ILC Software

Reconstruction

Hartwig Albrecht
DESY

Vancouver Linear Collider Workshop

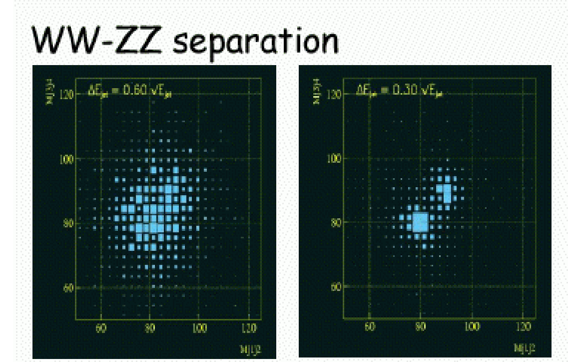
Vancouver, July 19 - 22, 2006

Outline

- Introduction
- Reconstruction Software Tools
  - **LCIO** – data model & persistency
  - Frameworks**: Marlin** vs **org.lcsim**
  - Reconstruction **org.lcsim Reconstruction** vs **MarlinReco**
  - Conditions data: **LCCD** vs **org.lcsim Conditions**
  - Geometry description: **GEAR** vs **Compact Geometry description**
  - Event viewer: **Wired4** (org.lcsim) vs **CED viewer** (Marlin)
  - Interoperability
- Summary & conclusion

Demands on Reconstruction

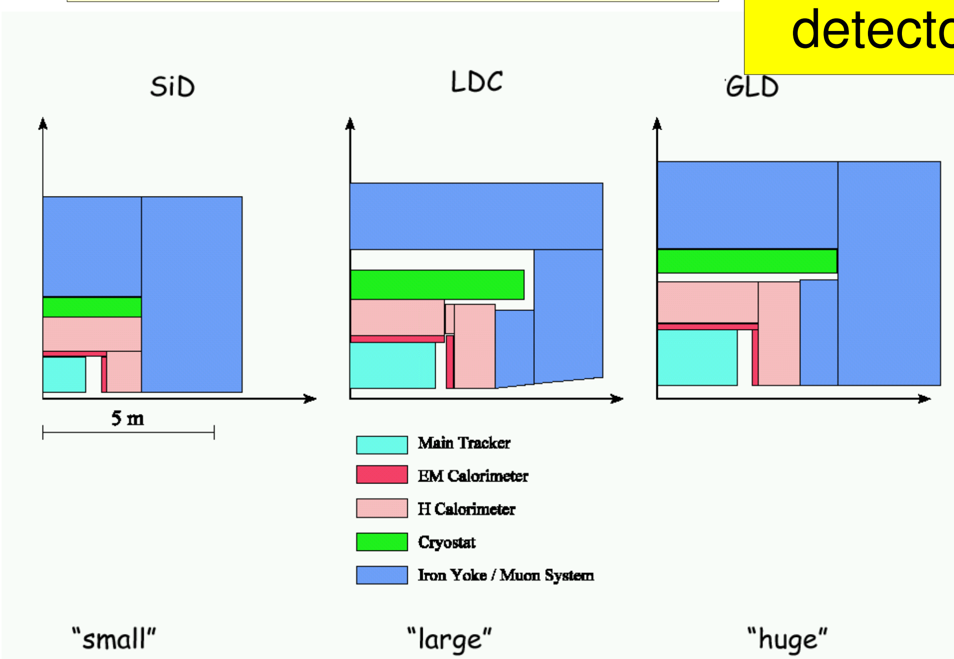
- precision tracking and vertexing
- take advantage of the high granularity in calorimeters
- very high jet-mass resolution needed ~30%/sqrt(E/GeV):



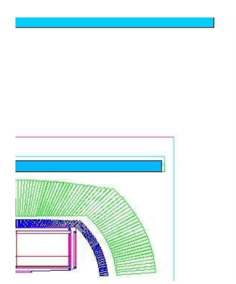
Particle Flow

- reconstruct all single particles
- use tracker for charged particles
- use Ecal for photons
- use Hcal for neutral hadrons

- reconstruction algorithms (pflow) determine the overall detector performance
- need full reconstruction to choose and optimize detector concepts



4TH

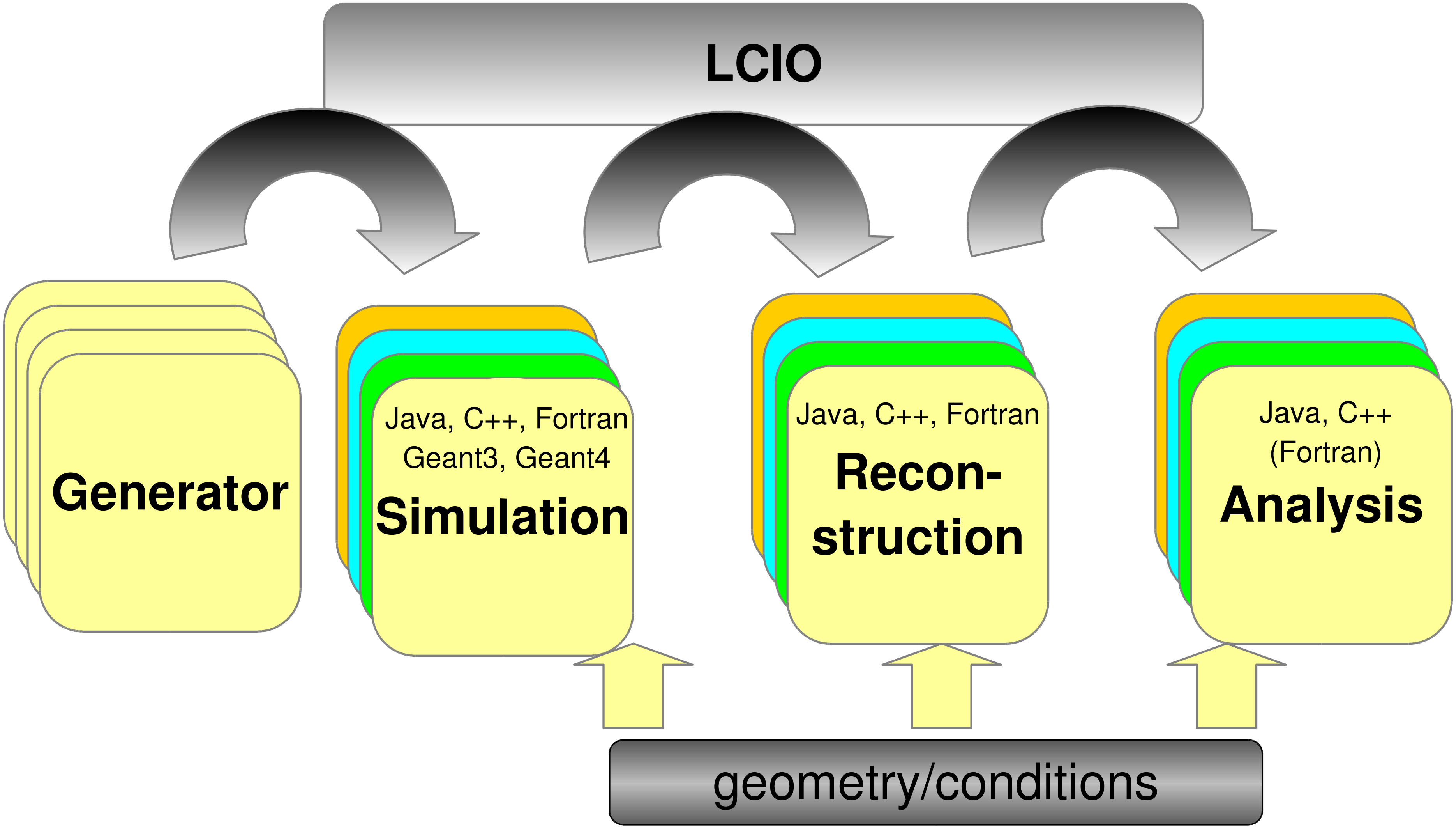


four interregional detector concept studies ongoing

ILC software packages

	Description	Detector	Language	IO-Format	Region
Simdet	fast Monte Carlo	TeslaTDR	Fortran	StdHep/LCIO	EU
SGV	fast Monte Carlo	simple Geometry, flexible	Fortran	None (LCIO)	EU
Lelaps	fast Monte Carlo	SiD, flexible	C++	SIO, LCIO	US
Mokka	full simulation – Geant4	TeslaTDR, LDC, flexible	C++	ASCI, LCIO	EU
Brahms-Sim	Geant3 – full simulation	TeslaTDR	Fortran	LCIO	EU
SLIC	full simulation – Geant4	SiD, flexible	C++	LCIO	US
LCDG4	full simulation – Geant4	SiD, flexible	C++	SIO, LCIO	US
Jupiter	full simulation – Geant4	JLD (GDL)	C++	Root (LCIO)	AS
Brahms-Reco	reconstruction framework (most complete)	TeslaTDR	Fortran	LCIO	EU
Marlin	reconstruction and analysis application framework	Flexible	C++	LCIO	EU
hep.lcd	reconstruction framework	SiD (flexible)	Java	SIO	US
org.lcsim	reconstruction framework (under development)	SiD (flexible)	Java	LCIO	US
Jupiter-Satelite	reconstruction and analysis	JLD (GDL)	C++	Root	AS
LCCD	Conditions Data Toolkit	All	C++	MySQL, LCIC	EU
GEAR	Geometry description	Flexible	C++ (Java?)	XML	EU
LCIO	Persistency and datamodel	All	Java, C++, Fortran	-	AS,EU,US
JAS3/WIRED	Analysis Tool / Event Display	All	Java	xml,stdhep, heprep,LCIO,	US,EU

ILC software chain



LCIO Overview

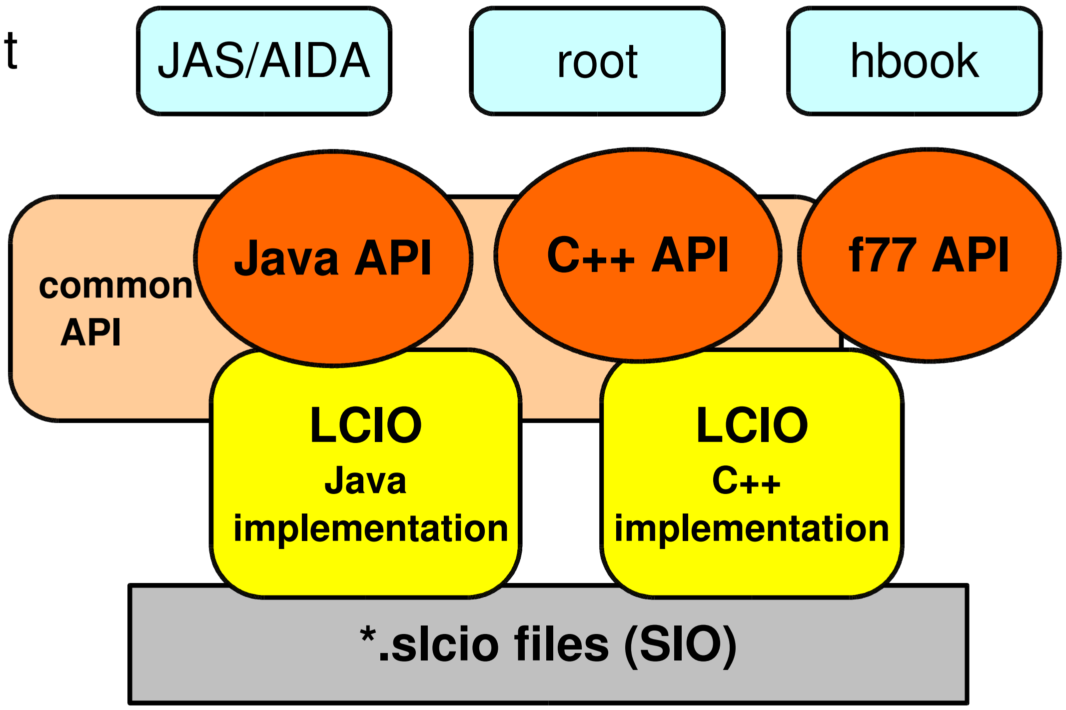
- DESY and SLAC joint project:
  - provide a common basis for ILC software
- Features:
  - Java, C++ and f77 (!) API
  - extensible data model for current and future simulation and testbeam studies
  - user code separated from concrete data format
  - no dependence on other frameworks

simple & lightweight

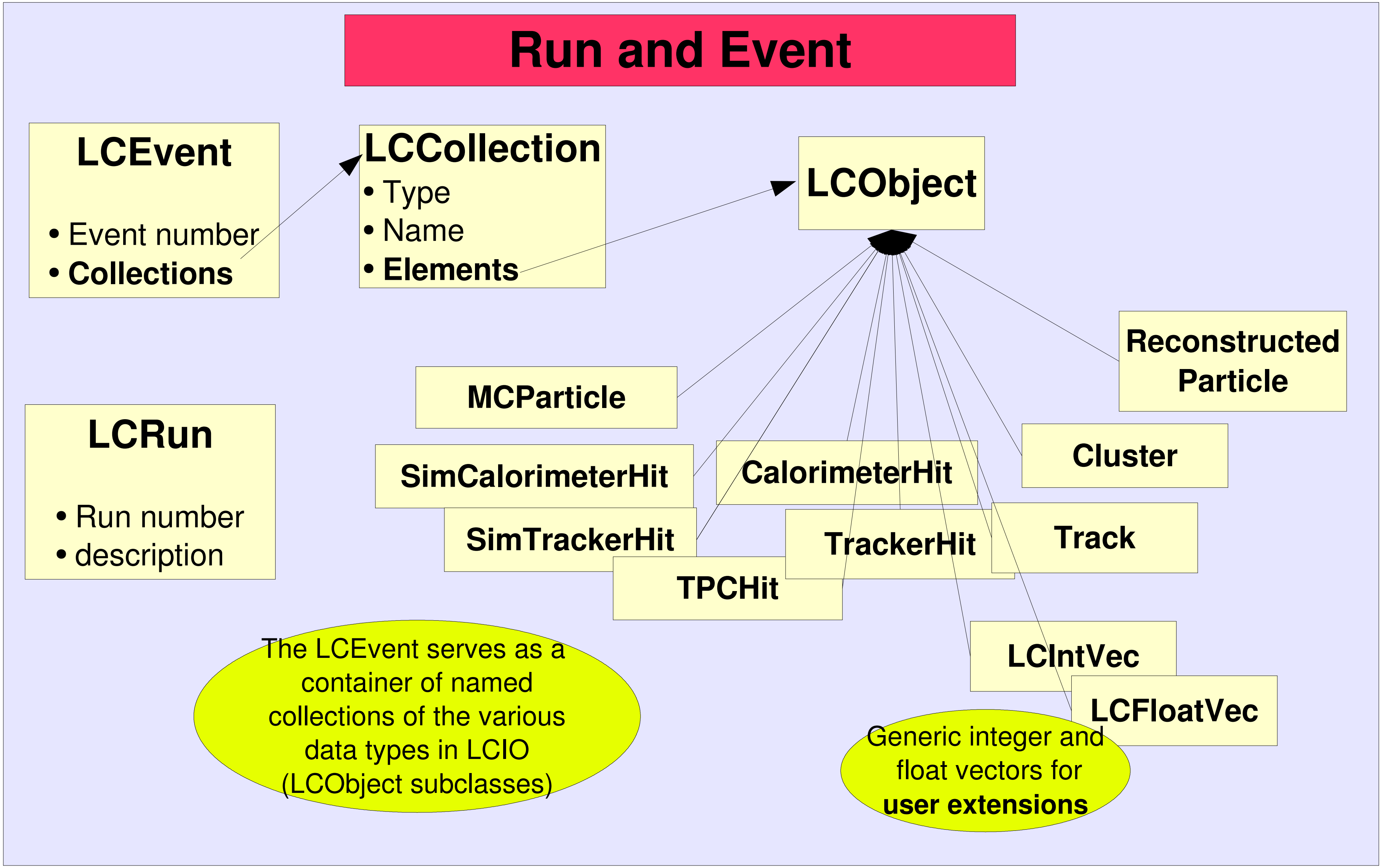
new release: **v01-07**

now de facto standard persistency & data model for ILC software

SW-Architecture



LCIO Event Data Model



LCIO Event Data Model

- the LCIO event data model is fairly complete, nevertheless flexible for future extensions
- thus, it has been adapted and can be extended as needed by the community
  - maintaining downward compatibility
  - with international discussion and agreement

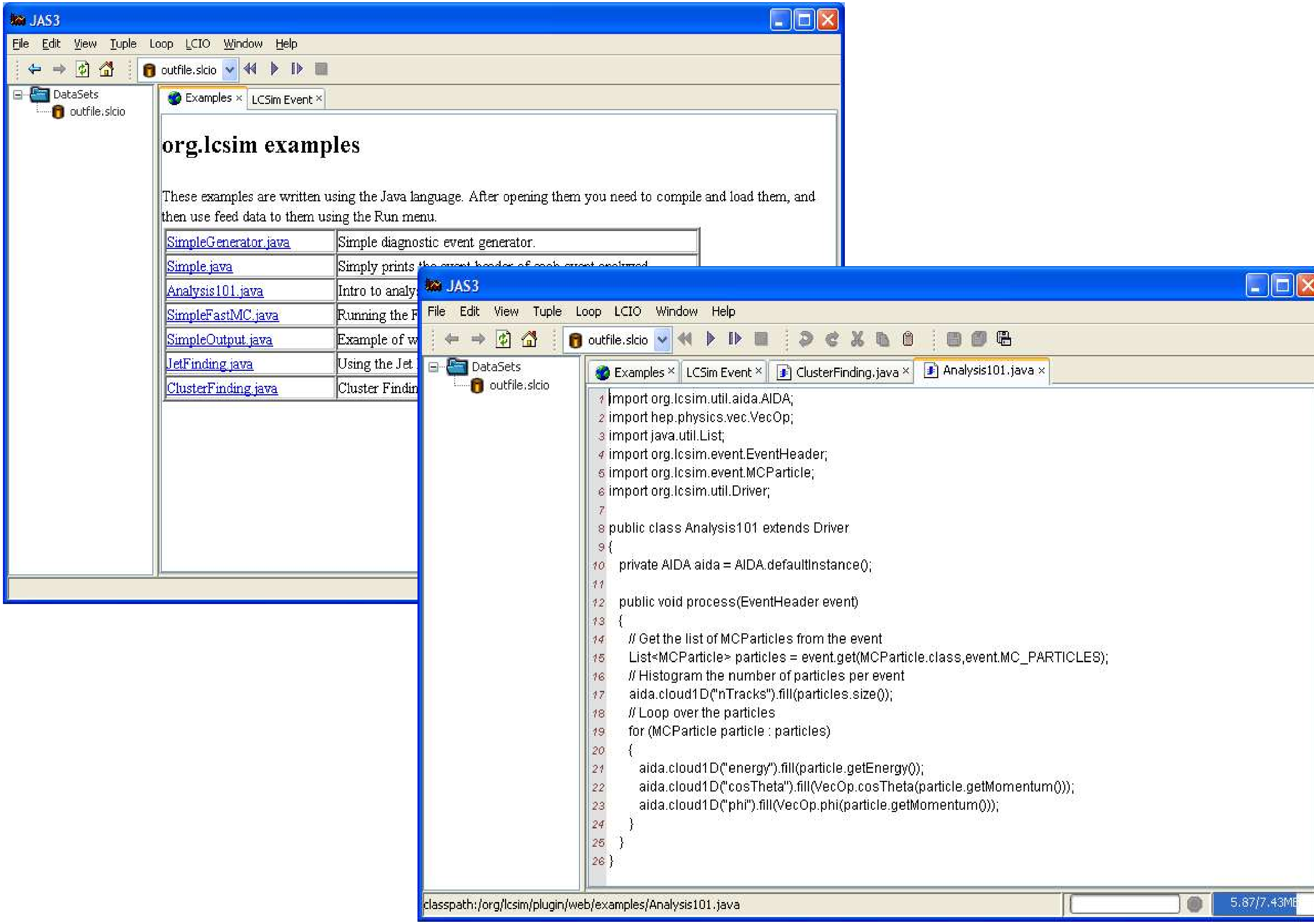
Framework I: org.lcsim

- "Second generation" ILC reconstruction/analysis framework
  - Builds on hep.lcd framework used since 1999
  - Full suite of reconstruction and analysis tools
- Uses LCIO for IO and as basis for simulation, raw data and reconstruction event formats
  - Isolate users from raw LCIO structures
  - Maintain full interoperability with other LCIO based packages
- Detector Independence
  - Make package independent of detector, geometry assumptions so can work with any detector
  - Read properties of detectors at runtime
- Written using Java (1.5)
  - High-performance but simple, easy to learn, OO language
  - Enables us last 10 years of software developments in the "real world"
- Ability to run standalone (command line or batch) or in JAS3 or IDE such as Netbeans, Eclipse

org.lcsim with JAS3

- JAS3 org.lcsim plugin adds:
  - Example Analysis Code
  - org.lcsim Event browser
- The org.lcsim can be used standalone, within IDE, or inside JAS3. Same code can be used in all modes, so it's easy to move back and forth
  - E.g. develop in IDE and run in JAS3
  - E.g. develop in JAS3 and run in batch
  - Easy viewing of analysis plots
  - WIRED event display integration

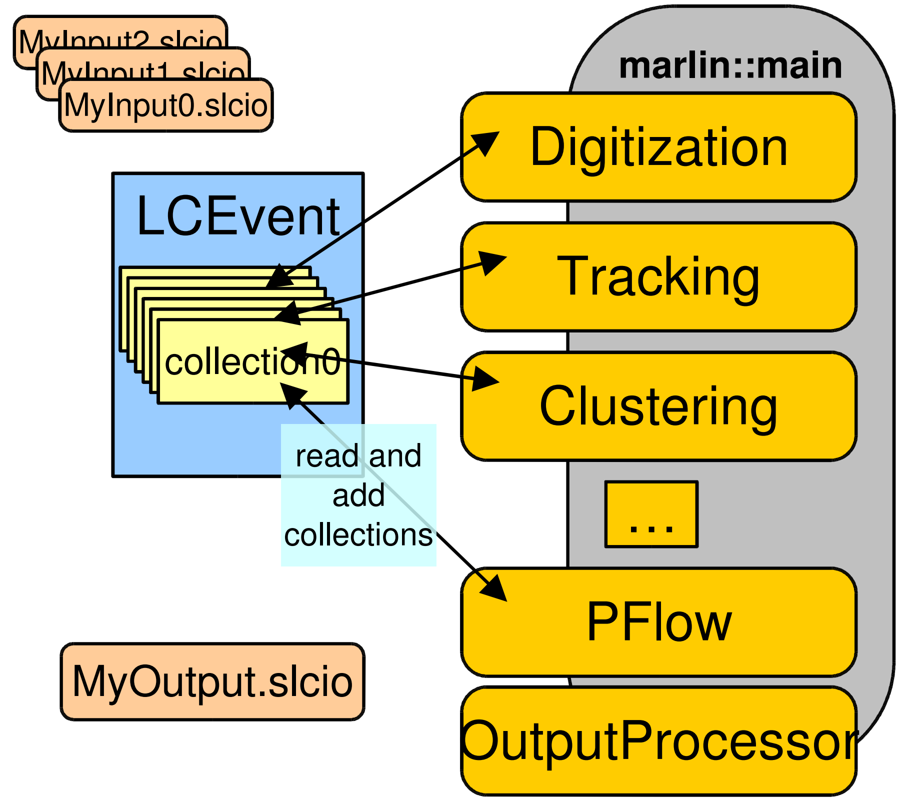
org.lcsim: Examples



Framework II: Marlin

Modular **A**nalysis & **R**econstruction for the **L I N**ear Collider

- modular C++ **application framework** for the analysis and reconstruction of LCIO data
- uses LCIO as data model
- software modules called Processors
- provides the main program
- provides simple user steering (XML):
  - program flow (active processors)
  - user defined variables
    - per processor and global
  - input/output files



Marlin features

- fully configurable through steering files:
  - program flow
  - input parameters (processor based and global)
- self-documenting:
  - ./bin/Marlin -x

    prints example steering file for all available processors
    with their parameters and example/default values
- AIDA interface for histogramming
  - easy creation of histograms through abstract interface
  - AIDAJNI/JAIDA, RAIDA (root based), ...
- configurable output
  - drop collections by name/type
- simple examples
  - user processor template, GNUmakefile,...
- easily extensible
  - makefiles 'automatically' include user packages with processors

Marlin: XML steering files

```
- <marlin>
  - <execute>
      <processor name="MyAIDAProcessor"/>
      <processor name="MyEventSelection"/>
    - <if condition="MyEventSelection">
        <group name="Tracking"/>
        <processor name="MyClustering"/>
        <processor name="MyPFlow"/>
        <processor name="MyLCIOOutputProcessor"/>
      </if>
    </execute>
  - <global>
      <parameter name="LCIOInputFiles"> simjob.slcio </parameter>
      <parameter name="MaxRecordNumber" value="5001"/>
      <parameter name="SupressCheck" value="false"/>
    </global>
  - <processor name="MyLCIOOutputProcessor" type="LCIOOutputProcessor">
      <parameter name="LCIOOutputFile" type="string">outputfile.slcio </parameter>
      <parameter name="LCIOWriteMode" type="string">WRITE_NEW</parameter>
    </processor>
  - <group name="Tracking">
      <parameter name="NTPCLayers" value="200"/>
      <processor name="MyTrackfinder" type="Trackfinder"/>
    - <processor name="MyTrackfitter" type="Trackfitter">
        <parameter name="Algorithm" value="DAF"/>
      </processor>
    </group>
    <!-- ... -->
  </marlin>
```

- Program flow defined in <execute>...</execute> section
- logical conditions from parameters evaluated at runtime

- global Parameters defined in <global/> section

- local Parameters defined in mandatory <parameter/> section

- Processors can be enclosed by <group/> tag
- Parameters in <group/> joined by all processors

a Marlin application is fully configured through the steering files (no user main program) !!

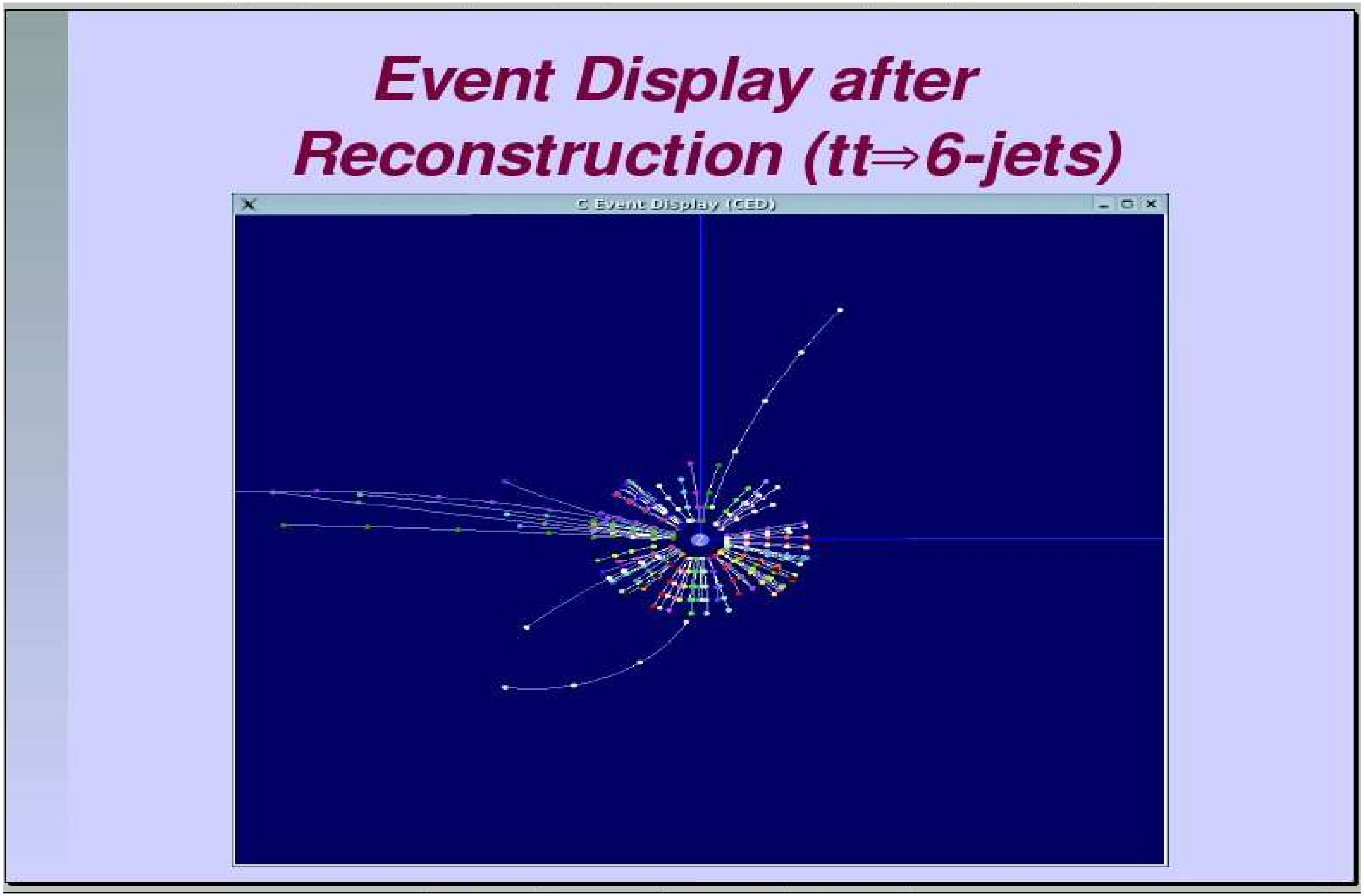
MarlinReco packages

- **TrackDigi**
  - TPCDigi
  - new VTXDigi
- **CaloDigi**
  - LDCCaloDigi
- **Tracking**
  - LEPTracking
  - new VTXTracking
  - TrackCheater
- **Clustering**
  - TrackwiseClustering
  - ClusterCheater
- **Particle flow**
  - ... next talk
- **Analysis**
  - EventShapes
  - SatoruJetFinder

most MarlinReco processors (algorithms) are **geometry independent**
–> they can be applied to **all detector concepts** (via Gear file)

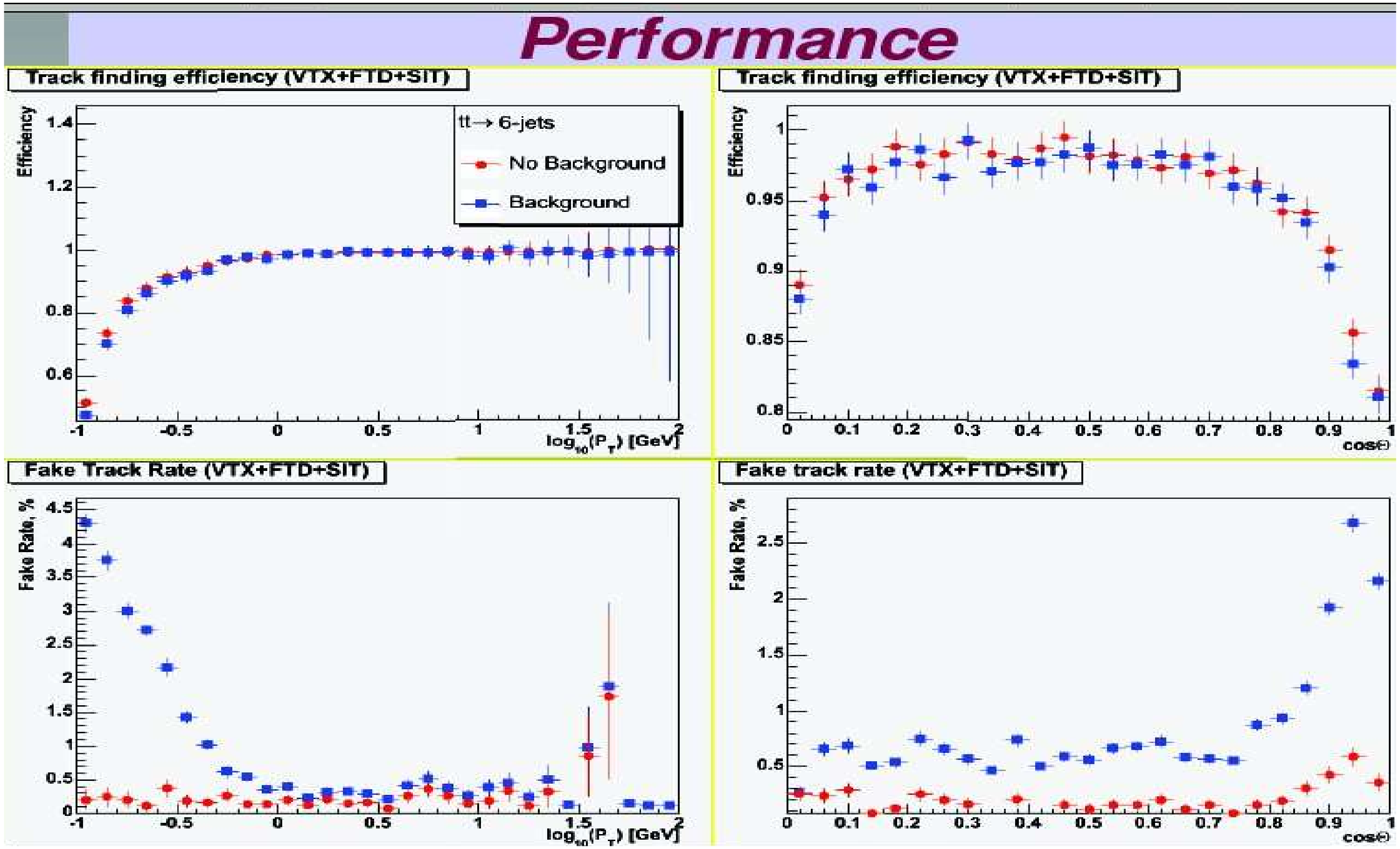
MarlinReco: Si Tracking (LDC)

Track finder & fitter with Kalman filtering:



Si Tracking for the LDC

- Track finder & fitter with Kalman filtering:



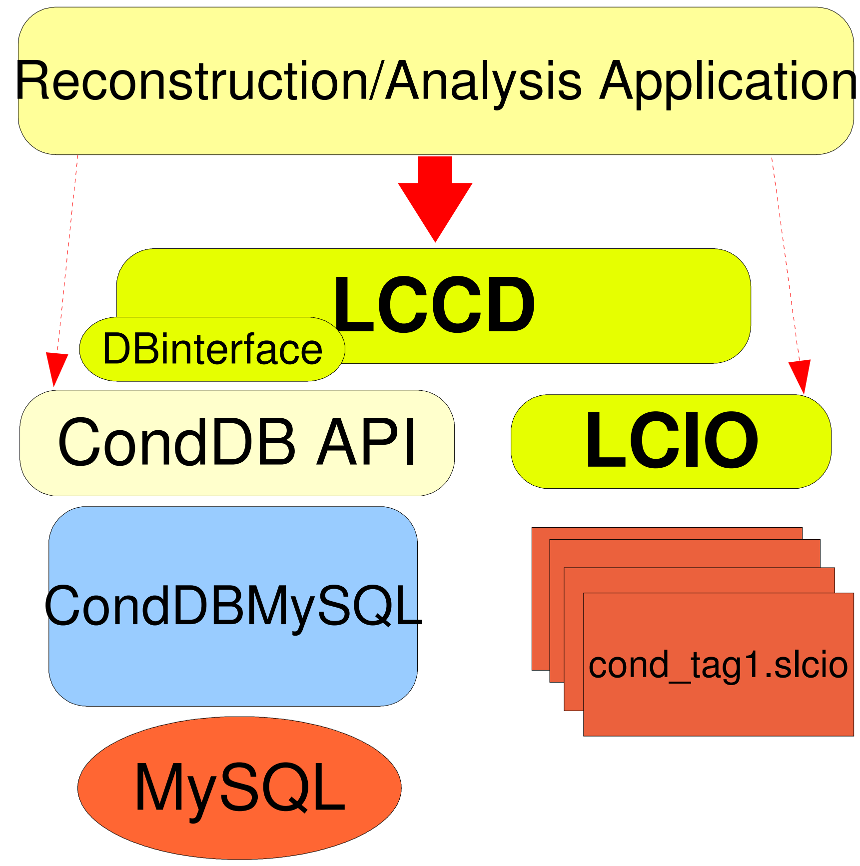
org.lcsim Reconstruction

- Reconstruction package includes:
  - Physics utilities:
    - Jet finders, event shape routines
    - Diagnostic event generator, stdhep reader/translator
    - Histogramming/Fitting/Plotting (AIDA based)
    - Event Display
    - Processor/Driver infrastructure
  - Fast MC
    - Track/Cluster smearing
  - Reconstruction
    - Cheaters (perfect reconstruction)
    - Detector Response
      - CCDSim, Digisim
    - Clustering Algorithms
      - Cheater, DirectedTree, NearestNeighbour, Cone
    - Tracking Finding/Fitting Algorithms
      - TRF,
    - Muon Finding, Swimming
    - Vertex Finding (ZvTop)

MarlinReco: LCCD

Linear **C**ollider **C**onditions **D**ata Toolkit

- Reading conditions data
  - from conditions database
  - from simple LCIO file
  - from LCIO data stream
  - from dedicated LCIO-DB file
- Writing conditions data
- tag conditions data
- Browse the conditions database
  - through creation of LCIO files
    - vertically (all versions for a timestamp)
    - horizontally (all versions for a tag)



LCCD is used by Calice for the conditions data of the ongoing testbeam studies

org.lcsim Conditions Data

- Provide access to a extensible set of conditions for each detector including:
  - Detector Geometry
  - Algorithm Specific Constants
    - E.g. FastMC smearing parameters
- Doesn't make assumptions about format of data
- Doesn't rely on internet access, or local database installation
- Detector Constants stored in .zip file
  - Typically contains:
    - Compact geometry file
    - Set of (ascii) constants for standard algorithms
  - Can additionally contain:
    - Arbitrary files (xml, ascii, binary) needed by other algorithms
    - Other geometry formats (HepRep, LCDD)
    - Full fieldmap
- To define a new detector just create a new .zip file.

Detector Geometry

Different sets of detector geometry descriptions are needed for:

- cad – not discussed here
- detector simulation (Geant)
- reconstruction / analysis

For each application, more, less, or different details are needed

Aim: One common source from which the descriptions for Geant, reconstruction, and analysis can be derived.

Two different solutions:

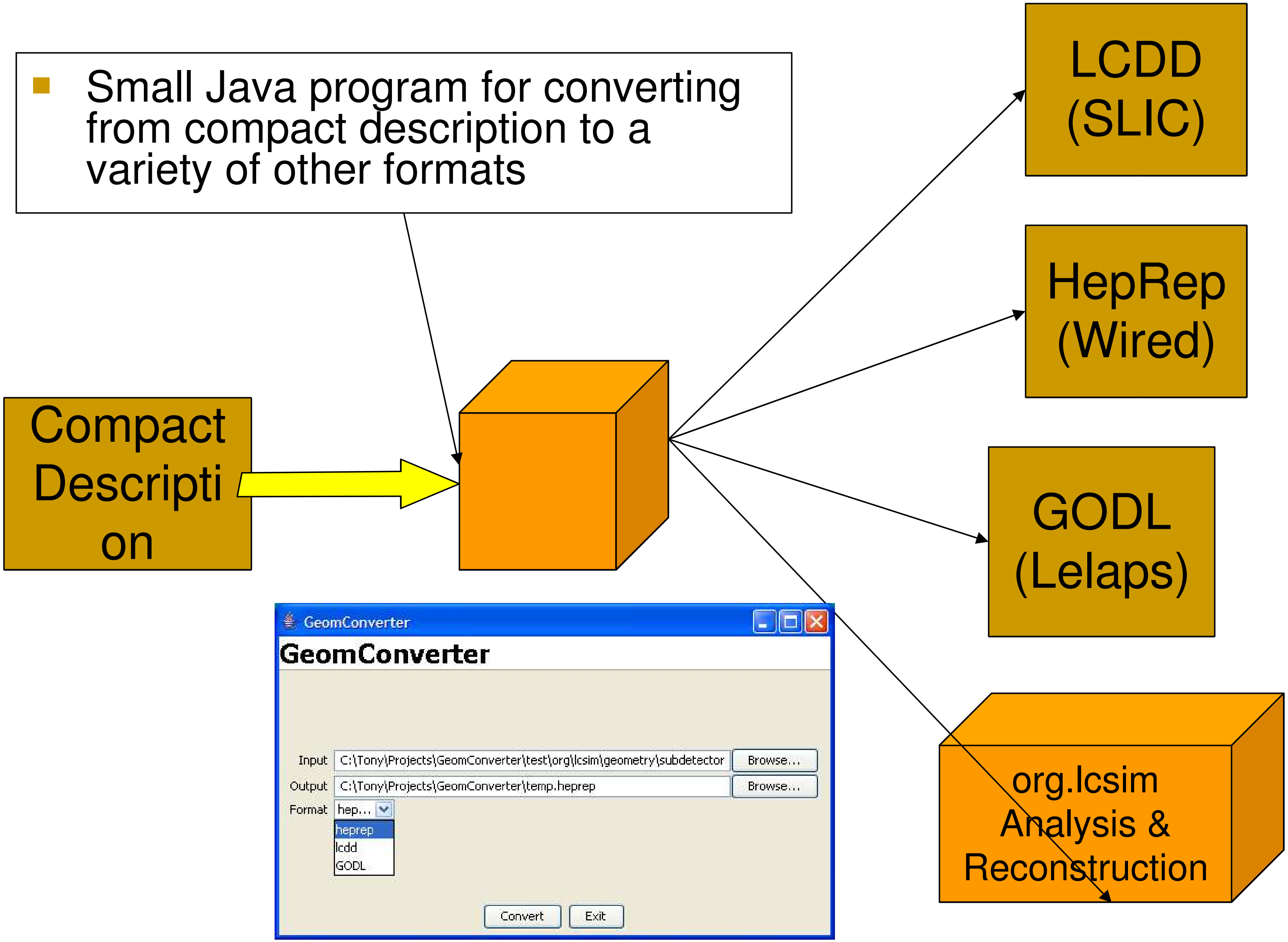
- org.lcsim: Compact geometry description
- MarlinReco: Gear

org.lcsim: Compact Geometry Description

- org.lcsim uses the "Compact Geometry Description" to define detectors
  - Simple XML format for describing ILC detectors
  - Handles typical ILC detector geometries
    - Range of detectors handled is extensible (by writing Java modules)
- Allows rapid prototyping of new detector geometries
- Does not require network access or installation of database software to run
- Automatic generation of full Geant4 LCDD geometry for full compatibility with SLIC

org.lcsim: Geometry Converter

- Small Java program for converting from compact description to a variety of other formats



Gear

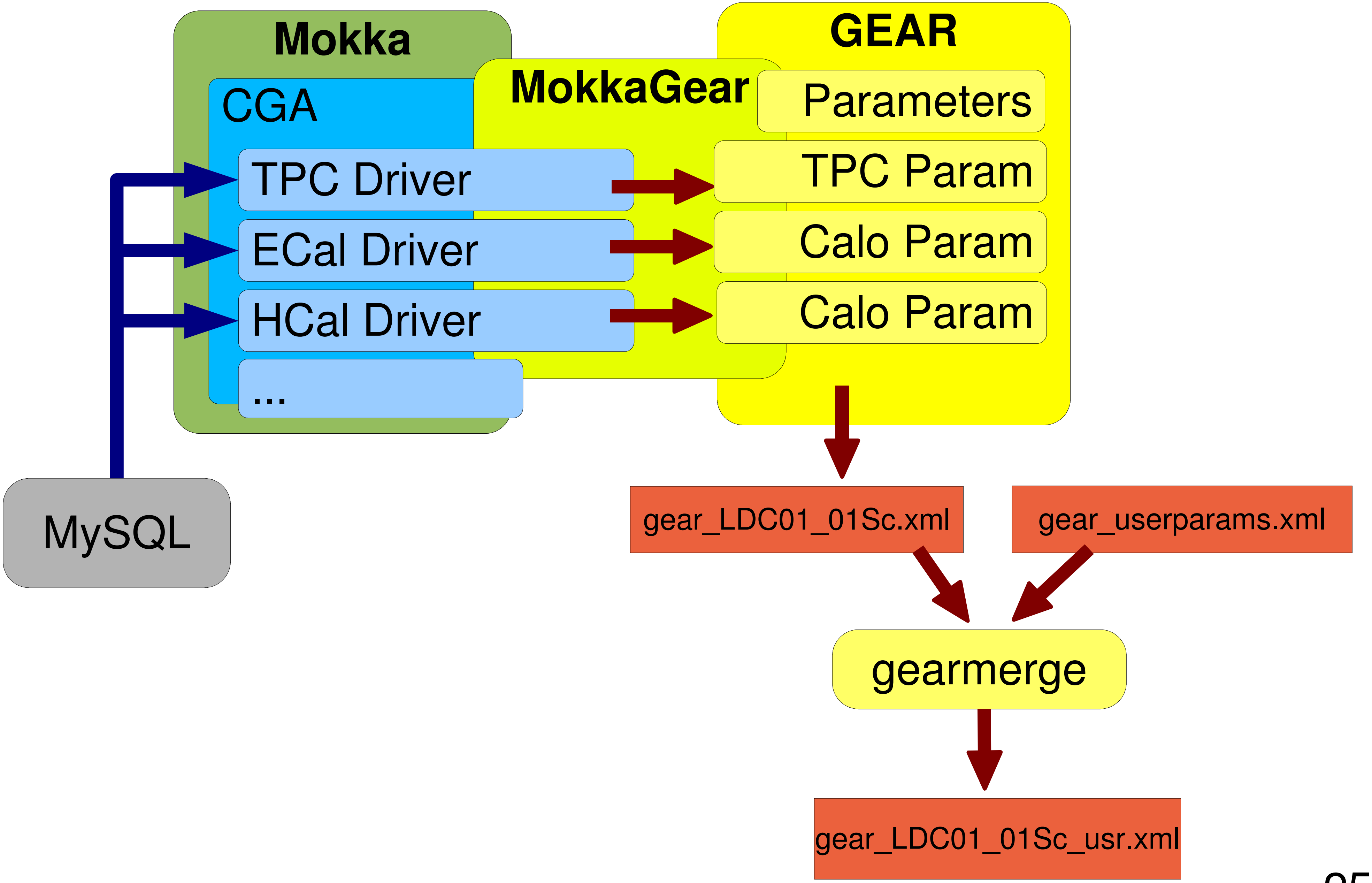
GEometry API for Reconstruction


```
- <gear>
 - <!--
   Example XML file for GEAR describing the LDC detector
   -->
 - <detectors>
  - <detector id="0" name="TPCTest" geartype="TPCParameters" typ
     <maxDriftLength value="2500."/>
     <driftVelocity value=""/>
     <readoutFrequency value="10"/>
     <PadRowLayout2D type="FixedPadSizeDiskLayout" rMin="386.0"
     maxRow="200" padGap="0.0"/>
     <parameter name="tpcRPhiResMax" type="double"> 0.16 </para
     <parameter name="tpcZRes" type="double"> 1.0 </parameter>
     <parameter name="tpcPixRP" type="double"> 1.0 </parameter>
     <parameter name="tpcPixZ" type="double"> 1.4 </parameter>
     <parameter name="tpcIonPotential" type="double"> 0.00000003
    </detector>
  - <detector name="EcalBarrel" geartype="CalorimeterParameters">
     <layout type="Barrel" symmetry="8" phi0="0.0"/>
     <dimensions inner_r="1698.85" outer_z="2750.0"/>
     <layer repeat="30" thickness="3.9" absorberThickness="2.5"/>
     <layer repeat="10" thickness="6.7" absorberThickness="5.3"/>
    </detector>
  - <detector name="EcalEndcap" geartype="CalorimeterParameters">
     <layout type="Endcap" symmetry="2" phi0="0.0"/>
     <dimensions inner_r="320.0" outer_r="1882.85" inner_z="2820.
     <layer repeat="30" thickness="3.9" absorberThickness="2.5"/>
     <layer repeat="10" thickness="6.7" absorberThickness="5.3"/>
    </detector>
  </detectors>
</gear>
```


compatible with US – compact format

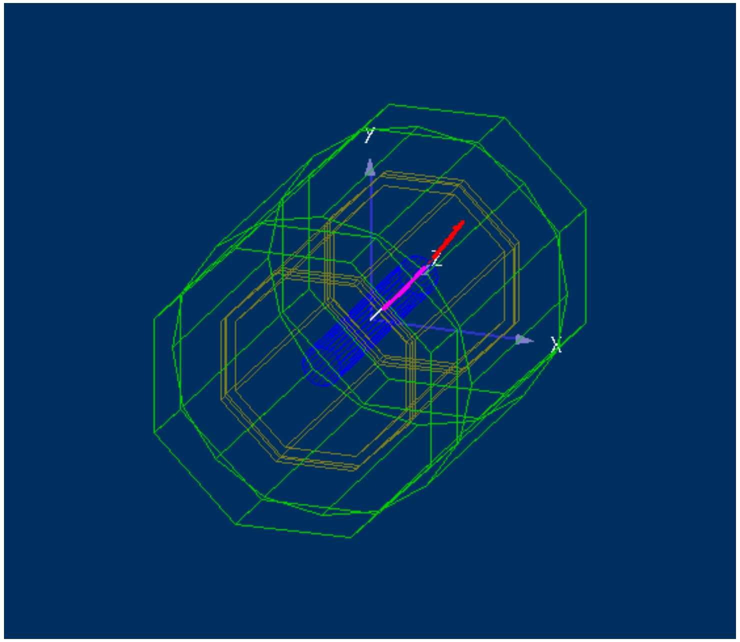
- well defined geometry definition for reconstruction that
  - is flexible w.r.t different detector concepts
  - contains all information needed for reconstruction and analysis
  - provides access to material properties (under development)
- abstract interface (a la LCIO)
- concrete implementation based on XML files
  - and Mokka-CGA

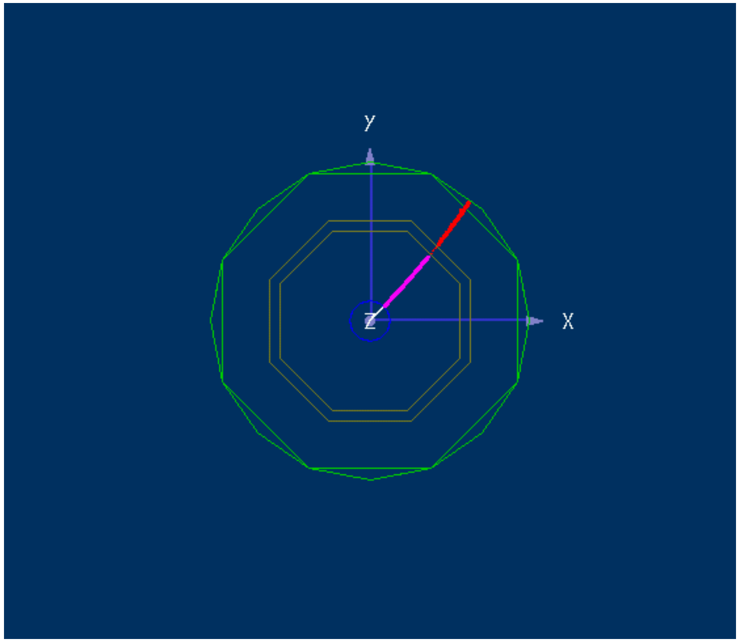
MokkaGear

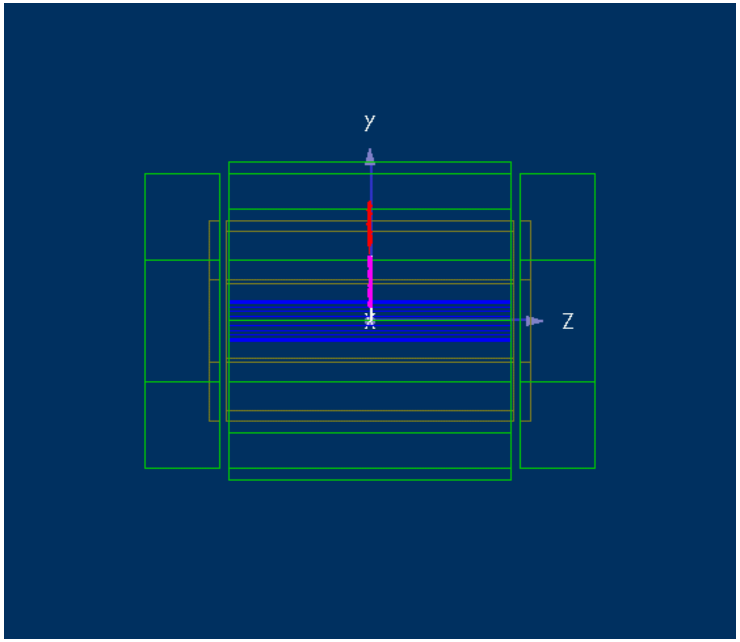

Mokka
CGA
MokkaGear
GEAR
Parameters
TPC Driver
ECal Driver
HCal Driver
...
TPC Param
Calo Param
Calo Param
MySQL
gear_LDC01_01Sc.xml
gear_userparams.xml
gearmerge
gear_LDC01_01Sc_usr.xml

Marlin: CEDViewer

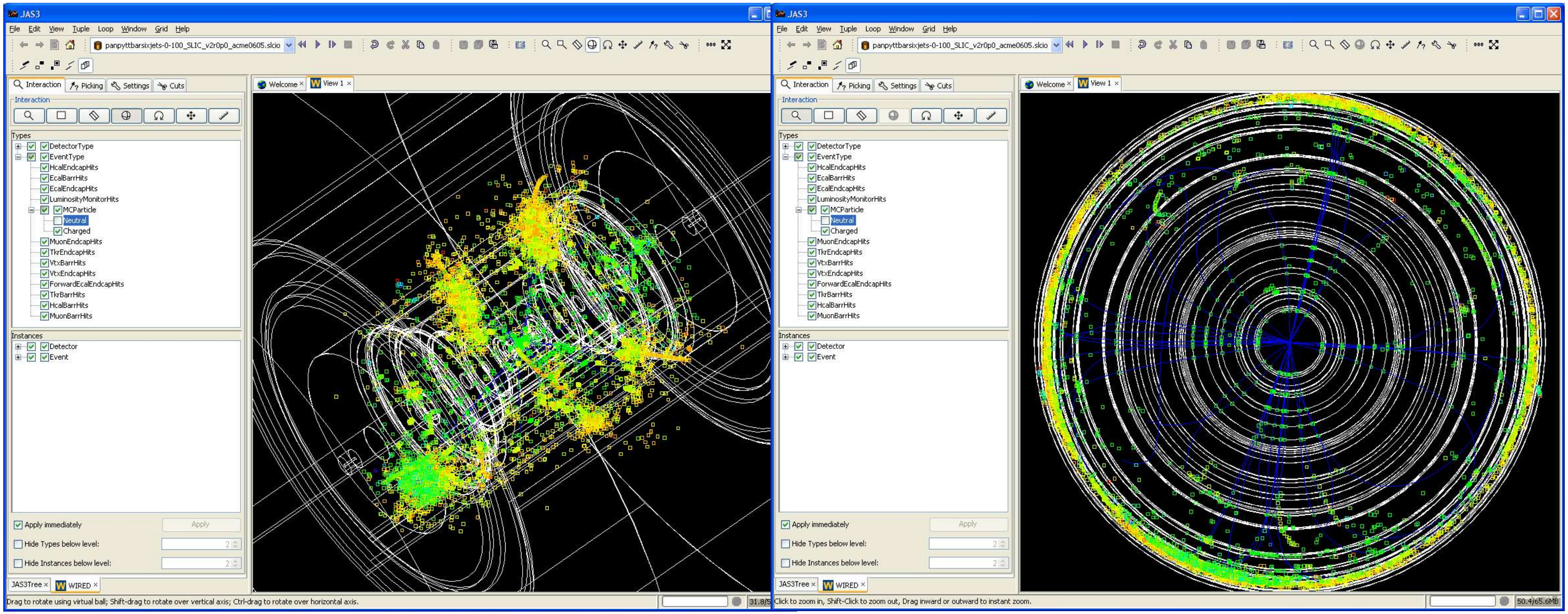
simple example taken from
$MarlinReco/examples/LDC
steer_ldc.xml
gear_ldc.xml



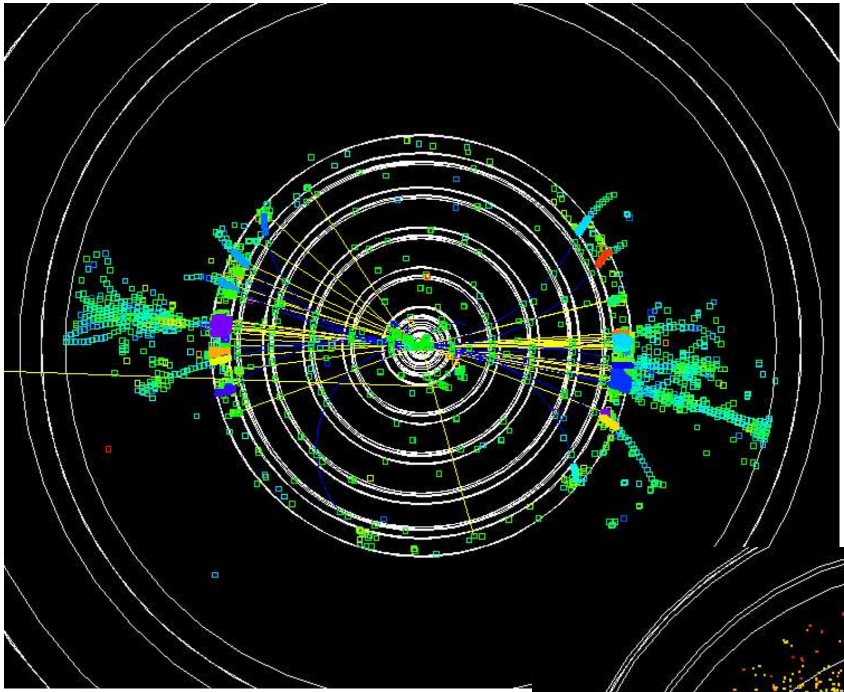




org.lcsim with WIRED

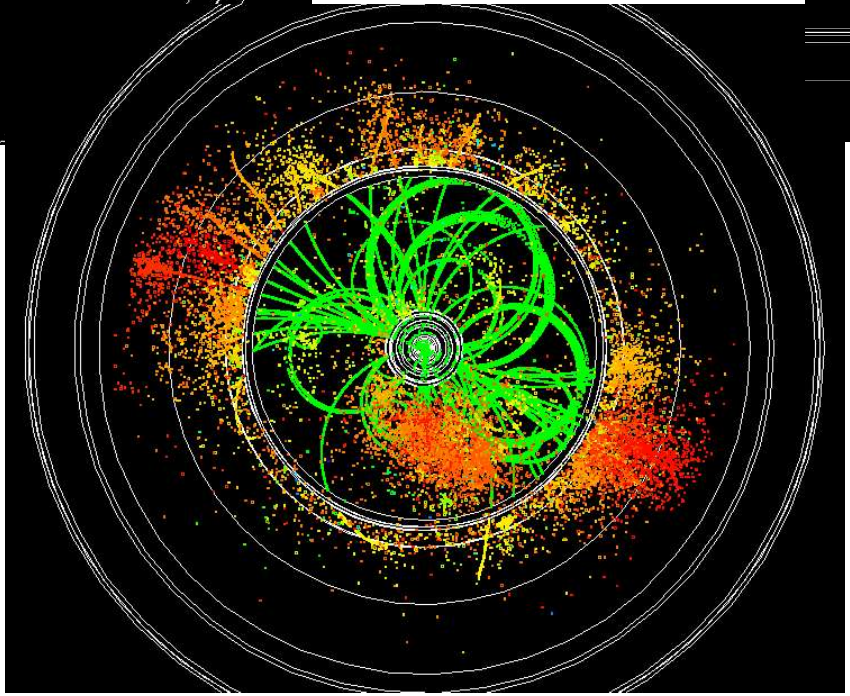


Interoperability: org.lcsim



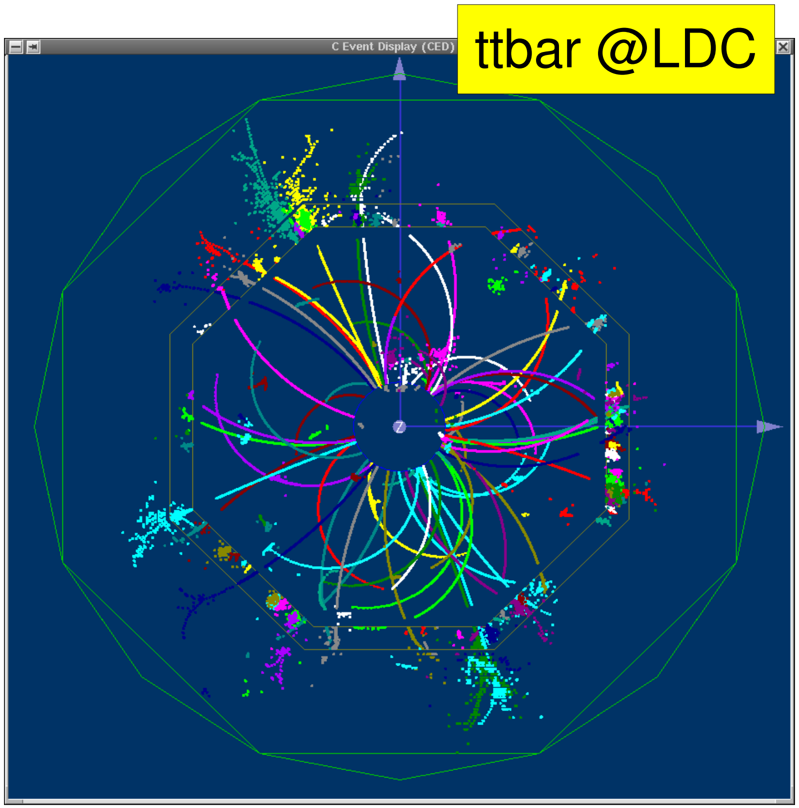
SiD

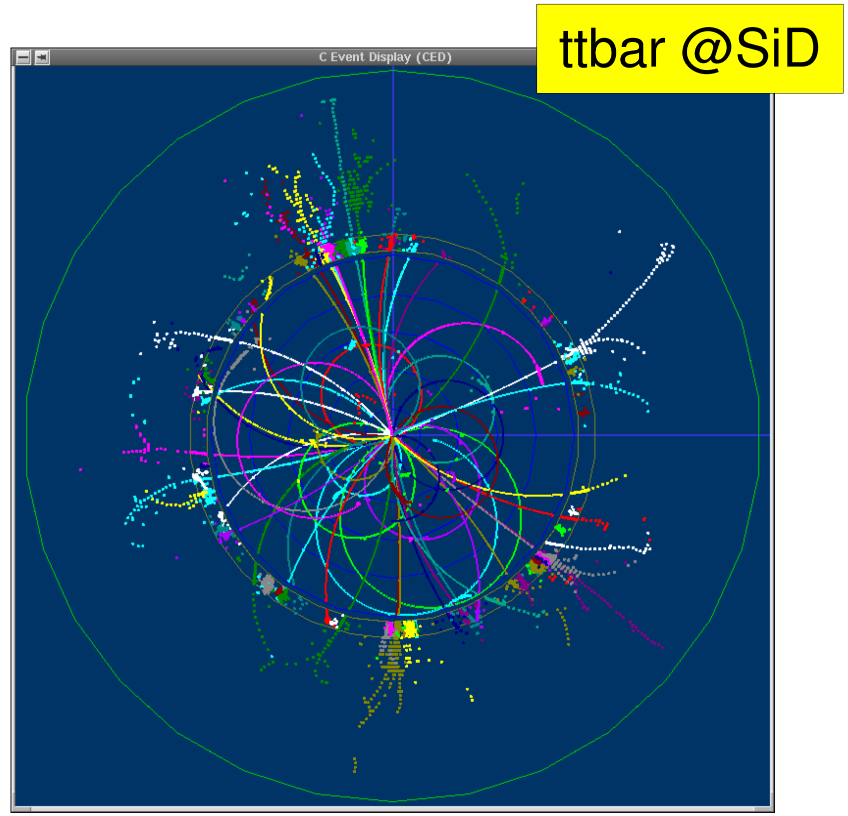
LDC



GLD

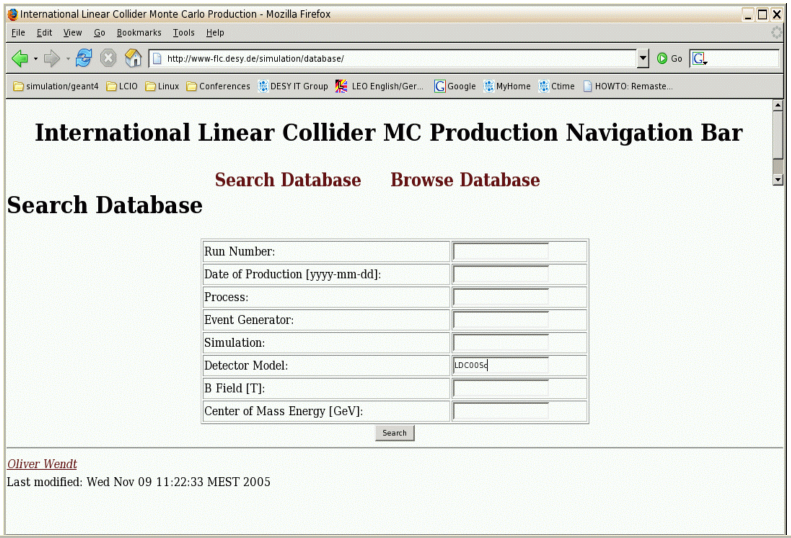
Interoperability: Marlin

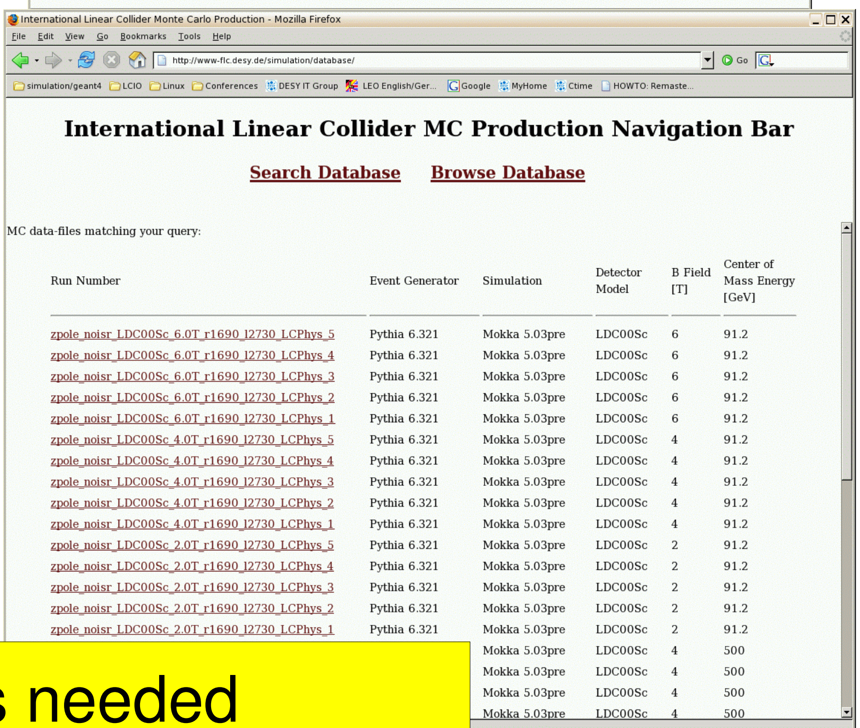




ILC grid - “mass production”

- detector optimization – vary
  - B,R_TPC,L_TPC,...
- need considerable number of events with detector parameter variations for benchmark reactions
- produced these files on the **grid** for VO ILC – 450 kevts:
  - Z0 and uds, ccbb , ttbar, WW, ZH @ 500 GeV
  - 4 detector variants, 3 T and 4 T field
- database with available data files
- use **grid tools** to distribute/download the data !





Run Number	Event Generator	Simulation	Detector Model	B Field [T]	Center of Mass Energy [GeV]
zpole_noisr_LDC00Sc_6.0T_r1690_l2730_LCPhys_5	Pythia 6.321	Mokka 5.03pre	LDC00Sc	6	91.2
zpole_noisr_LDC00Sc_6.0T_r1690_l2730_LCPhys_4	Pythia 6.321	Mokka 5.03pre	LDC00Sc	6	91.2
zpole_noisr_LDC00Sc_6.0T_r1690_l2730_LCPhys_3	Pythia 6.321	Mokka 5.03pre	LDC00Sc	6	91.2
zpole_noisr_LDC00Sc_6.0T_r1690_l2730_LCPhys_2	Pythia 6.321	Mokka 5.03pre	LDC00Sc	6	91.2
zpole_noisr_LDC00Sc_6.0T_r1690_l2730_LCPhys_1	Pythia 6.321	Mokka 5.03pre	LDC00Sc	6	91.2
zpole_noisr_LDC00Sc_4.0T_r1690_l2730_LCPhys_5	Pythia 6.321	Mokka 5.03pre	LDC00Sc	4	91.2
zpole_noisr_LDC00Sc_4.0T_r1690_l2730_LCPhys_4	Pythia 6.321	Mokka 5.03pre	LDC00Sc	4	91.2
zpole_noisr_LDC00Sc_4.0T_r1690_l2730_LCPhys_3	Pythia 6.321	Mokka 5.03pre	LDC00Sc	4	91.2
zpole_noisr_LDC00Sc_4.0T_r1690_l2730_LCPhys_2	Pythia 6.321	Mokka 5.03pre	LDC00Sc	4	91.2
zpole_noisr_LDC00Sc_4.0T_r1690_l2730_LCPhys_1	Pythia 6.321	Mokka 5.03pre	LDC00Sc	4	91.2
zpole_noisr_LDC00Sc_2.0T_r1690_l2730_LCPhys_5	Pythia 6.321	Mokka 5.03pre	LDC00Sc	2	91.2
zpole_noisr_LDC00Sc_2.0T_r1690_l2730_LCPhys_4	Pythia 6.321	Mokka 5.03pre	LDC00Sc	2	91.2
zpole_noisr_LDC00Sc_2.0T_r1690_l2730_LCPhys_3	Pythia 6.321	Mokka 5.03pre	LDC00Sc	2	91.2
zpole_noisr_LDC00Sc_2.0T_r1690_l2730_LCPhys_2	Pythia 6.321	Mokka 5.03pre	LDC00Sc	2	91.2
zpole_noisr_LDC00Sc_2.0T_r1690_l2730_LCPhys_1	Pythia 6.321	Mokka 5.03pre	LDC00Sc	2	91.2
		Mokka 5.03pre	LDC00Sc	4	500
		Mokka 5.03pre	LDC00Sc	4	500
		Mokka 5.03pre	LDC00Sc	4	500
		Mokka 5.03pre	LDC00Sc	4	500

- provide the simulated data that's needed
- exercise the software & computing infrastructure

Summary

- LCIO: Standard for ILC.

Controversial solutions on different sides of the Atlantic:

- Conditions data: Differences between LCCD and org.lcsim are of technical, not at all of philosophical nature.
- Common source for the geometry: Contradictory approaches. I don't dare to make any recommendation. Is there any convincing solution in other experiments?
- Event viewer: Many experiments use a variety of event viewers.
- Framework: org.lcsim vs Marlin: Unfortunate duplication of work ... but: an infinite amount of frameworks conceivably may meet our needs perfectly.
- Choice of language: Java or C++?
  Every decent framework is requested to support both of them!

Conclusion

About religious wars

In Germany, we have quite some experience with them. Example: The War of Thirty Years (1618 – 1648) between catholic and protestant parties.
At the end, there were only losers, and the population decreased by 30% (in some areas by 100%). It was ended by a treaty called “Peace of Westphalia”.
And what was the outcome of this treaty? Religious freedom?

Conclusion

About religious wars

In Germany, we have quite some experience with them. Example:
The War of Thirty Years (1618 – 1648) between catholic and protestant parties.
At the end, there were only losers, and the population decreased by 30% (in some areas by 100%). It was ended by a treaty called "Peace of Westphalia".
And what was the outcome of this treaty? Religious freedom?
Absolutely wrong!
The outcome was:
"The regional authorities decree the religion of their dependent subjects"

Let's avoid religious wars!

Save the manpower to work on real issues like PFA