π Project
Status and Plan

IX International Workshop on
Advanced Computing and Analysis Techniques
in Physics Research
KEK, Tsukuba, December 2003

Andreas Pfeiffer

CERN/EP

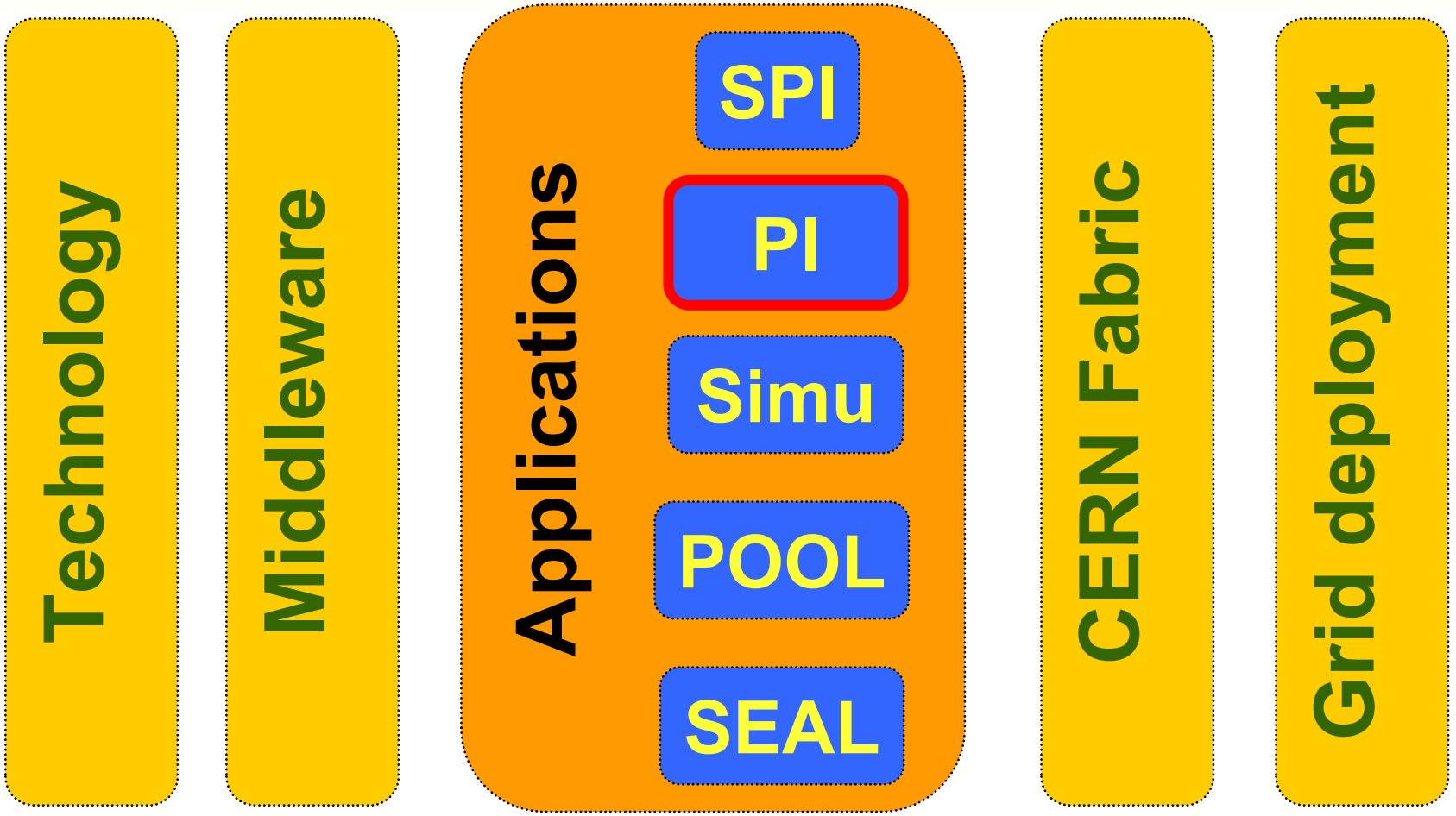
On behalf of the PI project team

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

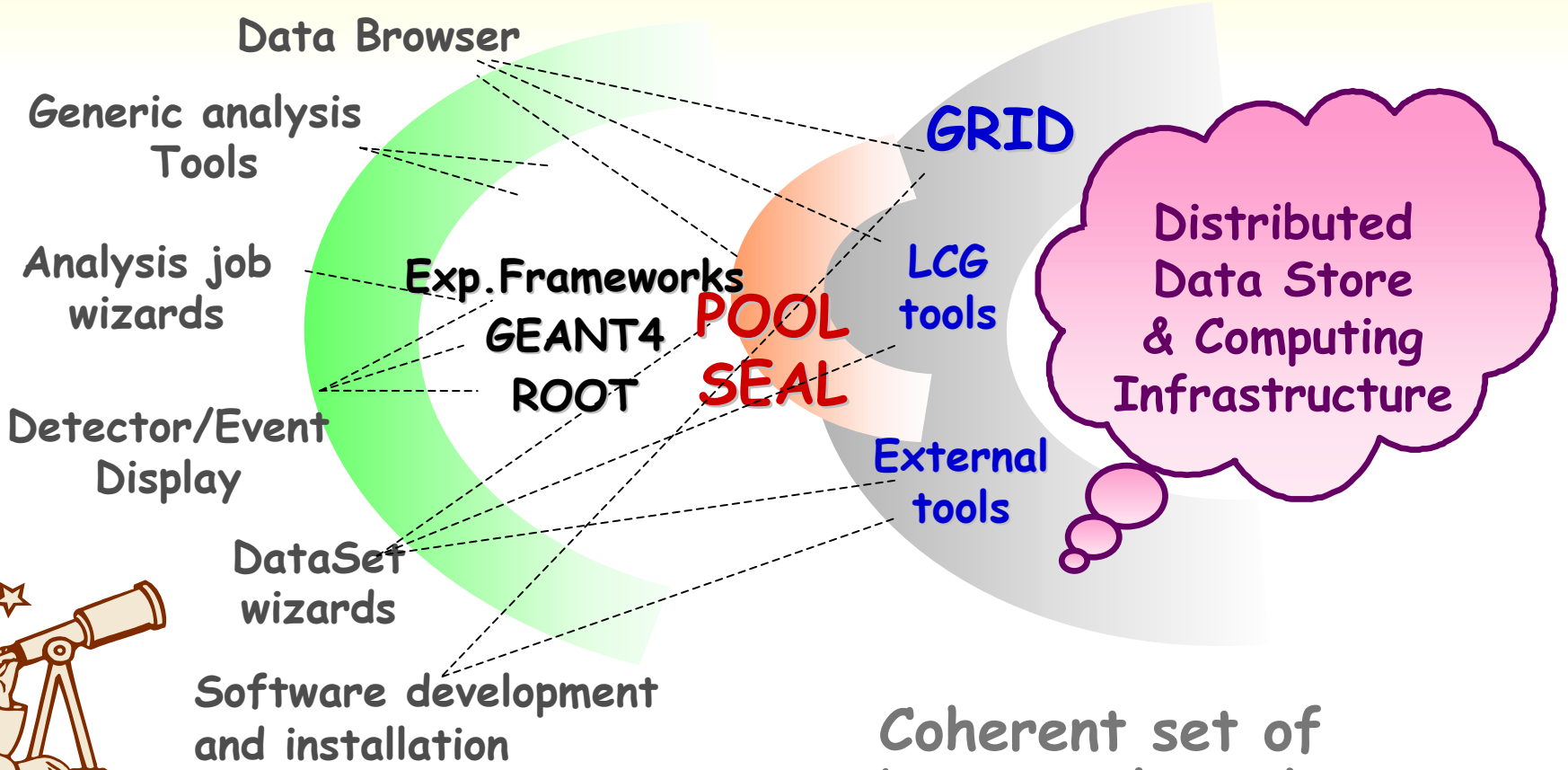
LCG Physicist Interfaces project

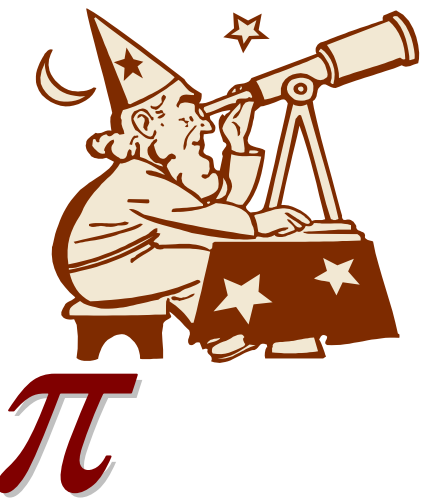
- **LHC Computing Grid (LCG) Application Area project**
  - **"Architecture Blueprint" RTAG identified the need for a *consistent set of interfaces and tools* by which physicists will directly use the software.**
- **Physicist Interface (PI) started end `02**
  - Review with experiments to define workplan
  - Project proposal to SC2 end Jan`03, reviewed in July`03

PI in the LCG Application Area



Architecture Overview





Consistent User Interface

Coherent set of basic tools and mechanisms

LCG Physicist Interfaces project

- **Five working areas identified which the interfaces should cover**
  - Analysis Services
  - Analysis Environment
  - Pool & Grid PI
  - Event & Detector Visualization
  - Infrastructures & Documentation
- **Not all items have same priority:**
  - Analysis Services first

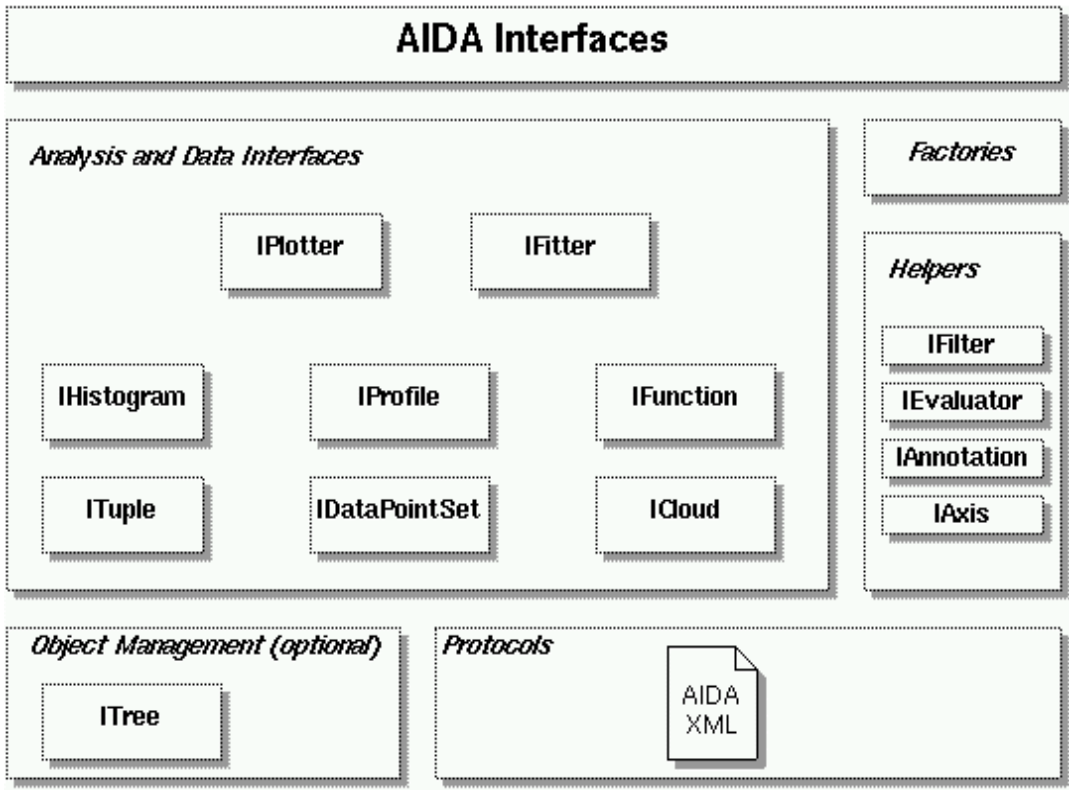
Analysis Services

- **AIDA**
  - Review, adapt, extend Interface to Data Analysis
- **Root implementation of AIDA**
  - Provide an implementation of the Interfaces to Data Analysis, as defined by AIDA, based on Root (using a wrapper to Root)
- **AIDA interface to SEAL and POOL services**
  - Use SEAL and POOL to provide services such as plugin manager, object whiteboard and persistency.
- **Blueprint compliant Analysis tool set**
  - Statistics analysis tools based on AIDA interface
    - Mainly external contributions
- **Maintenance issue: Re-*use* – don't re-*invent* !**
  - Integration of existing s/w

AIDA

AIDA – Abstract Interfaces for Data Analysis

- Started in late 1999 (HepVis WS)
- Open collaboration, teams from
  - **PI (CERN)**
  - **JAS (SLAC)**
  - **OpenScientist (LAL)**
- Version 3.0 since Oct. 2002
  - **User level interfaces**
  - **Pointers to objects with factories**
  - **XML protocol for data exchange**
  - **V-3.2 (Oct 2003): small changes**
- Added value from PI:
  - **Simplified value-semantic layer with constructors and operators**
  - **Developer-level interface to ease building generic manipulators and tools**



PI Analysis Services

Experimental Frameworks

Exposing/Using AIDA IFs

PI_AIDA

SEAL

POOL

Analysis Tools

AIDA

Analysis object manipulators

Root (wrapper)

Native

Open Scientist

JAS

The Goals

- **Interoperability**
  - Project any AIDA 2D histogram on an AIDA 1D histogram of different implementation
    - achieved using PI Proxy
  - Use different AIDA fitters with different implementations of histograms
  - Exchange histograms between ROOT (C++) and Jas (Java) (XML format)
- **Extensions**
  - Generic manipulator such as projectors, rebinners, etc
  - Build and manipulate aggregate of objects (CANs)
- **Interface to external applications**
  - Store AIDA histograms in POOL (connected to experiment specific data)
  - Display AIDA histograms using external tools such as HippoDraw or EXCEL
- **Framework to develop complex analysis tools**
  - Statistical comparison of data-sets
  - Modelling parametric fit problems using a MonteCarlo approach

Features of AIDA_Proxy

- All AIDA functionality is available
  - excluding ITree
- Easy to use
  - Hide factories from users
- Value semantics
  - Implemented operator “+” and “=“
- Copy between implementations
  - AIDA-native to AIDA-Root and vice versa
- Choose implementation at runtime
  - User decides implementation when constructing the objets
- Objects are managed by the user (not by AIDA Tree)
  - Easier integration with other frameworks

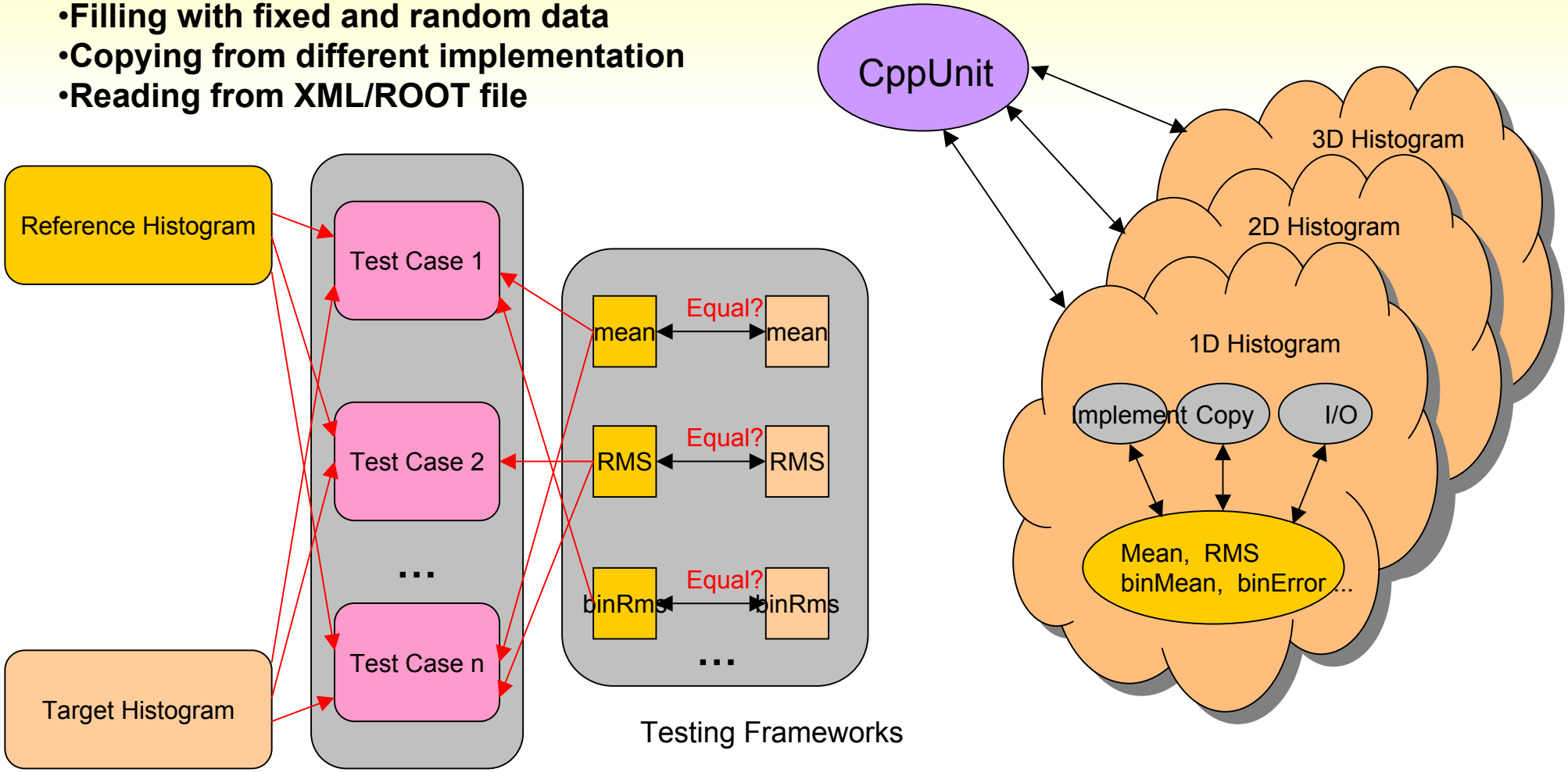
Latest Release: 1.1.0 – Main Features

- **Dynamic loading of implementation:**
  - AnalysisServices/AIDA_Proxy is a ***proxy*** to the AIDA interfaces.
  - Using the SEAL PluginManager to choose at runtime the corresponding implementation of the AIDA interfaces.
- **ROOT implementation**
  - Providing an AIDA implementation for Root Histograms. All types of ***binned*** Histograms (1-,2-,3-D) and Profiles (1-,2-D).
- **Flexible I/O system**
  - *Proxy_Store* giving the user the possibility to read/write AIDA objects (***all*** types of Histograms, plus DataPointSets and Tuples) in a compressed XML file using the AIDA XML format.
    - Used as exchange format among implementations
  - Other formats (**reading and writing**) for *binned* histograms and Profiles
    - **ROOT files, HBOOK files**
  - ➔ Easy *conversion* between all formats
- **UnitTest Framework**

Unit testing

Histogram Sources:

- **Filling with fixed and random data**
- **Copying from different implementation**
- **Reading from XML/ROOT file**

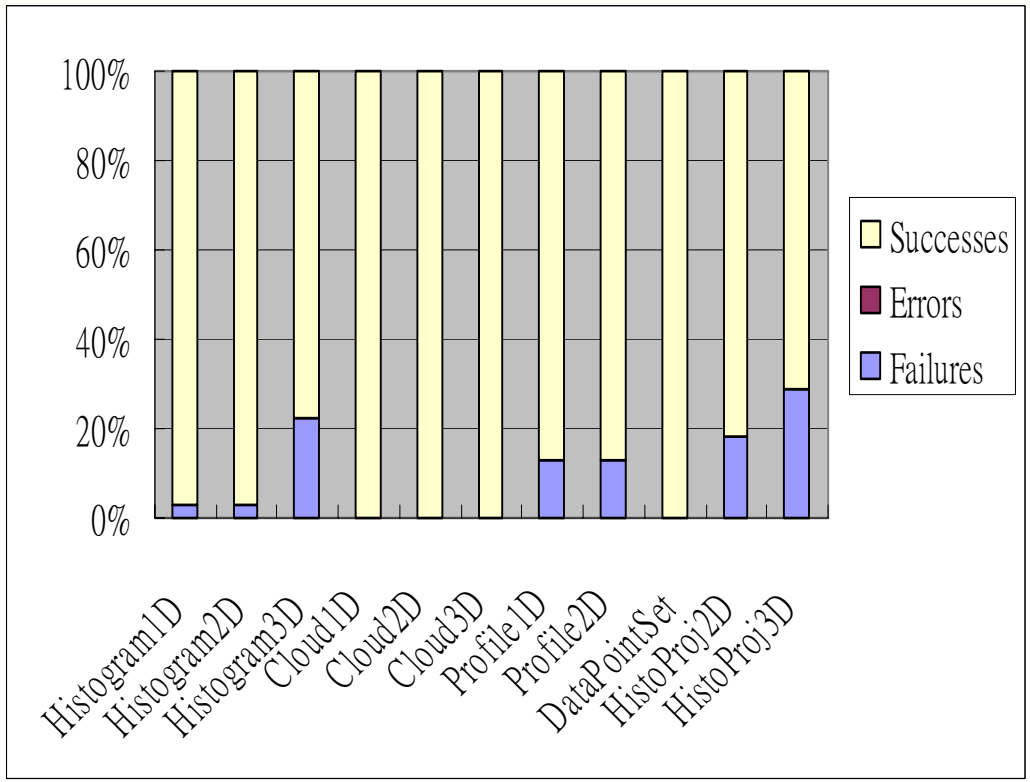


Hurng-Chun Lee

Academia Sinica Computing Centre, Taiwan

Testing Results

- **Purpose**
  - Check the functionalities of AIDA Proxy
  - Check the consistencies between different histogram implementations through AIDA Proxy
  - Develop a unit test framework for AIDA Proxy on which test-cases can be easily adapted
- **1164 CppUnit assertions**
  - 104 (<10%) failures
- **Failures are due to *implementation differences***
  - Root takes the binCentres instead of the values to calculate the global mean in H3D and Profiles
  - Root doesn't store the binMean
  - Error treatment in Profile



Hurng-Chun Lee

Academia Sinica Computing Centre, Taiwan

Integration with External Tools (1)

❖ Integration of AIDA and HippoDraw

- Prototype integration performed at the Python layer
  - AIDA Histograms are created and filled using AIDA Python bindings
    - **Bindings to AIDA generated with LCGDictionary (SEAL)**
  - Simple Python program to copy the AIDA objects in HippoDraw compatible objects
    - **Create an HippoDraw tuple from AIDA analysis objects**
    - **Can plot also AIDA Clouds and DataPointSets**
  - 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

- Bridge to Root from Python
- Use Python bindings to Root (PyROOT) from SEAL
  - Done using the Rootcint dictionary
- AIDA objects are copied in Root objects at the Python level
- Example:
  - display AIDA Histograms in a Root canvas from Python

Near Term Plan

- Port to Windows
  - End of 2003
- Conformance to AIDA version 3.2
  - End of 2003
- Interoperability with other existing AIDA implementations
  - OpenScientist(C++) and JAIDA/AIDAJNI (Java)
  - February 2004
- Implement fitting and fit tests
  - February 2004
- Performance checking infrastructure
  - March 2004

Summary

- PI provides Interfaces (and implementations) for Physicists
  - Tool set compliant with the “Architectural Blueprint” of LCG AA
- Analysis Services component developed
  - Component model, based on AIDA
  - Flexible, plug-in controlled set of implementations
    - Making use of SEAL project
  - Large number of unit tests show differences in implementations
  - Integration with external components/tools easily possible
    - Prototypes in Python connecting AIDA with HippoDraw and Root
- Review of AIDA completed
  - Feedback to AIDA collaboration
- Development on schedule
  - First release in May, production version 1.0.0 early october
  - Latest release 1.1.0 end November

More information

- On the PI project :
  - http://cern.ch/pi
- On AIDA:
  - http://aida.freehep.org
- On LHC Computing Grid:
  - http://cern.ch/LCG