

# *AIDA MetaData*

- Data-selection Workflow
- AIDA
  - Overview
  - Development Process
- Persistency API
- Tools
  - Anaphe
  - Open Scientist
  - FreeHEP
  - Root
- Objects
  - Tuples
  - CANs

# *Data-selection Workflow*

- Dataset selection using high-level metadata (AMI, etc.)
- **Event selection using Event-level metadata. Requires complex searching.**
- Event-based Analysis.

# *AIDA - Overview*

- Abstract Interfaces for **Data Analysis**
- Objects generally created by factories.
- Statistics:
  - Histograms, Clouds, Profiles, DataPointSets
- **(Meta)Data:**
  - **Tuples**
- Storage:
  - Trees, Directories
- Graphics:
  - Plotters
- Mathematisch:
  - Functions, Fitters

# *AIDA – Development Process*

- Several implementations of the same Abstract Interfaces (Java, C++, Python,...).
- Each implementation team does independent prototyping, prototypes are then discussed and common interfaces agreed. Currently discussed are:
  - IStore (several prototypes)
  - Value semantics (Anaphe prototype)
  - Tuple improvements
- Workshop every sever month, last one three weeks ago (nobody fom Atlas), next one in October-overmber.
- **NOT the least common denominator, but fully implemened rich interfaces based on experience of several teams.**
- **Lanuage neutral:**
  - Abstract interfaces written in AID pseudo-language.
  - Java and C++ interfaces created automaticaly.
  - Python interfaces not yet created automaticaly, each implementation has their own (to be consolidated).
- **Inter-operable:** Is is possible to create AIDA objets in one AIDA implementation store them using another one and display using another one (all that in one executable).
- Currently version 3.0.

# *Services – Persistency API*

- Objects can be Managed or NonManaged.
  - Managed objects are part of a Tree. Tree looks very much like Unix filesystem. Branches can be linked or mounted (even from remote servers – similar to afs/nfs/...). Tree handles persistency.
  - Persistency of NonManaged objects has to be handled individually.
- Supported persistency formats:
  - **XML**: all implementations
  - **HBook**: all implementations
  - **SQL**: FreeHEP via JDO, OpenScientist directly
  - **JDO**: FreeHEP; supports all existing databases
  - **Root files**: all implementations (FreeHEP and OpenScientist independent on Root implementation); unresolved question whether to store AIDA objects (not understandable by Root) or Root objects (more primitive) .
- **SQL Interoperability (under discussion):**
  - **Common SQL schema**
  - **SQL scripts for create/write/read/search/delete stored themselves in the database; they have common interfaces.**
  - **Full Client-Server operation**

# *Tools - Anaphe*

- Originally developed within LHC++ & Lizard, now adopted by LCG/PI.
- C++ with Python binding.
- Prototypes for:
  - Value-semantics API (based on top of standard API): creation by factories replaced with creation by constructors.
  - **CANs (Canned Analysis)**: containers of related AIDA objects together with their properties and instructions for construction and use.
  - **Native Root storage (using Pool & Seal)**.
  - Graphics using HippoDraw.
- Not the most functional implementation.

# *Tools – Open Scientist*

- Standalone implementation from LAL.
- C++ with many other bindings.
- Prototypes for:
  - **Stand-alone C++ Root IO** independent on Root.
  - **SQL IO** (interoperable with FreeHEP).
- Very functional, harder to use.

# *Tools – FreeHEP*

- Part of extensive FreeHEP library (contains also JAS, Wired, HepRep, RootIO,...), result of wide open collaboration. Hosted at SLAC.
- Java with C++ interface and other bindings.
- Prototypes for:
  - Direct storage (just about any storage format supported directly, via JDO or both).
  - **Client-Server operation.**
  - **Grid integration.**
  - **Web Service integration (using standard Web Service protocols).**
- By far the most complete and usable implementation.

## *Tools – Root*

- No native support for AIDA.
- All other implementation can read/write Root analysis objects in Root files:
  - Anaphe via Root/Pool/Seal.
  - Open Scientist via its own implementation (Rio Grande).
  - FreeHEP via its own implementation (hep.io.root).
- Root analysis objects are more limited and have much worse design than AIDA objects. Currently, AIDA implementation are storing Root analysis objects, but this leads to the loss of information.

# Objects - Tuples

- Tuples are generalisation of Paw/Root ntuples.
- They can contain any primitive number, some more complex objects, another Tuples or any other Object (in case of FreeHEP implementation).
- They allow standard analysis operations, like cuts (queries, searches), projections to other analysis objects, ...
- **Tuples stored in SQL DB profit from SQL query optimisation. Cuts on Tuples can be expressed in a natural (Java/C++/Python/...) way; they are translated to SQL internally.**
- **Tuple architecture is very close to the Pool Event AttributeList architectures with several missing features (to be implemented before next AIDA workshop):**
  - The Tuple schema/signiture doesn't exist as a standalone object. Tuple schema is defined on the Tuple itself. Also, Tuples can only be filled by single numbers not by complete entry objects.
  - Tuple columns are addressable only by numbers, not by names.
  - Tuple can't store Ref<T>.

# *Objects - CANs*

- **CANs (Canned Analysis): containers of related AIDA objects together with their properties and instructions for construction and use.**
  - Can serve as a replacement for Atlas Common ntuple.
  - Can be used by DIAL to communicate analysis between servers.
  - Can be wrapped in Mobile Agents.
- Very early prototypes exist in Anaphe.

# *Conclusion*

- Effective selection of Events requires high level analysis functionality. Users will require real analysis functionality on Event Metadata.
- **The Interface of AIDA Tuple corresponds very well to required Event Metadata Interface.**
- AIDA provides all required functionality, including:
  - Graphics analysis.
  - Persistency (XML, SQL, Root,...).
  - Distributed operation.
  - Grid integration.
- AIDA is natively supported by three HEP general frameworks, independent implementations exist for another framework.