The Whizard-JNI Package: A Java Interface to Whizard

M. Ronan*
Lawrence Berkeley National Laboratory
University of California, Berkeley CA USA
(Dated: November 27, 2001)

The Whizard-JNI package is a stand-alone software system for setting up and running the Whizard Monte Carlo program from within Java. Full Whizard functionality is provided through a simple "straght-forward" interface to native FORTRAN 95 code. The package includes examples for running Whizard within a Java framework for Linear Collider detector (LCD) simulations. These LCD examples use Whizard for event generation, execute the U.S. Fast Monte Carlo (FMC) detector simulation and perform user analysis in Java and/or FORTRAN 95.

I. INTRODUCTION

Whizard is a generic Monte-Carlo generator for multi-particle processes at high-energy colliders [1], implemented in FORTRAN 95. The Java Language Environment [2] offers the possibility of developing physics analysis for the next generation of HEP facilities in a clean object-oriented methodology. A reliable Java interface to Whizard would allow new physics and detector studies to take full advantage of Whizard.

The Whizard-JNI package [3] has been developed to provide a convenient general purpose Java interface to the Whizard Monte Carlo system, following similar developments of a Pythia-JNI package [4]. Basic physics generation and fragmentation processing is provided through a Java Native Interface (JNI) to underlying Whizard routines, such as integrate and generateEvent. Whizard can simply be run as a stand-alone application through its Java interface. In a Java framework package, the generated events are accessed through a JNI interface to the HEPEvt common block and passed to Java analysis modules. Provision has been made to run existing User FORTRAN 95 code through a UserF90Analysis class and corresponding UserF90-JNI interface.

A Java software framework has been developed for Linear Collider detector (LCD) [5] simulations. In that framework, the Java Analysis Studio (JAS) [6] provides a powerful environment for reading in existing Monte Carlo event files and running LCD reconstruction and analysis applications. For unique or large statistics physics and fast detector studies, specialized stand-alone applications can generate and analysis events saving intermediate histograms and results for subsequent analysis. JAS can then be used to view histograms, analyze intermediate results and prepare final presentations. A number of simple stand-alone Java analysis examples for Linear Collider physics analysis and Fast Monte Carlo (FMC) detector simulations are discussed below.

II. INTERFACING TO WHIZARD

The Java Development Kit (JDK) [2] provides easy to use tools for constructing interfaces to native code such as Whizard and the HEPEvt FORTRAN common block. The developer describes the Java signature of underlying "native" methods with the required arguments and return variables. The JDK writes a header file specifying how the interface should be implemented. A simple ANSI-C implementation is used to invoke Whizard routines or to access HEPEvt event variables.

The **hep.generator.whizard** Java package contains a "straight-forward" interface to Whizard through simple interfacing methods. The **Whizard** Java class describes the signature of Whizard routines. A simple implementation **WhizardImp.c** provides the interface from the Java language methods to the underlying FORTRAN 95 routines, such as integrate and generateEvent. A standard GNU Makefile is used to compile the Java classes, make the native interface definitions and compile the C implementations. Links to Whizard, O'Mega and CERN object libraries are made in a shared object library that is loaded into the Java Virtual Machine (VM). Whizard can then be executed from its Java main method or included in applications.

```
Useage:
        To create the Whizard-JNI interface
                Whizard whizard = new Whizard();
```

*ronan@lbl.gov

```
To read in Whizard data files and integrate
         whizard.integrate();

To generate an event
         whizard.generateEvent();
```

A **HEPEvt** class, derived from the JAS [6] package, provides a Java interface to the HEPEvt FORTRAN common block. An ANSI-C implementation **HEPEvtImp.c** is used to interface the Java methods to the HEPEvt data structure viewed from C. Java native methods such as getNEVHEP() and getNHEP() return the generated event number and number of particles, respectively. While methods such as getPHEP(i,j) return double precision values for the four momenta of each particle. The HEPEvt-JNI is loaded in the Whizard shared object library to allow access to the generated event quantities.

III. JAVA FRAMEWORK

A simple Java framework for data processing has been developed as part of the JAS [6] environment. This base framework has been extended for Linear Collider detector (LCD) [5] simulations. In stand-alone Whizard-JNI packages [3], the main program uses the base framework in generating Whizard events. The LCD simulation and analysis framework is then used for Fast Monte Carlo detector simulation and user analysis.

A. Event generation

The **hep.analysis** package provides a basic Java simulation and analysis framework. A **Job** class has methods for adding **EventSource**'s and **EventAnalyzer**'s to the processing stream. When the processors have been loaded the **Job** executes a specified number of events, writing out **StdHEP** events if requested, and then saves histograms and any other output at termination.

In simulations, a main program method creates an analysis job and then adds Whizard as the event source. Various LCD simulation and analysis modules are then added as described below.

B. LCD Software (The hep.lcd Class Library)

The **hep.lcd** class library [5] provides an extensible Java framework for running sophisticated analyses such as fast parameterized Monte Carlo (FastMC) simulation and full event reconstruction. The LCD simulation system allows analysis of generator-level four vectors, FastMC quantities or space point data from the full tracking package. Use of the Java Language Environment allows rapid development of analysis modules and reconstruction algorithms with excellent through-put in processing. A suite of physics analysis tools, such as histograming, event display and jet finding, enable users to build on proven code. The overall design emphasizes flexibility and extensibility, typically providing multiple algorithms in following object-oriented design rules. The framework can be used as a standalone or run inside Java Analysis Studio. Within the modular LCD design, **Driver** objects receive the **LCDEvent** data from the Monte Carlo generator and execute a list of **Processor**'s.

The **WhizardGenerator** class in the **hep.lcd.generator.whizard** package provides a wrapper to the Whizard Java interface described in Sec. II. It executes Whizard methods to generate each event and then reads out the **HEPEvt** common block in forming a valid **LCDEvent** object.

The **hep.lcd.mc.MCFast** processor performs a simple parameterized Fast Monte detector simulation with straight forward acceptance cuts on final state particles. Charged particle momentum resolution tables calculated for tracks at various momenta and angles are used for smearing the generated particle momenta. Simple parameterized energy resolutions are used for photons and neutral hadrons.

Java analysis modules can simply be added to the LCD framework as described below. A UserF90-JNI is available to fill ntuples for use in FORTRAN 95 analyses or to be written out for subsequent analysis.

IV. JAVA ANALYSIS EXAMPLES

In the event generation and analysis examples, a main **GenWhizard** class configures Whizard for $e^+e^- \rightarrow W^+W^-/\ ZZ$ event generation at 1 TeV. It creates a **hep.analysis.Job** adding Whizard as the event source.

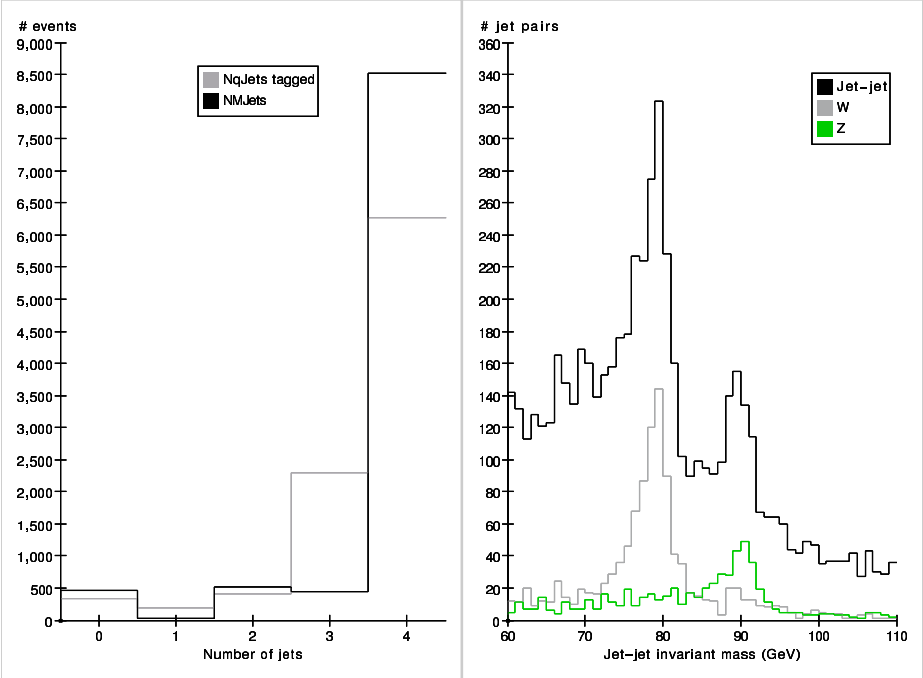

FIG. 1: Comparison of the total number of reconstructed jets and the number of tagged jets (a), and jet-jet invariant mass distributions for pairs of large-angle jets in the W / Z mass region (b) in 1 TeV $e^+e^- \rightarrow W^+W^-/\ ZZ$ events.

A simple analysis **Driver** executes a **ShapeAnalysis** process to determine the event Thrust axis and then executes the Fast Monte Carlo simulation. The analysis **Processor**'s described below are then executed.

The **WhizardEventAnalysis** class gets the tracks and clusters from the simulated event, writes out the quanties for the first few events and makes some simple histograms. The code and output histograms are available from the web release directory [3].

The **WhizardJetAnalysis** class uses tracks and clusters to find jets in simulated $e^+e^- \rightarrow W^+W^-/\ ZZ$ events. Reconstructed jets are compared to partons and leptons from the Whizard event generation to identify hadronic jets. Figure 1a shows a comparison of the total number of hadronic jets found and the number that are tagged. The tagged jets provide a good reconstruction of the original parton direction and energy. The difference being due to jets that are merged or mismeasured. Jet-jet invariant mass distributions for pairs of large-angle jets in the W and Z mass region are shown in Fig. 1b. Also show are the mass distributions for jet pairs recoiling against a reconstructed W or Z. The plot is indicative of the intrinsic W / Z separation that is possible in the reconstruction of color neutral particles in e^+e^- collisions.

Acknowledgments

I'd like to thank Wolfgang Kilian and Thorsten Ohl for expressing interest in the Whizard-JNI development, and Tim Barklow for setting up the FORTRAN 95 compiler and Whizard native libraries.

---

[1] Whizard V1.13, "A generic Monte-Carlo generator for multi-particle processes at high-energy colliders", W. Kilian, http://www-ttp.physik.uni-karlsruhe.de/Progdata/whizard.
[2] JDK1.1.8 and higher, Sun Microsystems, Inc., http://java.sun.com.
[3] M. Ronan, "Whizard Java Native Interface (JNI) Software", http://www.lbl.gov/∼ronan/docs/Whizard-JNI.
[4] M. Ronan, "The Pythia-JNI Package: A Java Interface to Pythia" in these proceedings, and http://www.lbl.gov/∼ronan/docs/Pythia-JNI.
[5] M. Ronan et al, "Java Analysis Studio and the hep.lcd Class Library", International Workshop on Linear Colliders, Sitges, Barcelona, Spain, 1999; G. Bower et al., "Java-based LCD Reconstruction and Analysis Tools", International Linear Collider Workshop, Fermilab, 2000; and N. Graf et al., "LCD Software Status", in these proceedings.
[6] A.S. Johnson, *Java Analysis Studio*, http://jas.freehep.org/.