JLC Study Framework

Akiya Miyamoto, KEK
13-June-2001 @ROOT2001

Contents:

1. Introduction
2. Concepts of JSF
3. Packages in JSF
4. Run example
5. Problems of ROOT

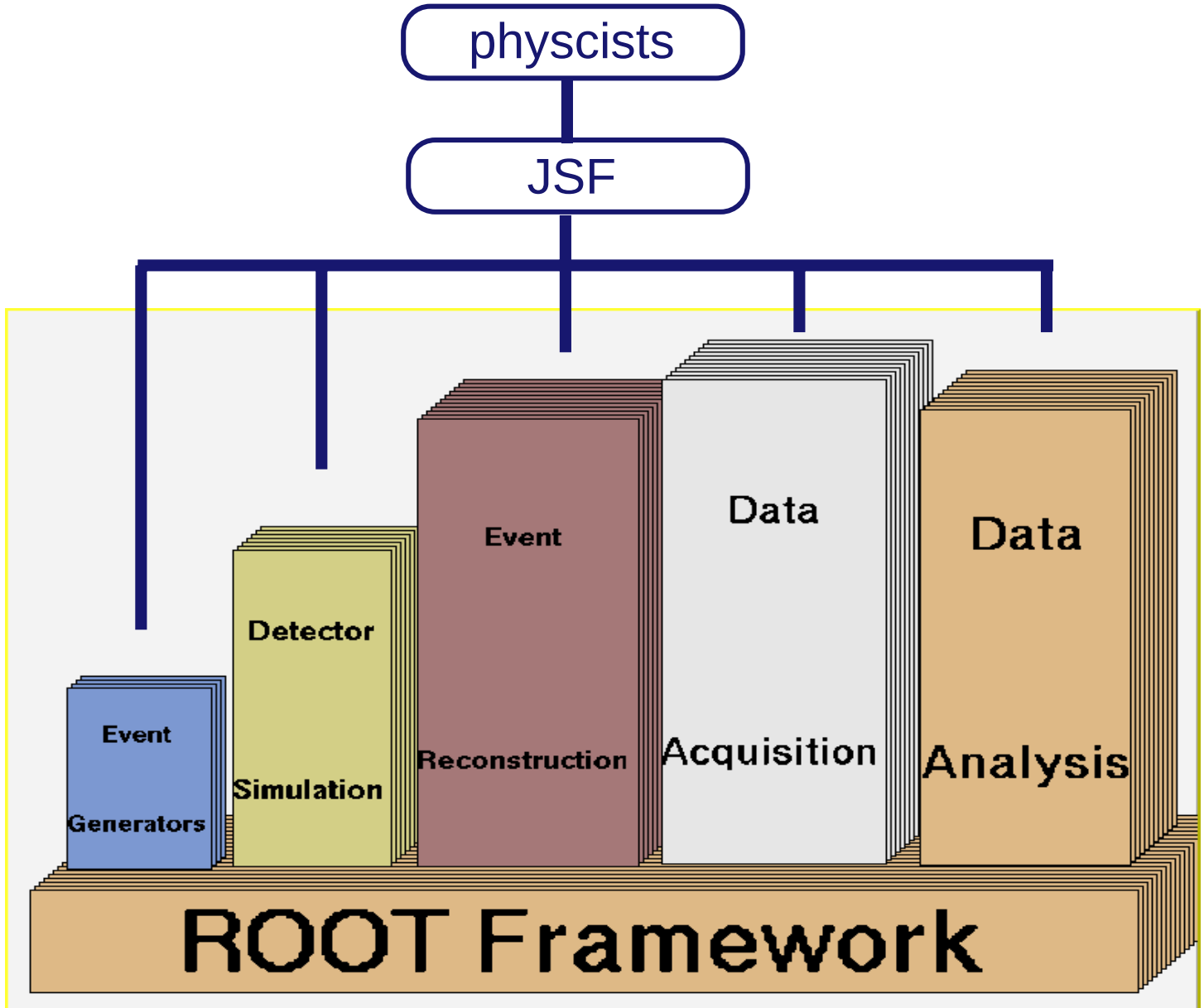
JSF home page: http://www-jlc/subg/offl/jsf

ACFA Joint Linear Collider
Physics and Detector Working Group

http://acfahep.kek.jp/

ROOT and JSF

JSF is a ROOT bases program to provide a common interface to physcists



JSF features

1. Common framework for Monte Calro simulation,
   its analysis, and analyses of beam test data, so on.

   User needs to learn only one language, C++

2. Frame work for modular analysis

3. GUI for run control

   Interactive and Batch in same framework

4. Event display and Histograming included

5. Object I/O

6. Still using many old fortran routines
   for generators and simulators

Concept of JSF run control

General feature of HEP data analysis:

1. Event-by-event analysis
2. Event data consists of several sub-components, analizer of them needs initialization and termination when job or run begins

Standard flow of JSF job

Create *modules*
Job Initialize
Begin run
Event Analysis
End Run
Job Termination

- Execution flow are controled by a class JSFSteer.
- One Modules are created, calls of their function are controled by JSFSteer
- Thus, inclusion/exclusion of analysis module is easy.
- The flow is defined by Macro(cint). So it can be easily modified.

All analysis classes must be inherited from JSFModule and JSFEventBuf

JSFModule : provide functions such as

Initialize(), BeginRun(), Process(), EndRun(), Terminate()

JSFEventBuf : A class to save event data in a ROOT file as a tree

Macro for job control

For standard analyses, a macro gui.C is prepared. gui.C can

1. GUI based job control, such as

   Read data or event simulation, type of generator, etc.

2. Show and set job parameters

3. Invoke event display

4. Show histograms

5. Call user macro function event by event bases for simple data analysis

6. Include non-standard module in the analysis sequence.

7. Same macro for batch and interactive analysis

8. Add user own menu by macro.

Type of data analysis in JSF

1. By a class inherited from JSFModule

   Write C++ codes, Makefile, etc.
   Execution is fast, good for well established analysis.
   A lot of code writting is required to get started.

   A script to help writting a program

   Show files

   Thus created module can be included in gui.C or user can prepare his/her own macro and define analysis sequence

2. Simple analysis using MACRO

   Global functions such as UserInitialize(), UserAnalysis() are called when gui.C is used. These functions are useful for handy data analysis.

Parameter file

All parameters are managed by JSFEnv class

In the user program, they are obtained by a class

JSFEnv::GetEnv(*"Parameter.Name"*, *default*)

At run time, parameter can be changed by three method

1. In a file, jsf.conf

   Parameter.Name : value
   #!*argname*
   # Comments
   .....

   argname is an alias of *Parameter.Name* used to parse command line argument

2. As a command line argument, like

   [%] jsf *--argname=value* gui.C

3. By popup menus of JSF Control panel

   PythiaGenerator: Type of process, CM energy, etc
   DebugGenerator: Particle ID, momentum, etc...

Each user can add their own menu by a function, *UserMenu()*

File I/O of JSF

Output file:

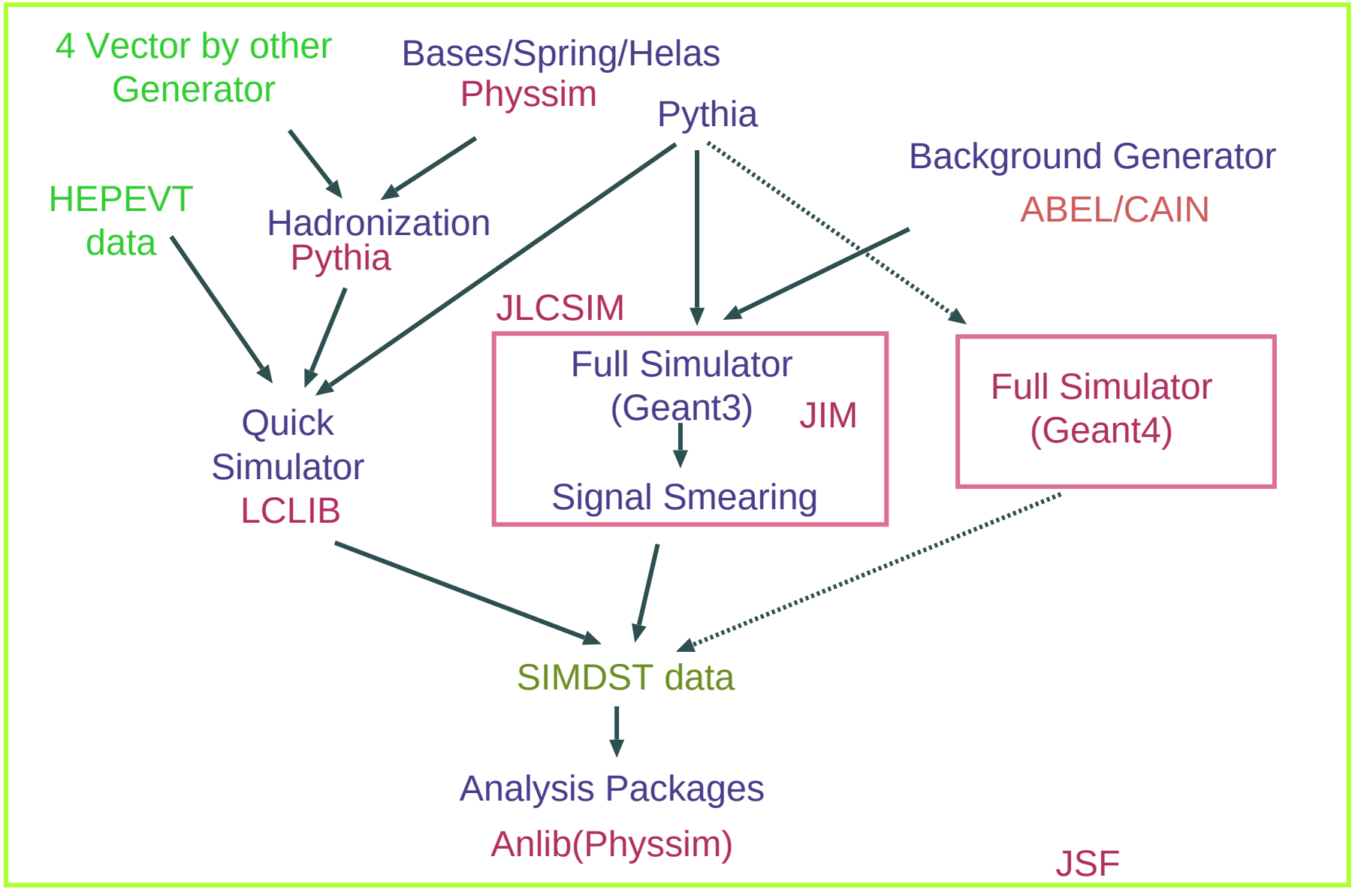
1. event tree: Event data of each module is stored as a branch.
2. conf : List of module names used for analysis Module parameters and run summary, etc
3. Histograms/ntuple

Input file:

Root file(Created by JSF), ZEBRA (JIM), ASCII(Generator), HEPEVT format files.

When reading the ROOT file, information in conf directory is used to define objects where data are stored

Packages included or related to JSF



Physsim Packages

(http://www-jlc.kek.jp/subg/offl/physsim)

Generator and Analysis packages

Generator:

Based on HELAS+BASES+SPRING

Processes:

$$
\begin{array}{ll}
\text{Higgs} & : e^+e^- \to Zh \\
\text{SUSY} & : e^+e^- \to \tilde{f}\bar{\tilde{f}}, \tilde{\chi}\bar{\tilde{\chi}} \\
\text{Top} & : e^+e^- \to e^+e^- t\bar{t}, \nu\bar{\nu}t\bar{t}, t\bar{t}h, t\bar{t}, t\bar{t}Z \\
\text{Two Photon} & : e^+e^- \to e^+e^- f\bar{f}(f = e, \mu, \tau, q) \\
\text{WZ} & : e^+e^- \to e^+e^- W^+W^-, e^+e^- Z, e\nu W, \\
 & \quad \nu\bar{\nu}W^+W^-, \nu\bar{\nu}Z, W^+W^-, W^+W^-Z, ZZ
\end{array}
$$

Included effects

Bremsstrahlung and beamstrahlung
W, tau polraization
Virtual Higgs diagrams

Analysis:

Packages such as jet finder, event shape routines,

Example of JSF run

Process: $e^+ e^-$ -> WWZ -> 6 partons

Generator: coded by HELAS, phase space integration and 4 vector generation by BASES/SPRING package.

Simulator: QuickSimulator

Analysis: Histogram number of charged tracks, etc.

Procedure:

1. Integration :

2. Event generation/Simulation/Analysis

Start jsf

Problems and Limitations of ROOT

Our platforms:

Linux, MacX, HP-UX10.20, IBM AIX

AIX system has a large disk space and HPSS.

Problems:

ROOT(3.00.06) does not run well on AIX with xlC5.

Langauge specification of xlC5 is different with the previous version of xlC, such as a treatment of a forward declaration of class and function.

The probem was solved by inserting declaration

Now several programs in tutorial and test directory works, but others (h2root, etc) are not. It seems that a class table is not created properly when shared libraries are loaded. We are discussing with local IBM SE.