AIDA, JAIDA AND AIDAJNI: DATA ANALYSIS USING INTERFACES

M. Donszelmann, SLAC, Stanford, CA 94309, USA
T. Johnson, SLAC, Stanford, CA 94309, USA
V. Serbo, SLAC, Stanford, CA 94309, USA
M. Turri, SLAC, Stanford, CA 94309, USA

Abstract

AIDA, Abstract Interfaces for Data Analysis [1], is a set of abstract interfaces for data analysis components: Histograms, Tuples, Functions, Fitter, Plotter and other typical analysis categories. The interfaces are currently defined in Java, C++ and Python and implementations exist in the form of libraries and tools using C++ (Anaphe/Lizard [2], OpenScientist [3]), Java (JAIDA [4]) and Python (PAIDA [5]).

This paper will describe AIDA in more details and will focus on Java implementation of AIDA, which is part of FreeHEP Java library [4]. FreeHEP library also include many other useful utilities; this paper will describe two of them. AIDAJNI – library that enables C++ program to use Java implementation of AIDA, and AIDA Tag Library – set of HTML-like tags that allows user to easily include live plots into the web pages.

AIDA

AIDA [1] has been created cooperatively by a group of developers working on high-energy physics data analysis tools. The goal of the AIDA project is to provide user with a powerful set of interfaces which can be used regardless of which analysis tool they are using. The idea is to define only a “protocol” for analysis objects, no internal details. AIDA defines the behaviour; analysis tool provides the implementation (see Figure 1).

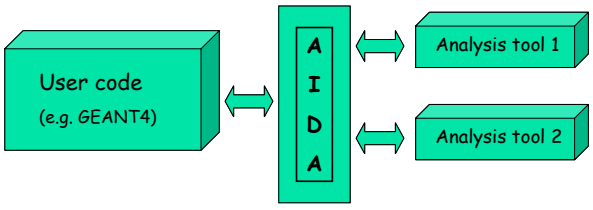

Figure 1: Usage of AIDA.

AIDA also defines XML format for representation and storage of analysis objects.

When using AIDA, users only have to learn one set of interfaces, but use any available AIDA-compliant tool.

All AIDA interfaces are defined in a language-independent format, and later the C++, Java, and Python interfaces are generated automatically from this one source using the AID (Abstract Interface Definition [6]) tool. This setup makes it easy to add new language and to maintain consistent set of interfaces across different languages.

Any AIDA program begins with Factories. Instead of creating objects directly using "new" one uses factories. There is one "master" factory, IAnalysisFactory from which other factories are obtained. The IAnalysisFactory allows you to obtain factories for creating Trees (ITreeFactory), Histograms, Clouds and Profile Histograms (IHistogramFactory), Tuples (ITupleFactory), etc. Please have a look at the piece of code below:

```
IAnalysisFactory af = IAnalysisFactory.create();
ITreeFactory tf = af.createTreeFactory();
ITree tree = tf.create();
IHistogramFactory hf = af.createHistogramFactory(tree);

IHistogram1D h = hf.createHistogram1D("test",50,0,5);
….
```

Using Factories makes user code longer by several lines, but it provides a guarantee that AIDA program will work with any implementation, since it does not explicitly rely on any concrete implementation-specific classes.

The AIDA ITree interface provides two capabilities: the ability to group analysis objects such as Histograms, Clouds, Tuples, etc. into hierarchical directories (or folders), and the ability to save and restore sets of analysis objects into files or databases.

AIDA supports 1D, 2D, and 3D Histograms, Profile Histograms, and Clouds. Clouds are unbinned collections of data. They are used for scatter plots or dynamically rebinnable Histograms. A Cloud can be automatically converted to a Histogram when the number of entries exceeds a given threshold, or can be manually converted by the user.

Other AIDA interfaces deal with creation and manipulation of Tuples, DataPointSets, Functions, and Plotting of analysis objects.

The AIDA IFitter interface provides the user the possibility to fit Functions to any AIDA data storage object. Binned fits can be performed on Histograms, Profile Histograms and DataPointSets, while unbinned fits can be performed on Clouds and Tuples. Simple fits

can be performed directly on the data storage objects while the IFitData interface is to be used for a greater control over the data, in particular its ranges and the connection to the variables a Function. Through the IFitter it is also possible to change the underlying optimization engine as well as the fit method used.

More information about AIDA, including the full list of interfaces and complete Users Guide can be found on the AIDA home page [1].

The current stable version of AIDA, implemented in C++, Java, and Python is 3.2.1. There are ongoing efforts to improve existing interfaces, as well as prepare the next major upgrade (AIDA 4).

JAIDA

JAIDA is the full implementation of AIDA in Java. It is part of FreeHEP Java library [4], and is used internally by JAS3 (Java Analysis Studio [7]) as its analysis core. It can also be used independently for either batch or interactive processing, or for web applications to access data, make plots and simple data analysis through a browser.

Some of the JAIDA features are the ability to open ASCII text, AIDA XML, ROOT, and HBOOK data files and the support of an extendable set of fit methods (chi-square, least squares, binned/unbinned likelihood, etc.) to be matched with an extendable set of optimizers including Minuit and Uncmin.

Also JAIDA provides full support for AIDA plotting as well as ability to save plots in a variety of high quality graphics formats including: PDF, EPS, SVG, SWF, PNG, GIF, JPG, etc. Live JAIDA plots can be embedded in other Java GUIs or be used in servlets for web-based applications.

JAIDA is ideal for batch analysis and jobs that don't require GUI functionality. For interactive analysis it is better to use JAS3 – a general purpose, open-source, interactive data analysis tool. JAS3 uses JAIDA internally for analysis, but adds GUI interactivity. It can be easily extended by writing custom modules. JAS3 also includes support for scripting, currently Pnuts and Python.

The current version of JAIDA is 3.2.3, implementing AIDA 3.2.1.

AIDAJNI

AIDAJNI is the glue code between C++ and Java that allows any C++ code to access a Java implementation of the AIDA interfaces. For example AIDAJNI is used with Geant4 to access JAIDA (see Figure 2).

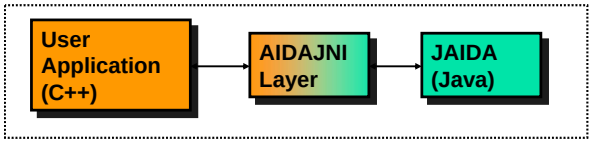

Figure 2: Usage of AIDAJNI.

AIDAJNI is part of FreeHEP Java library [4], but can be downloaded as a separate, system-specific module.

When using AIDAJNI, before compiling C++ program users need to:

- Install JAIDA and AIDAJNI
- Define JAIDA_HOME and AIDAJNI_HOME environment variables
- Run JAIDA and AIDAJNI setup scripts (included in the distribution)

Detailed step-by-step instructions how to use AIDAJNI are included in release notes [8] and also in the Exercise 5 of Geant4 Tutorial CD at SLAC [9].

The current version of AIDAJNI is 3.2.2, implementing AIDA 3.2.1.

AIDA TAG LIBRARY

Based on AIDA and JAIDA, the tag library AIDATLD (AIDA Tag Library [10]) has been developed. It provides a very easy way to include live plots into Java Server Pages (.jsp) files. AIDATLD is designed to work in a container supporting JSP 2.0 or greater, such as Tomcat 5.0.*. It consists of a set of AIDA-based, HTML-like tags: <aida:tree>, <aida:plotter>, <aida:style>, etc. Such tag structure makes it easy for users that are familiar with AIDA interfaces to read, understand, and create jsp pages.

AIDATLD can be especially useful when it is needed to quickly make data and plots accessible to collaborators.

Figure 3 shows example of AIDATLD page, as rendered by Internet Explorer.

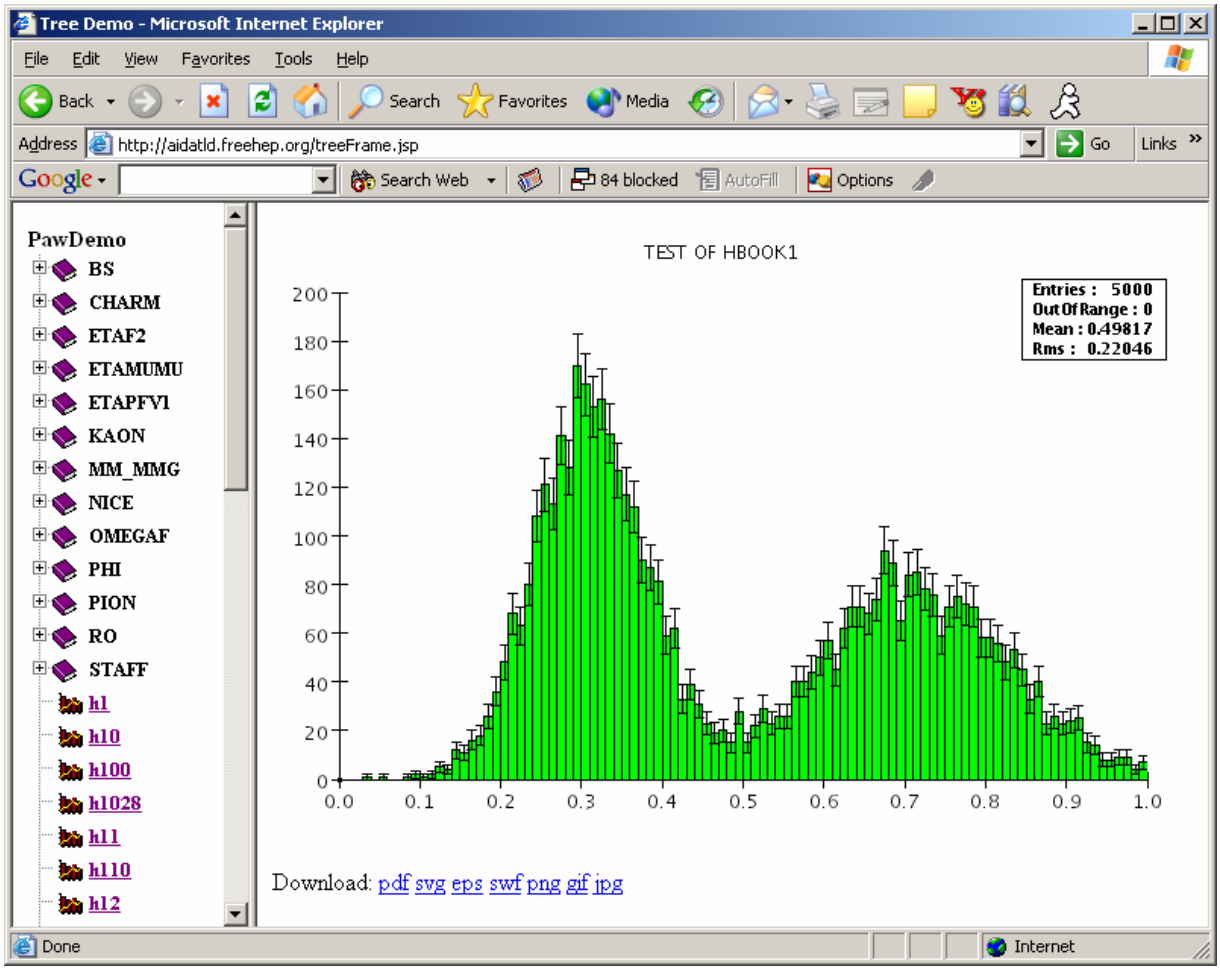

Figure 3: Example of AIDATLD page.

Here is example of jsp page code that illustrates the use of AIDATLD tags:

```
<%@taglib prefix="aida" uri="http://java.freehep.org/jsp/aida" %>
<%@taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>

 <html>
   <head><title>Single plot of a histogram accessed via rootd</title></head>
       <body>
          <c:set var="rootDataURI" value="root://rh92.slac.stanford.edu/demo.root" />
          <c:set var="histoPath" value="/h110" />
          <aida:plotter>
                <aida:region>
                   <aida:plot dataSourceURI="${rootDataURI}" plotObjectPath="${histoPath}" />
                </aida:region>
          </aida:plotter>
       </body>
</html>
```

AIDATLD documentation, examples, and demonstrations are available at [10].

CONCLUSION

AIDA provides rich, easy to learn and to use set of interfaces for data analysis. Ongoing efforts to improve and extend AIDA interfaces are based on user input. There is a bug tracking system in place [10], please post any bug reports, problems, requests for new features there. Also you are welcome to take part in the discussion forum [11].

Several implementations of AIDA are currently available in C++, Java, and Python. They all share the same AIDA XML format for representation and storage of analysis objects. So that Tuples, Histograms, etc. saved in this (possibly compressed) XML format can be shared between different AIDA implementations.

There is a bridge between the C++ and Java worlds, AIDAJNI, which is a glue layer that allows using Java implementation of AIDA from C++ program.

Java implementation of AIDA, JAIDA, and interactive analysis tool, JAS3, provide convenient way to do interactive and batch analysis using AIDA interfaces.

AIDA Tag Library and JAIDA provide an easy way to include live plots into HTML pages.

All products above use AIDA interfaces, which does not require user to learn any new analysis systems.

REFERENCES

[1] http://aida.freehep.org.
[2] http://cern.ch/pi.
[3] http://www.lal.in2p3.fr/OpenScientist.
[4] http://java.freehep.org.
[5] http://paida.sourceforge.net.
[6] http://java.freehep.org/aid.
[7] http://jas.freehep.org/jas3.
[8] http://java.freehep.org/aidajni/ReleaseNotes-3.2.2.html.
[9] http://geant4.slac.stanford.edu/g4cd.
[10] http://aidatld.freehep.org.
[11] http://bugs.freehep.org.
[12] http://forum.freehep.org.