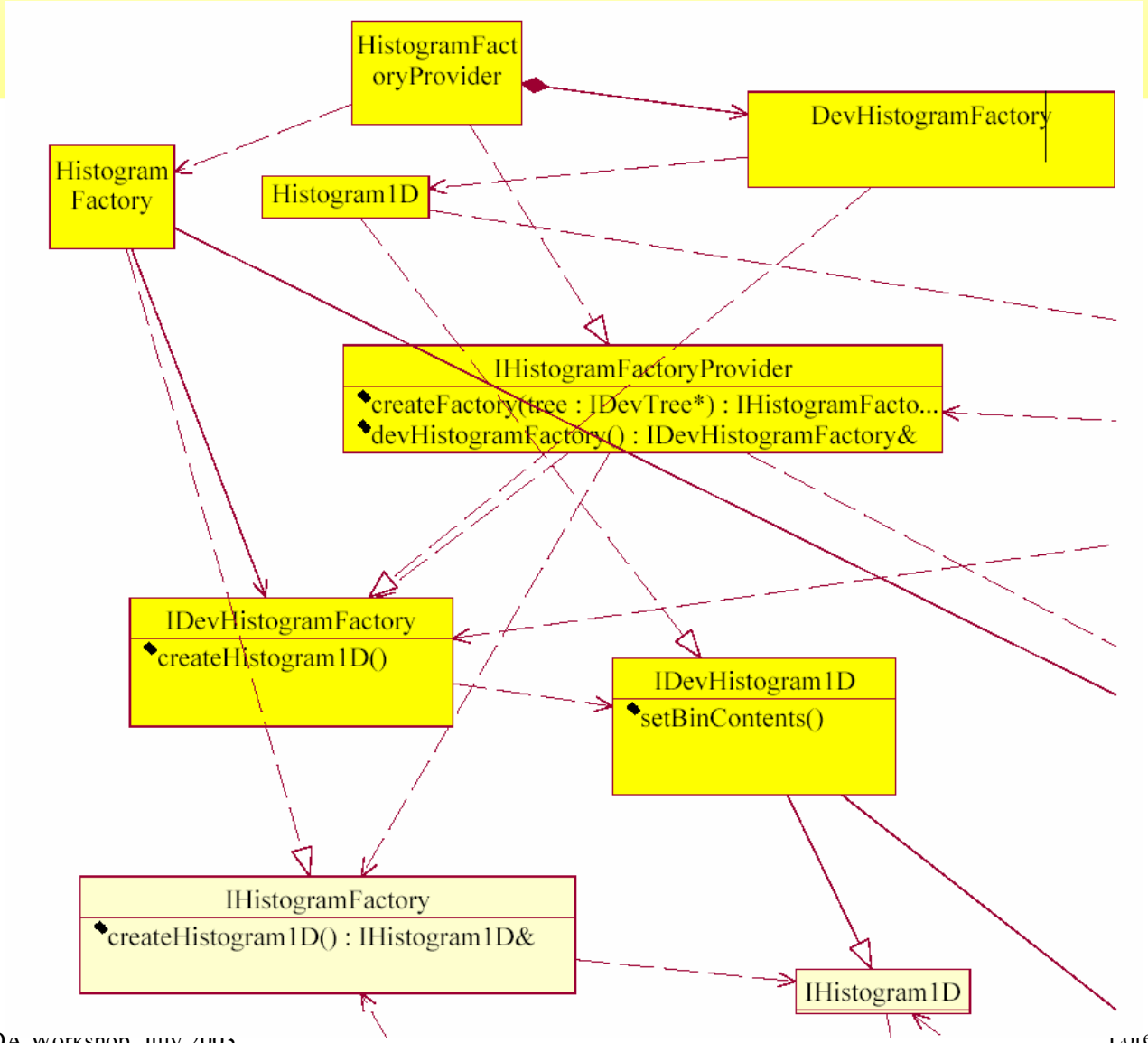
Histogram Developer Interfaces

- **Histogram Developer interface:**
  - **IDevHistogram** :
    - Inherits from IHistogram1D and IDevManagedObject
    - Methods:
      - bool setBinContents(iBin, entries, height, error , centre);
      - bool setRms(rms);
  - **IDevHistogramFactory**
    - no inheritance from IHistogramFactory
    - Factory to create unmanaged objects
    - Same methods as in IHistogramFactory but without the name
      - IDevHistogram1D createCopy(const IHistogram1D & h);
  - Anaphe::HistogramFactory implements IHistogramFactory using the IDevHistogramFactory

Inheritance tree for Anaphe::Histogram1D



Histogram Dev interfaces



IDevManagedObject

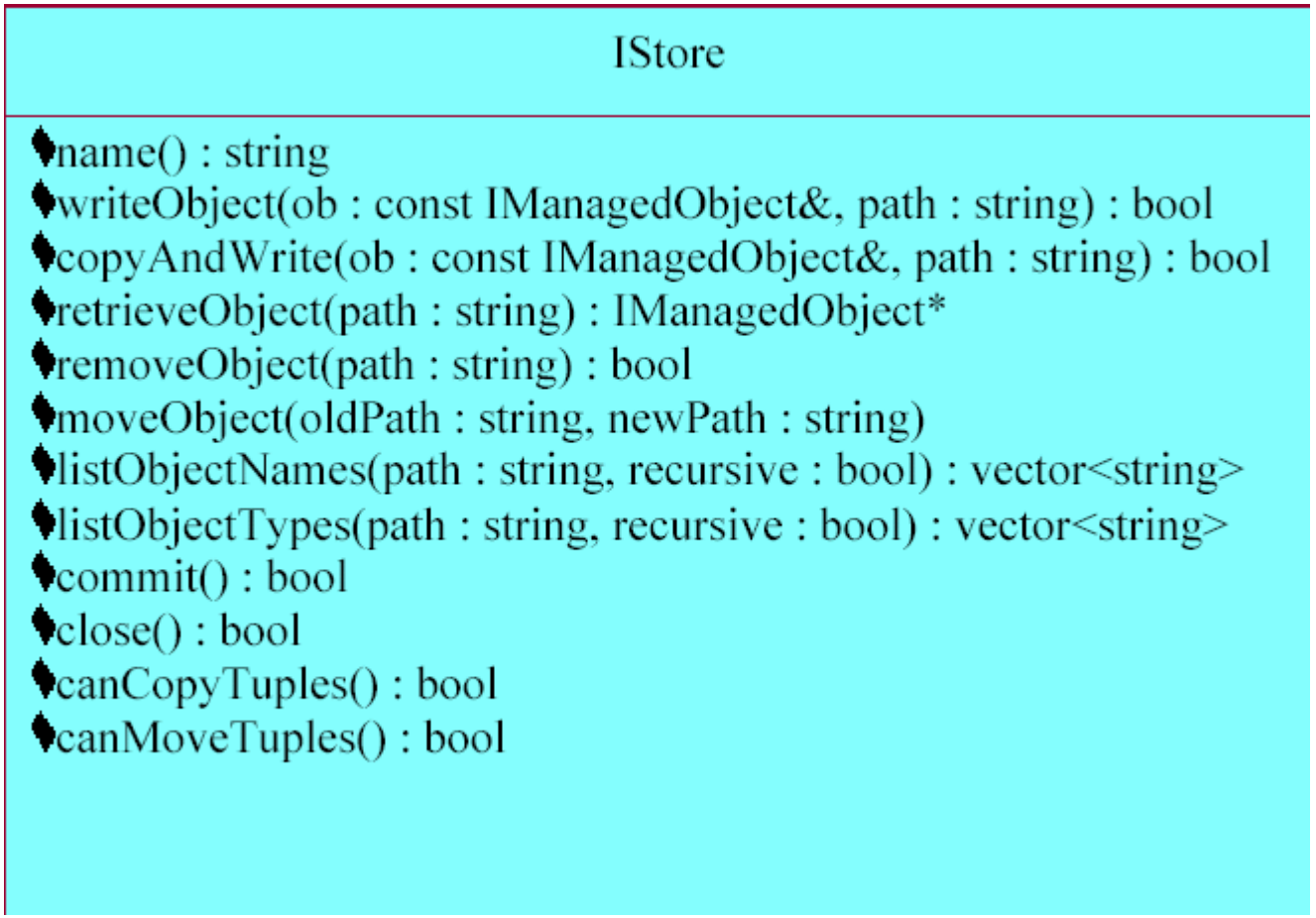
- **IDevManagedObject**
  - All developer interfaces for objects which are going to be stored inherits from IDevManagedObject
    - Store needs to know if an update is needed of an existing persistency object
  - Methods:

```
bool isUpToDate();
void setUpToDate( bool isUpToDate );
bool setName( const std::string& newName );
std::string userLevelClassType();
```

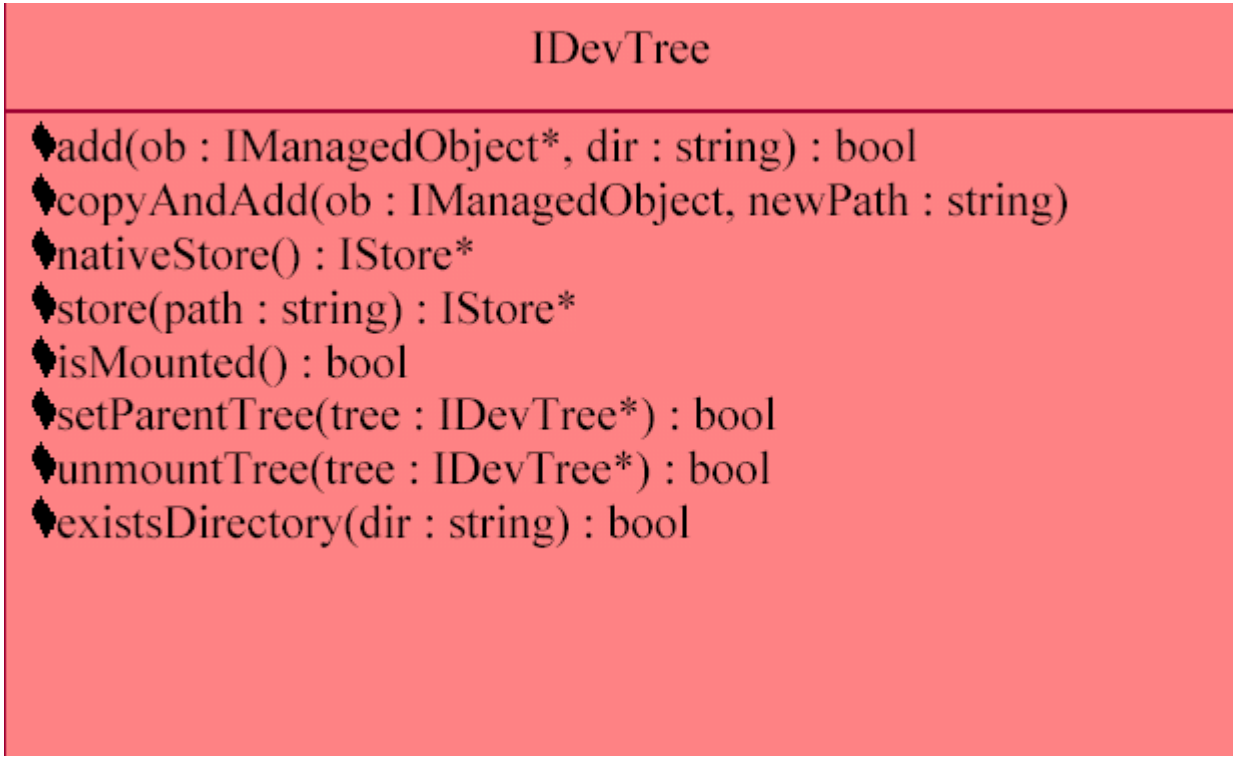
Anaphe Store

- **Tree is separated from Store implementation**
  - Depends only on developer interface IStore
  - Store does not depend on the Tree
- **No dependency between Store and any particular implementation of the data objects**
  - Store deals only with IDevHistogram, IDevClouds, IDevTuple, etc…
    - Copy them in the corresponding persistency objects when writing
    - Use developer factory interface to create them when reading
      - No need to use tree to create a IDevHistoframFactory

IStore developer interface



IDevTree interface

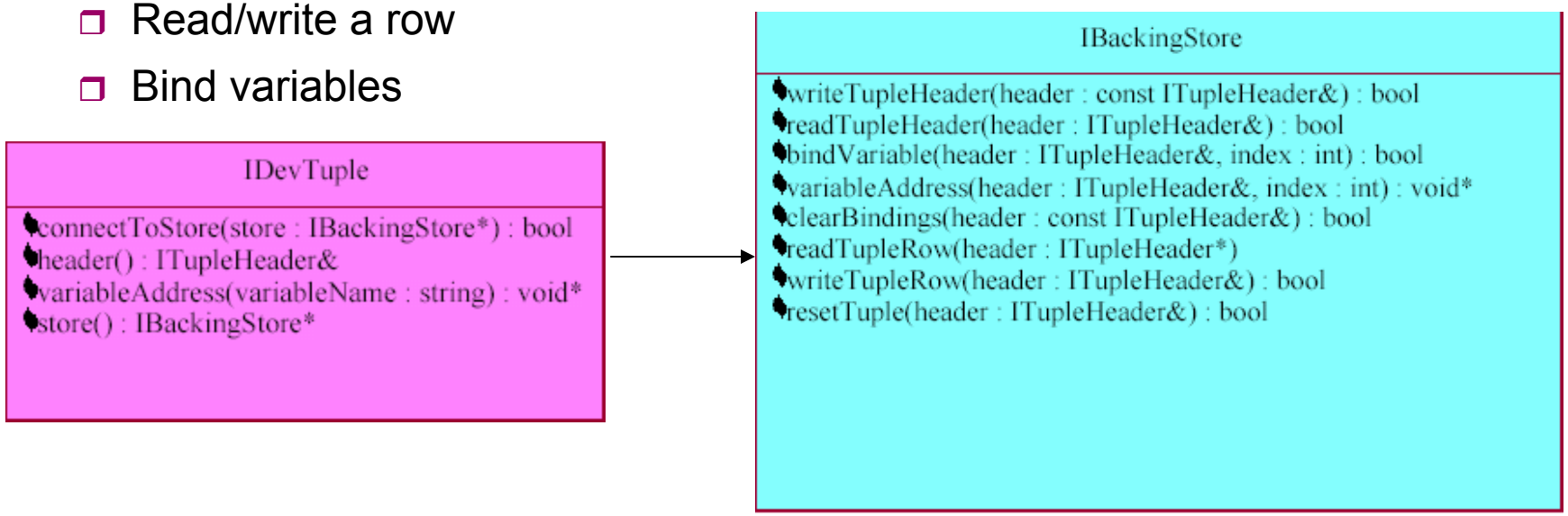


Factory Providers

- **Each component library defines a unique global symbol :**
  - FactoryProvider for all the various data objects
    - Histograms, Tuples, Functions and DataPointSet
  - StoreFactories fo each store implementation
  - TreeFactoryProvider for the tree
  - AnnotationFactory for the annotation implementation
- **Access to the symbol is provided through the IDevAnalysisFactory interface**
  - When the library is loaded it registers in the AnalysisFactory
  - No dynamic loaded is supported but can be easley added, by implementing the AnalysisFactory with a PluginManager

Tuples developer interfaces

- **IDevTuple**
- **IBackingStore**
  - Read/write a row
  - Bind variables



Other Tuple developer Interfaces:
ITupleHeader, ITupleVariableDescription, ITupleVariableStatistic

Functions and Fitting

- IDevFunction
- IDevFunctionCatalog
- IDevFunctionFactory

- IDevFitter
- IDevFitData
- IDevFitDataIterator
- IDevFitResult
- IDevFitParameterSettings

Conclusions

- **Need developer interfaces for AIDA objects :**
  - Setter methods for efficient copying
  - Some common properties for storing and plotting
    - E.g. isValid() ?
  - Type information ?
- **Need to be able to create unmanaged objects**
  - IDevFactories ?
  - Can we remove managedObject from user interface ?
    - Possibility to leave management to specific implementations
    - Now IManagedObject appear only in
      - IManagedObject * find(std::string path);
      - std::string findPath(const IManagedObject & obj);
- **Store interface**