Introduction to Software Tools

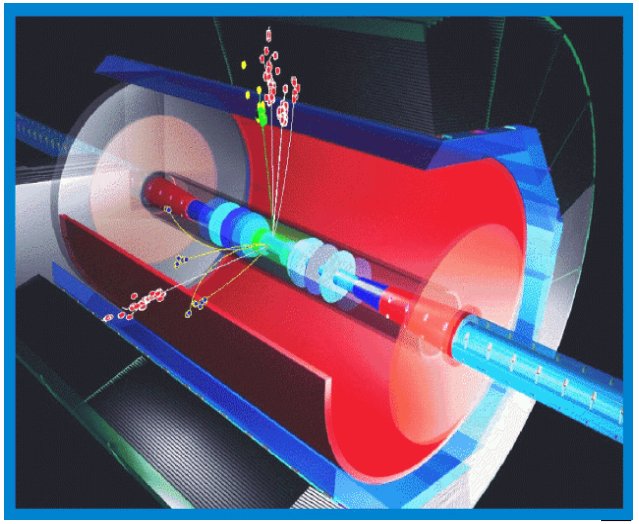
ILC-LDC Simulation and Reconstruction

Frank Gaede
DESY

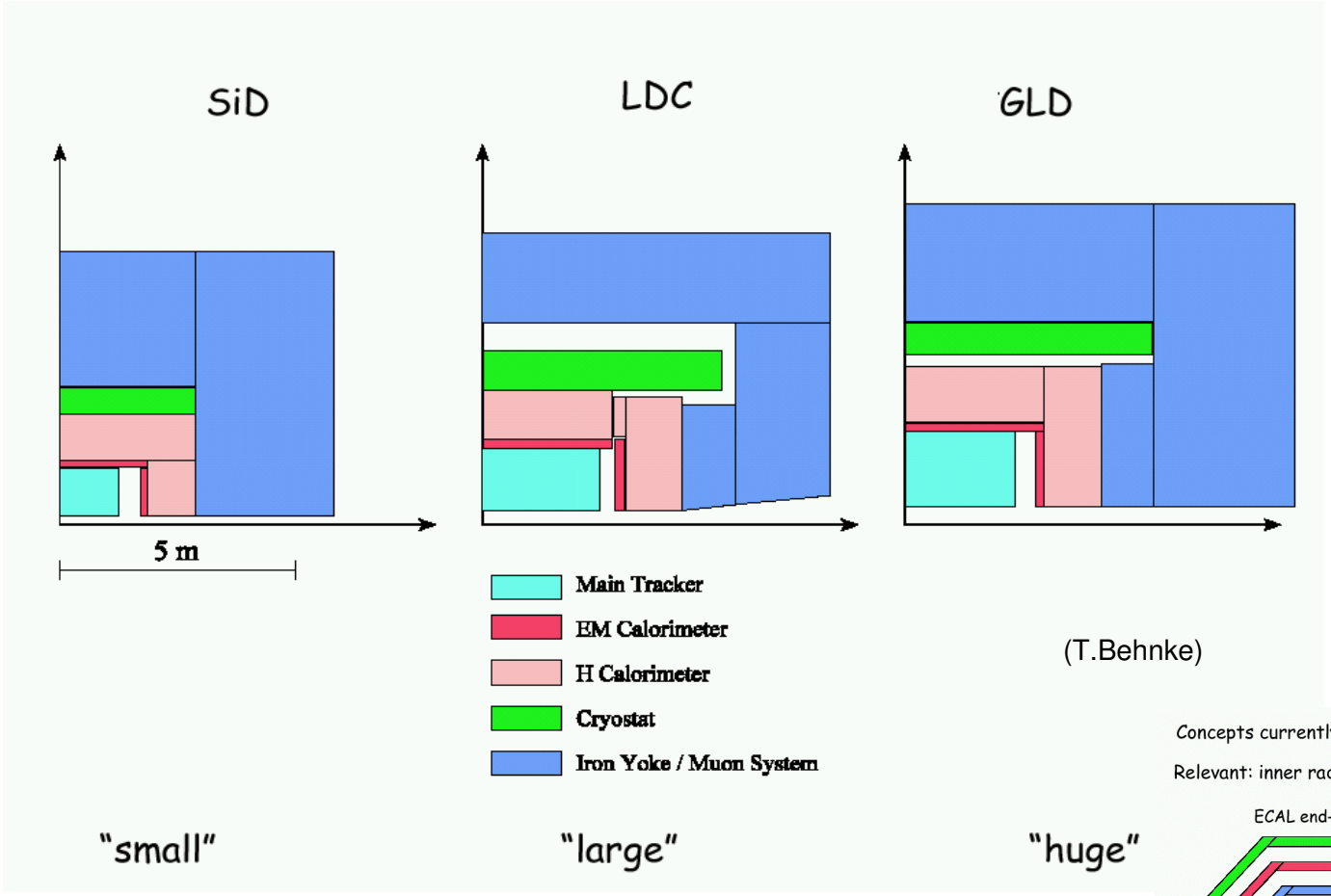
ECFA ILC Workshop, Vienna
November 14-17, 2005

Outline

- Introduction - overview international software
- Central tools for LDC study
  - LCIO – data model & persistency
  - Simdet – fast simulation
  - Brahms – geant3 full simulation and reconstruction
  - Mokka – geant4 full simulation
  - **Marlin** – C++ reconstruction framework
  - LCCD - conditions data toolkit
  - **GEAR** – geometry description
- Summary & Outlook



Detector Concept Study

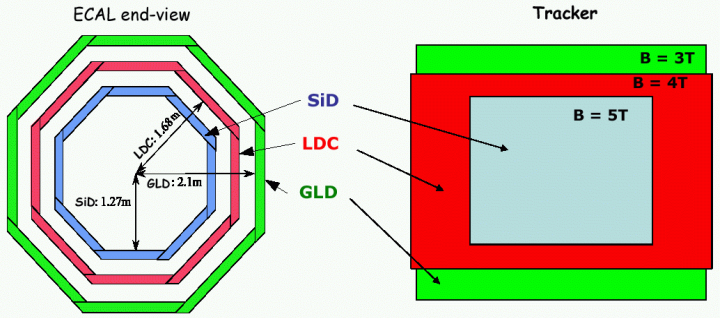


three interregional detector concept studies ongoing

(T.Behnke)

Concepts currently studies differ mainly in SIZE and aspect ratio

Relevant: inner radius of ECAL: defines the overall scale



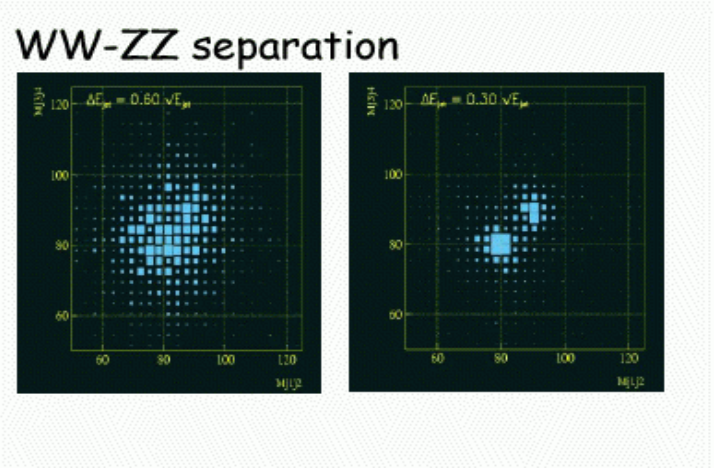
SiD: Silicon based concept

GLD: even larger detector concept

LDC: large detector concept

Need (common?) **Simulation** and **Reconstruction** software to study detector concepts' performance !

Reconstruction @ the ILC



- general ILC detector features:
  - precision tracking
  - precision vertexing
  - high granularity in calorimeters
    - (Ecal ~1cm, Hcal ~1-5cm)
- important: very high jet-mass resolution ~30%/sqrt(E/GeV)

Particle Flow

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

dominant contribution (E<50 GeV):

- Hcal resolution
- confusion term

$$\sigma_{E_{jet}}^2 = \epsilon_{trk}^2 \sum_i E_{trk,i}^4 + \epsilon_{ECal}^2 E_{ECal} + \epsilon_{HCal}^2 E_{HCal} + \sigma_{confusion}^2$$

$$\epsilon_{trk} = \delta(1/p) \approx 5 \cdot 10^{-5}, \quad \epsilon_{ECal} = \frac{\delta E}{\sqrt{E}} \approx 0.1, \quad \epsilon_{HCal} \approx 0.5$$

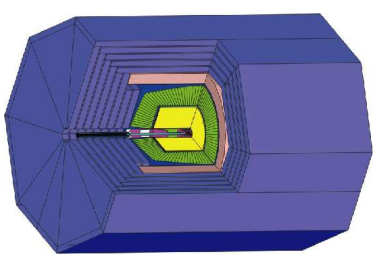
Need (common?) software framework
to develop, check and compare
different Pflow algorithms

ILC software packages

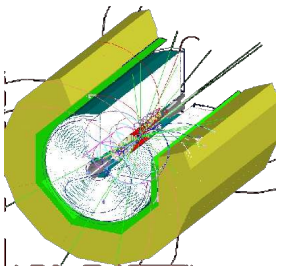
	Description	Detector	Language	IO-Format	Region
Simdet	fast Monte Carlo	TeslaTDR	Fortran	StdHep/LCIO	EU
SGV	fast Monte Carlo	simple Geometry, flexible	Fortran	None (LCIO)	EU
Lelaps	fast Monte Carlo	SiD, flexible	C++	SIO, LCIO	US
Mokka	full simulation – Geant4	TeslaTDR, LDC, flexible	C++	ASCI, LCIO	EU
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, LCIO	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 Simulation Frameworks (Geant4)

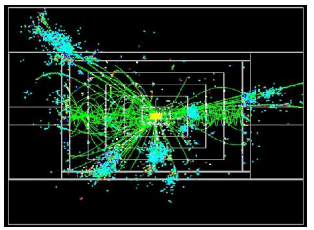
- *Geant4, StdHep and LCIO are common feature*
- *Each trying to be generic with different approach*
  different ways to define geometries



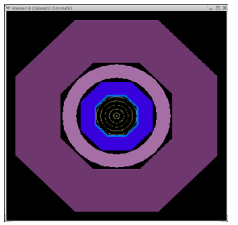
GLD,…



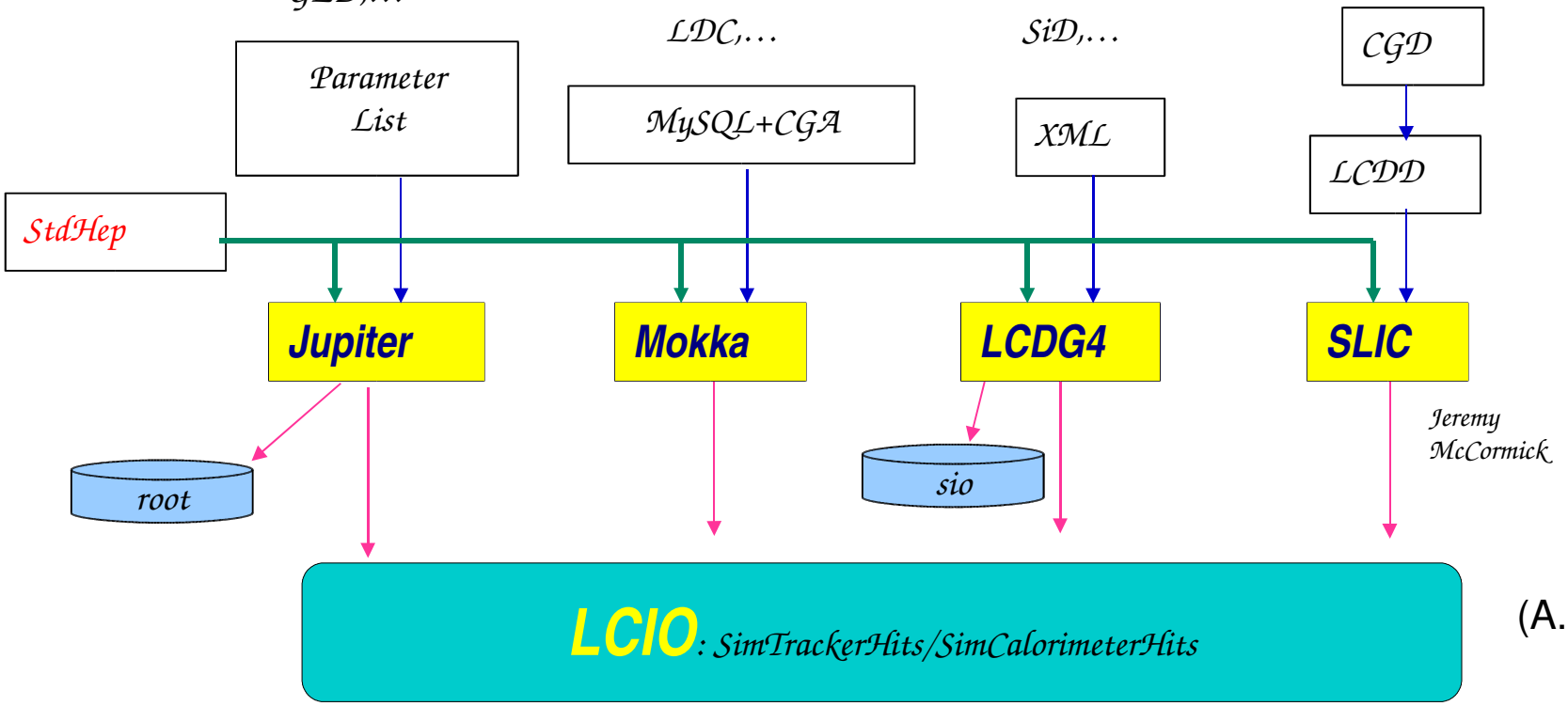
LDC,…



SiD,…

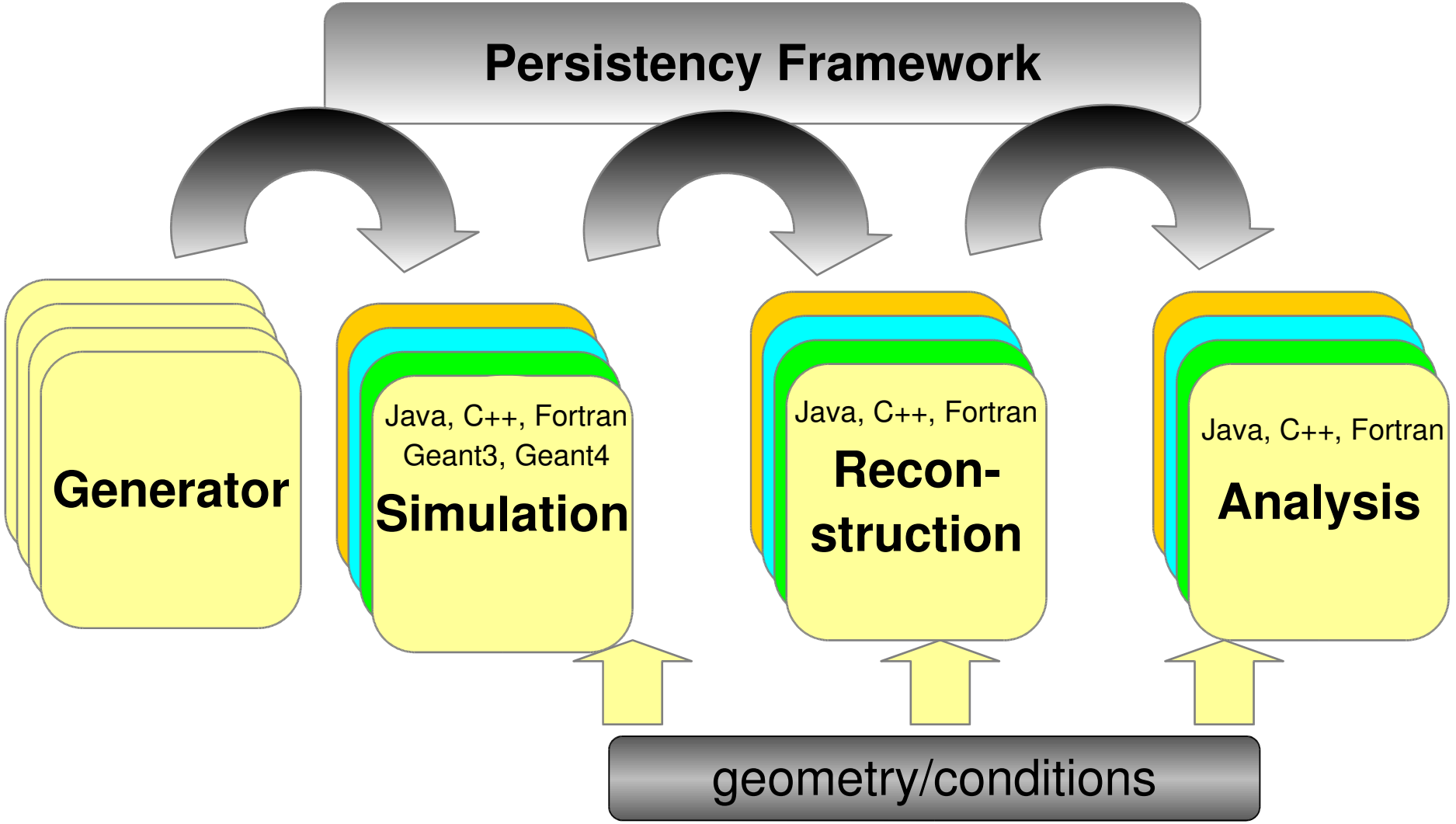


SiD,…

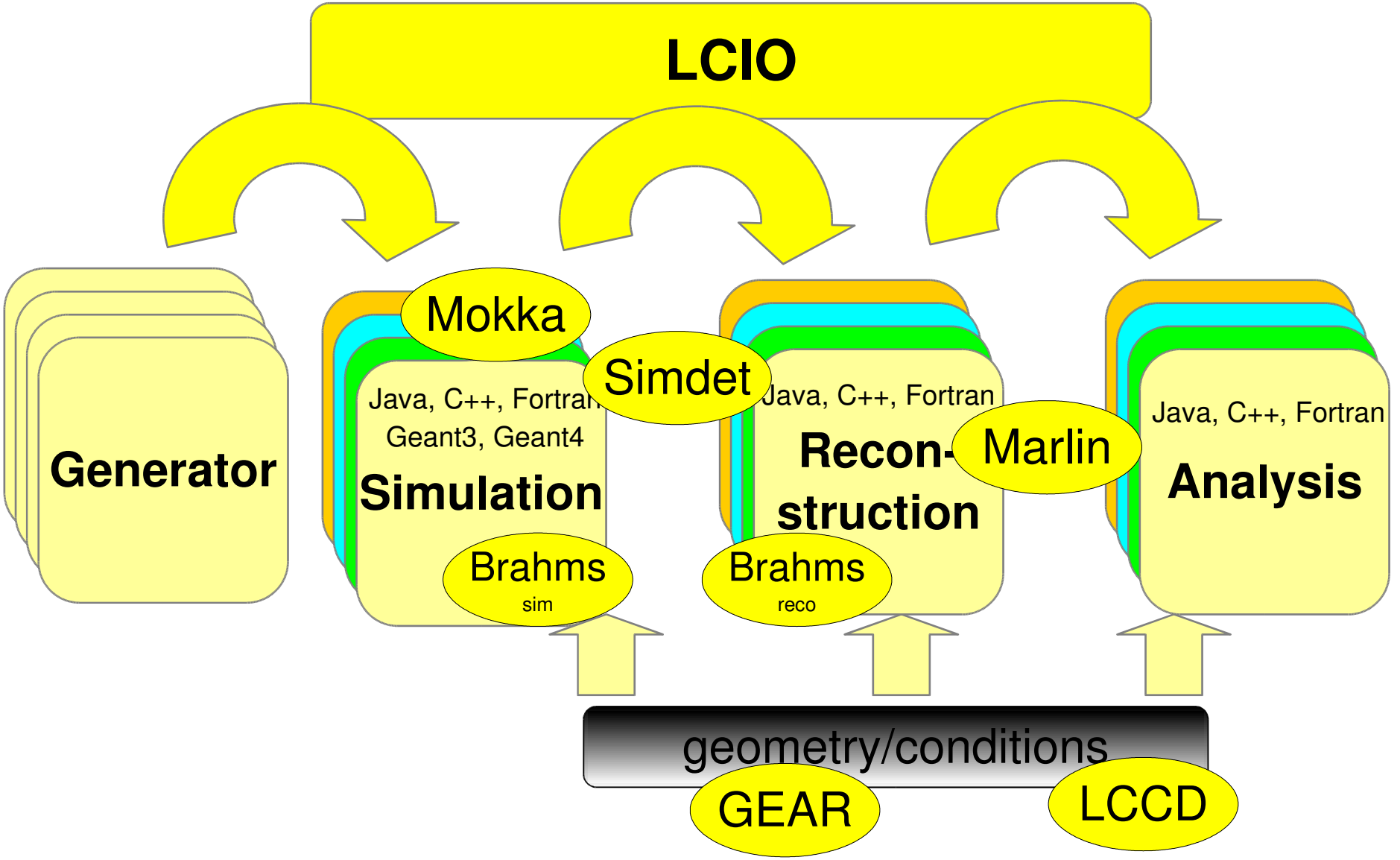


(A.Miyamoto)

ILC software chain



ILC software tools used for LDC



LCIO overview

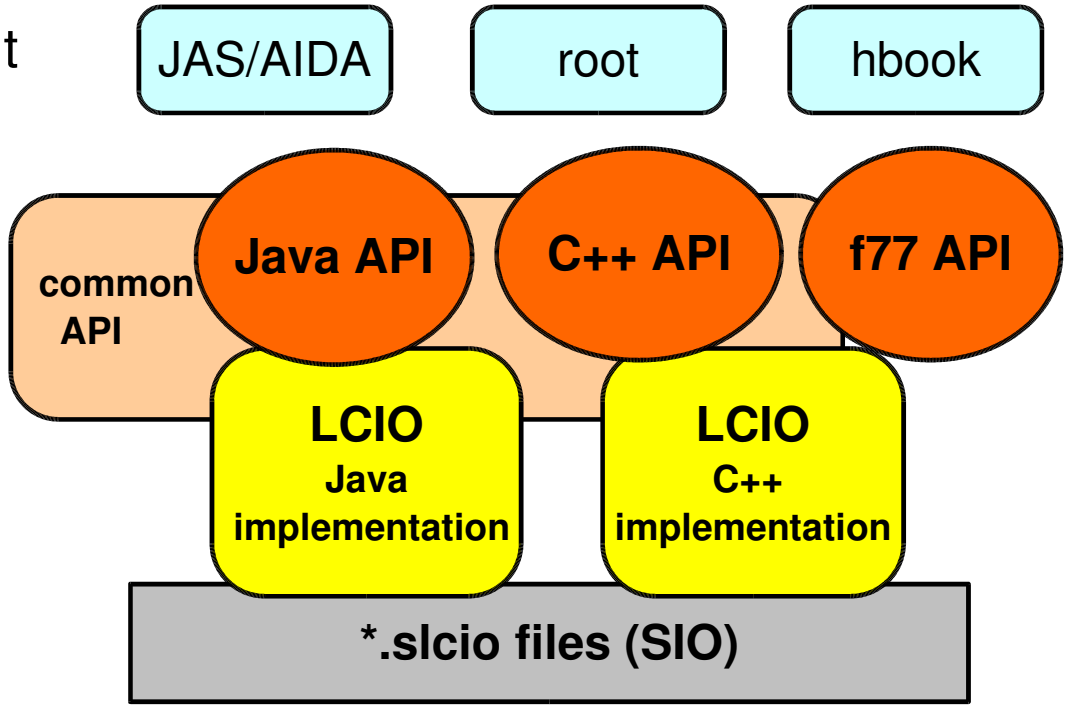
- DESY and SLAC joined project:
  - provide common basis for ILC software
- Features:
  - Java, C++ and f77 (!) API
  - extensible data model for current and future simulation and testbeam studies
  - user code separated from concrete data format
  - no dependency on other frameworks

simple & lightweight

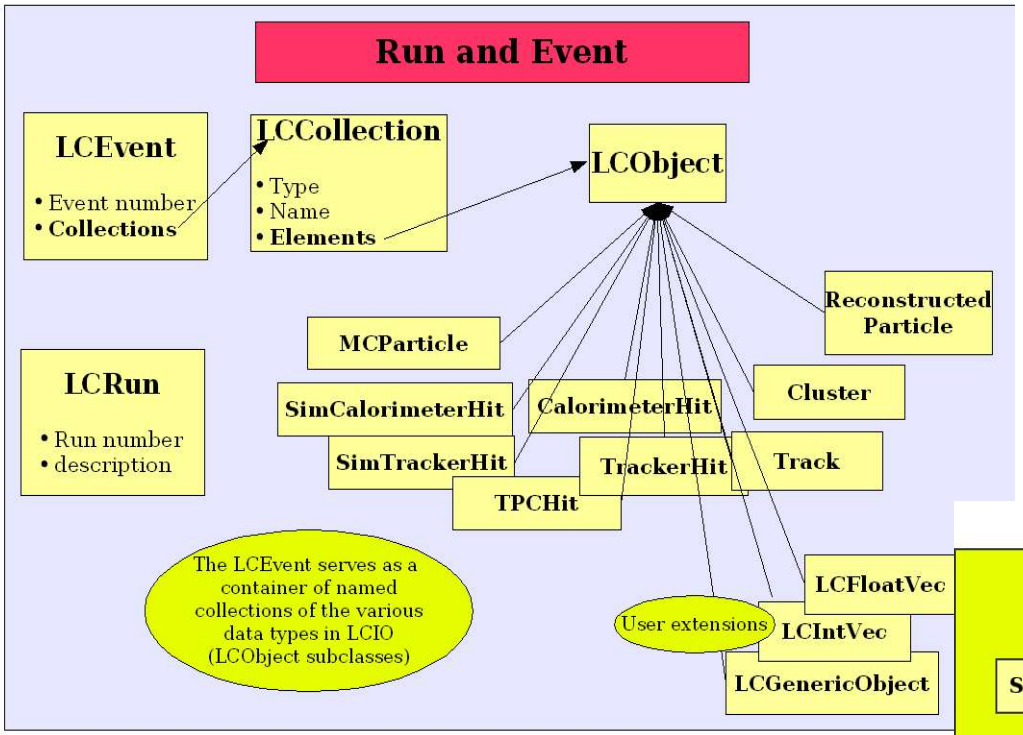
new release: **v01-06**

now de facto standard persistency & datamodel for ILC software

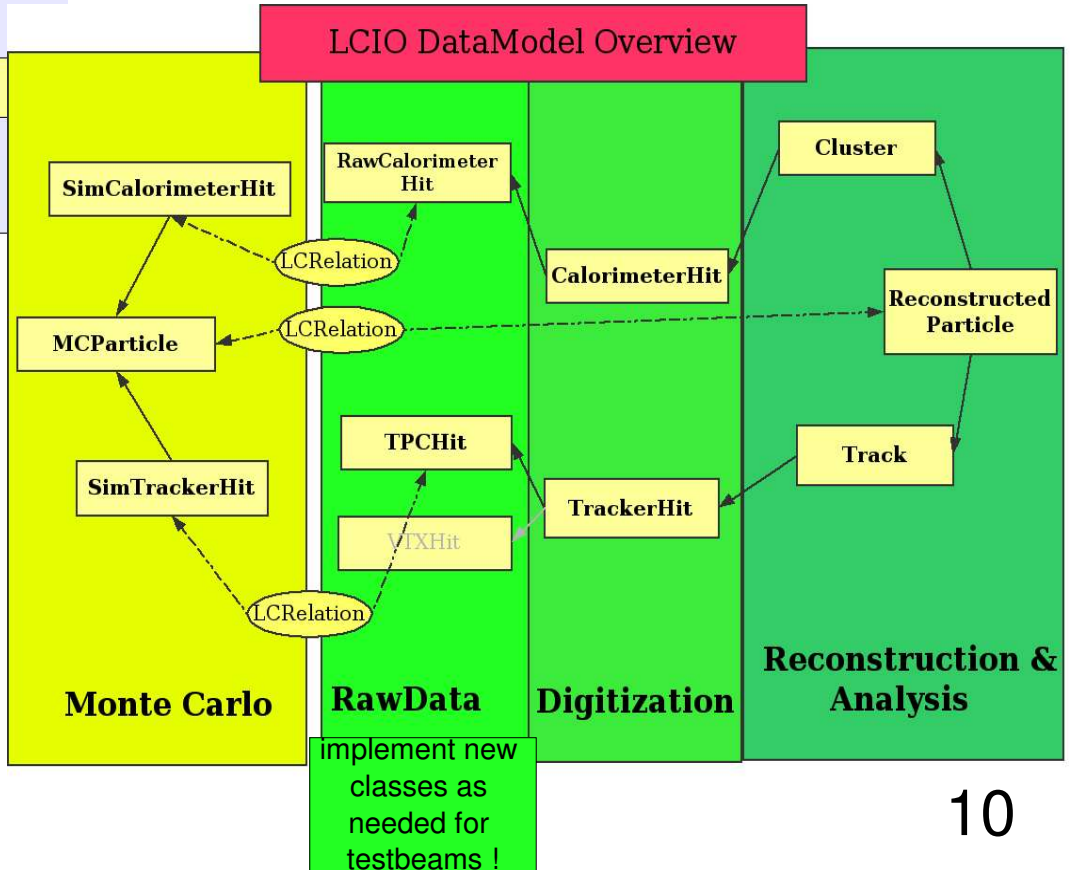
SW-Architecture



LCIO data model



event serves as container of untyped collections



hierarchy of data objects in the event

Simulation tools: Simdet, Brahms

- **SIMDET**
  - parameterized fast Monte Carlo (f77)
  - tracks + cov. matrix and clusters
  - hard coded geometry: TESLA TDR Detector

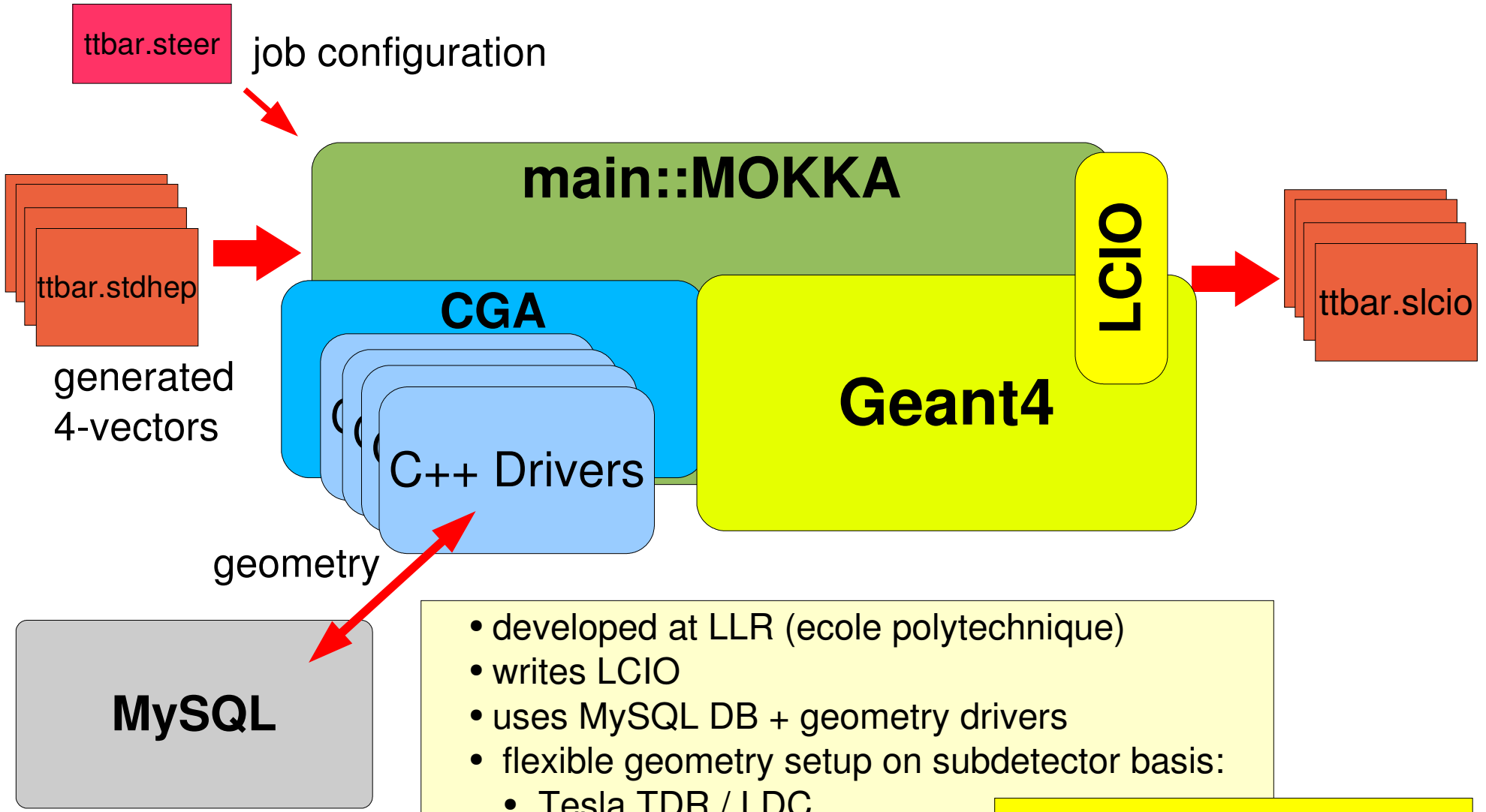
writes LCIO

- **Brahms**
  - geant3 full simulation (f77)
  - hard coded geometry: TESLA TDR Detector
  - full standalone reconstruction part (pflow)
    - tracking based on LEP reconstruction code

reads + writes LCIO

for download (cvs web interface)
and more information:
http://www-zeuthen.desy.de/linear_collider

Mokka overview



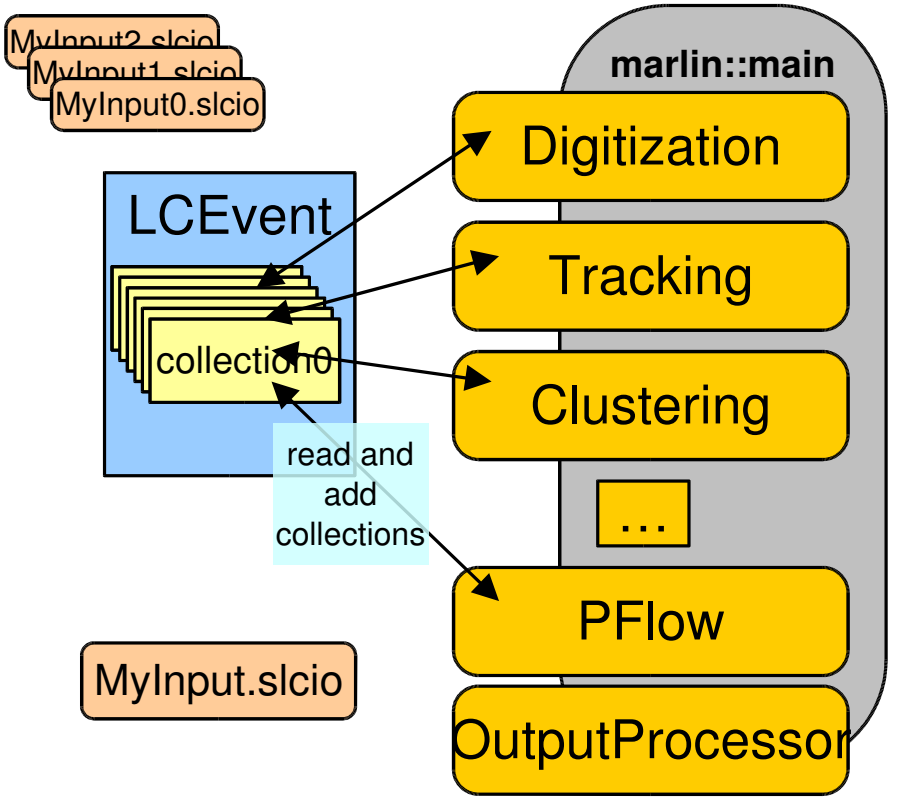
- developed at LLR (ecole polytechnique)
- writes LCIO
- uses MySQL DB + geometry drivers
- flexible geometry setup on subdetector basis:
  - Tesla TDR / LDC
  - SiD
  - Calice testbeam prototypes
- used in European/ LDC study

- new feature:
scalable detector models
-> detector optimization !!

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 transient data model
- software modules called Processors
- provides main program !
- provides simple user steering:
  - program flow (active processors)
  - user defined variables
    - per processor and global
  - input/output files
  - **Plug&Play** of processors



Gear

GEometry **A**PI for **R**econstruction

```
- <gear>
  - <!--
      Example XML file for GEAR describing the LDC detector
    -->
  - <detectors>
    - <detector id="0" name="TPCTest" geartype="TPCParameters" type
        <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>
```

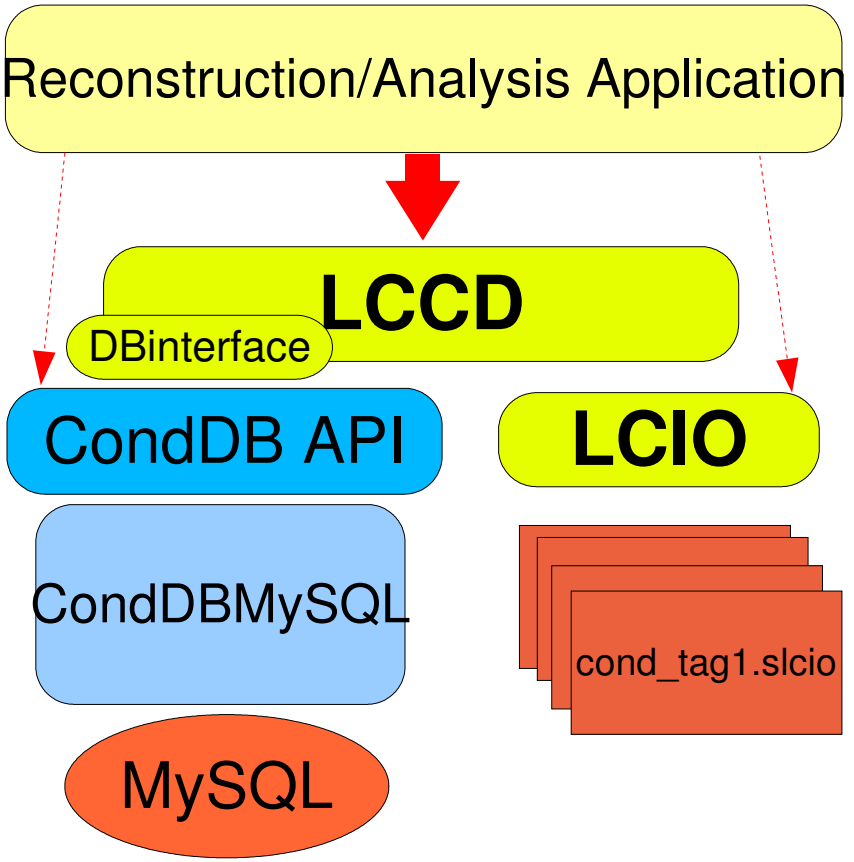
compatible with US – compact format

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

LCCD

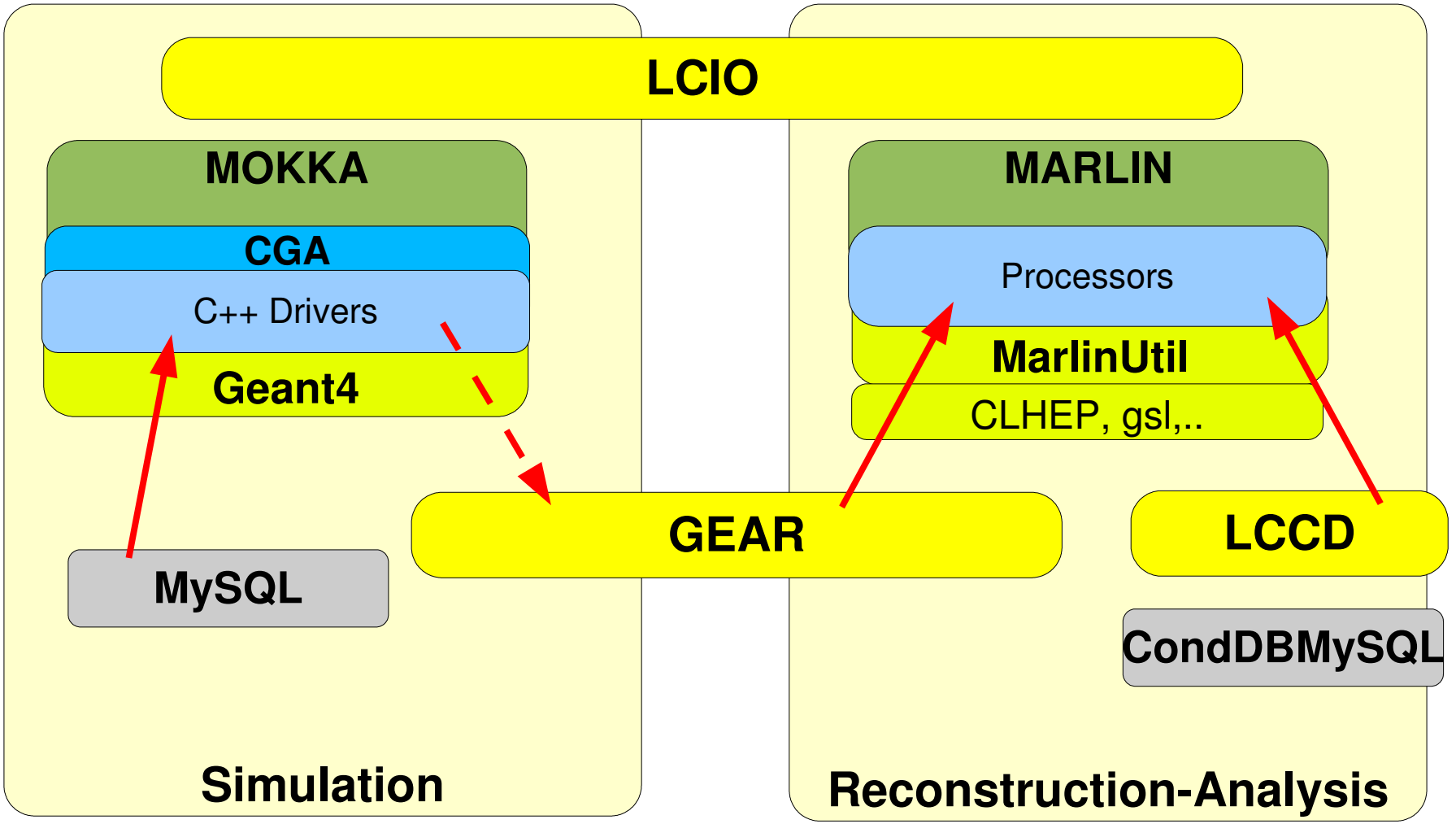
Linear **C**ollider **C**onditions **D**ata Toolkit

- 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 timestamp)
    - horizontally (all versions for tag)



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

LDC simulation framework



MarlinReco

- Marlin serves as a **framework** for the distributed development of reconstruction algorithms
  - provides a well defined modularity
- MarlinReco is a **toolkit** which aims at providing reconstruction algorithms for detector concept studies
  - (almost) complete set of standard reconstruction (pflow)
  - cheaters for cross checks (and replacements)
  - all processors can seamlessly be combined together with other reconstruction code or plugged into your analysis
    - e.g. together with MAGIC-clustering

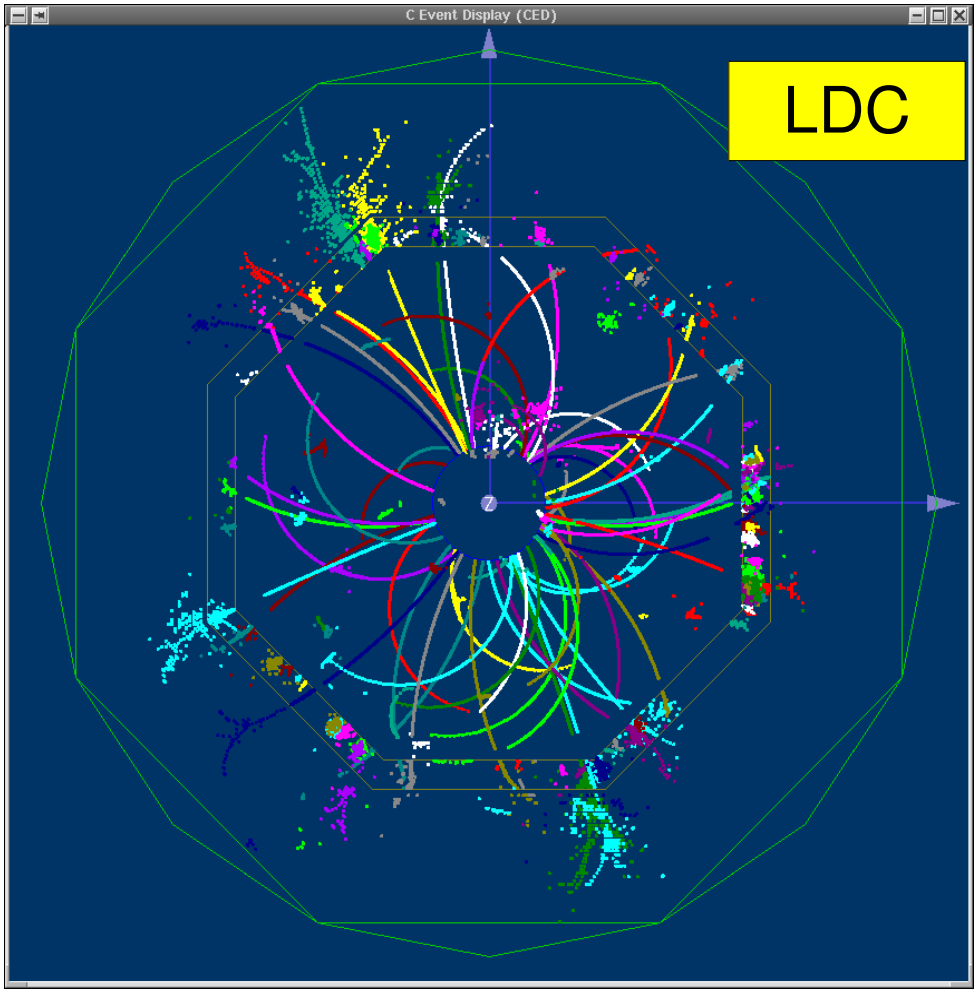
MarlinReco packages

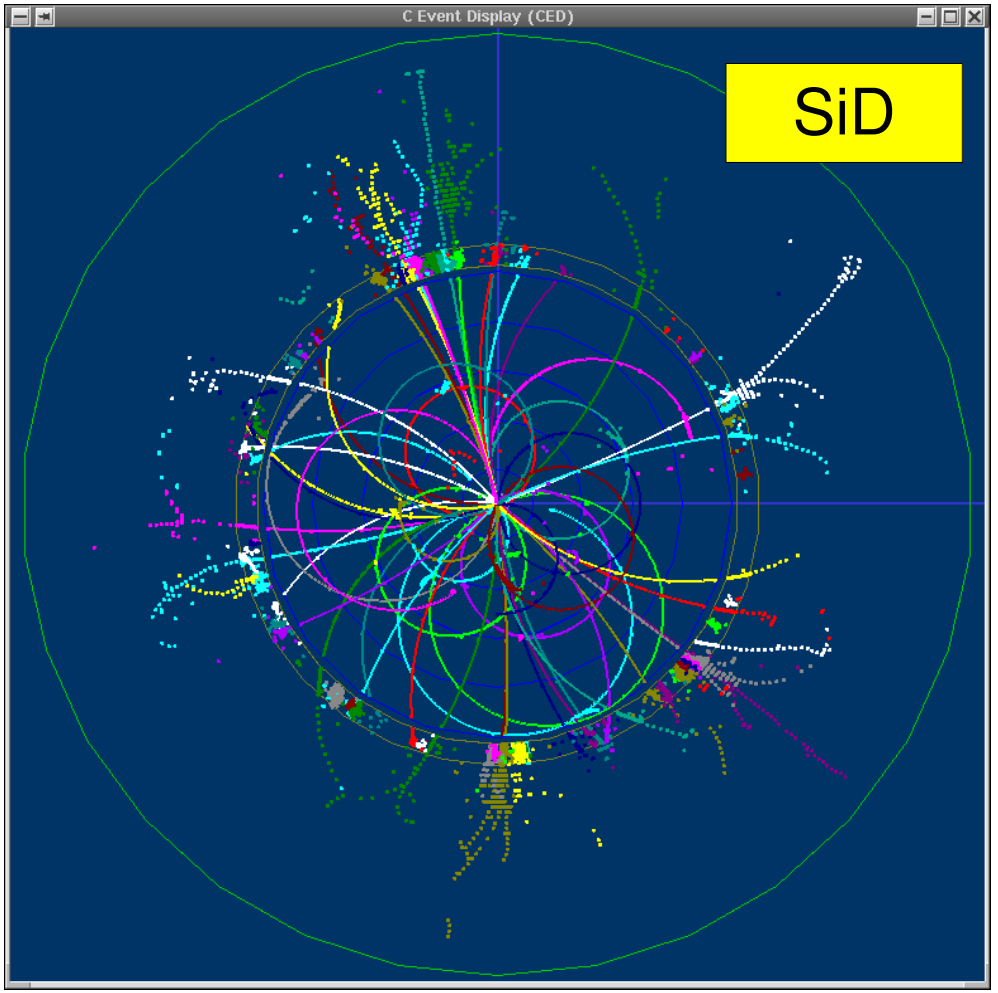
- **TrackDigi**
  - smearing in TPC
- **CaloDigi**
  - calibration, E-cut, ganging
- **Tracking**
  - central tracks in TPC+VTX
  - track cheater
- **Clustering**
  - trackwise clustering
  - cluster cheater
- **Pflow**
  - track-cluster match, PID
- **Analysis**
  - event shapes
  - jet finder
- MarlinUtil, CEDViewer

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

ttbar events with MarlinReco

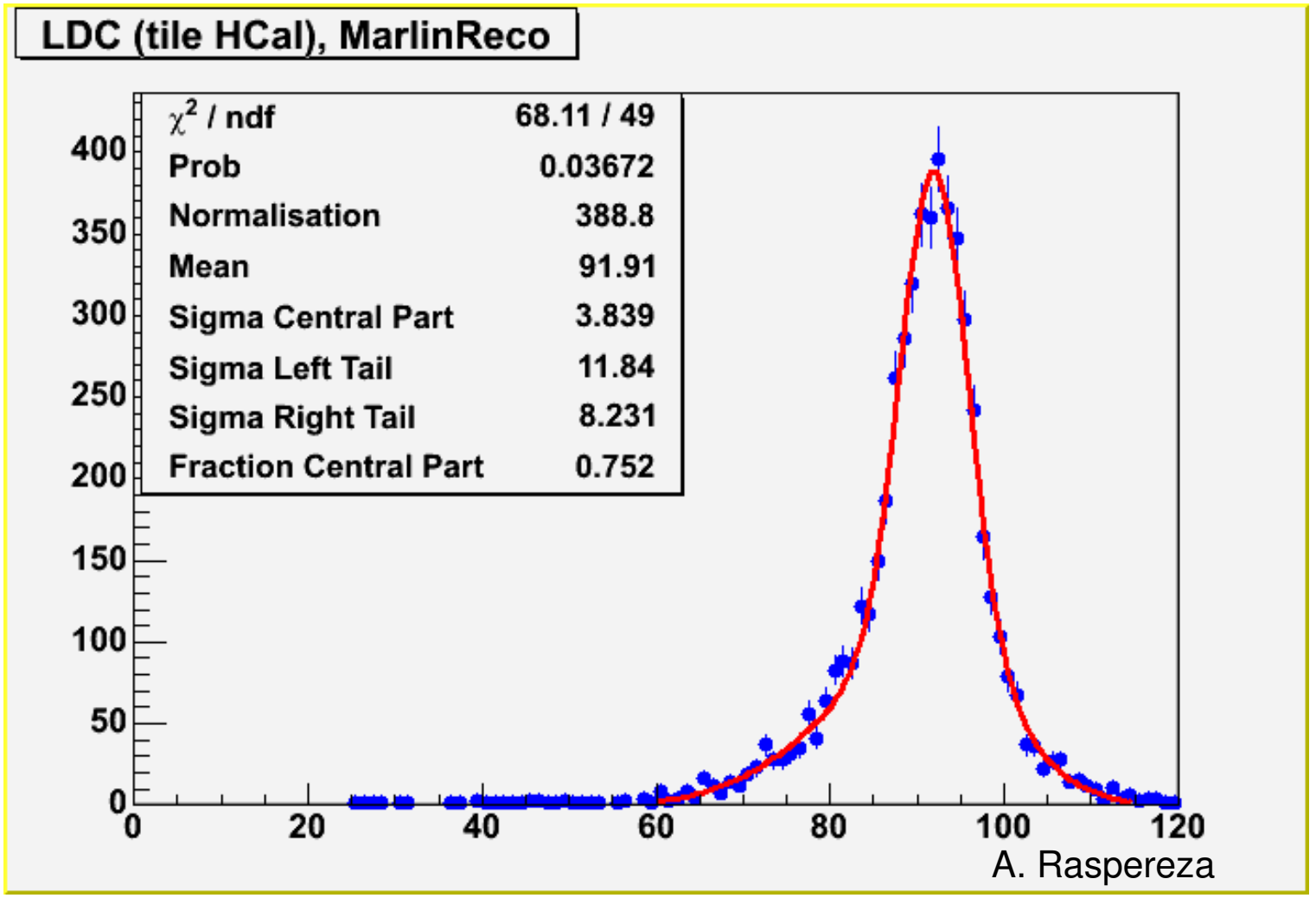
No cheaters, only full reconstruction





Initial results

- e+e- –> Z –> qqbar using full reconstruction



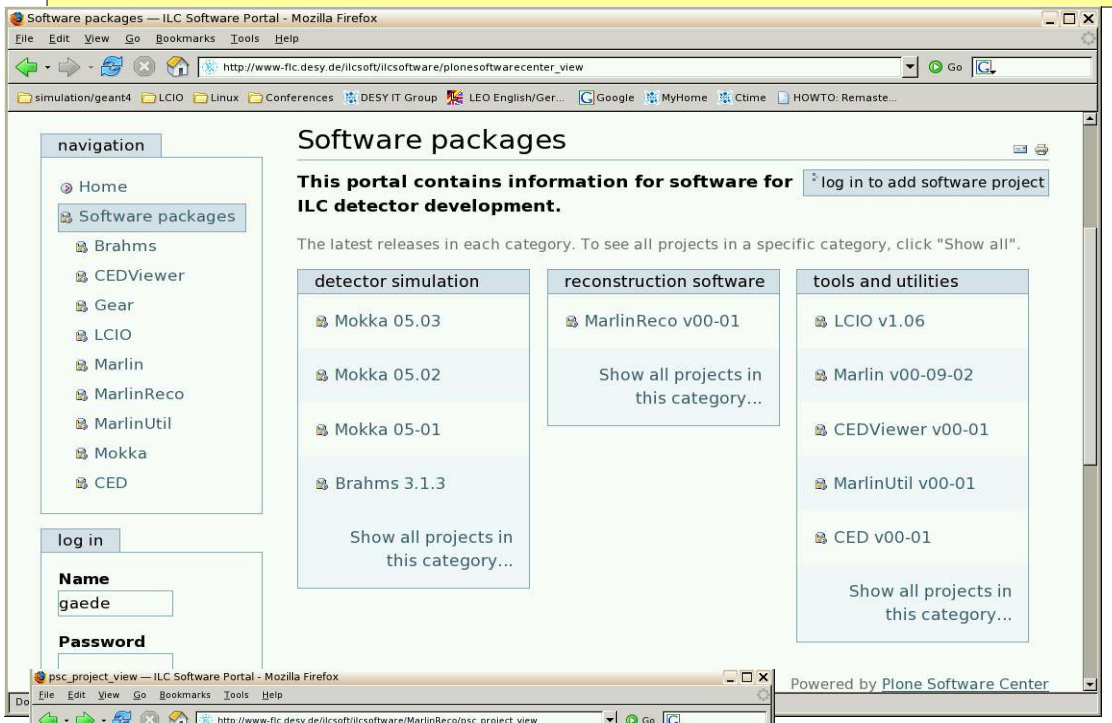
"How can I create my own Zpole plot ?"

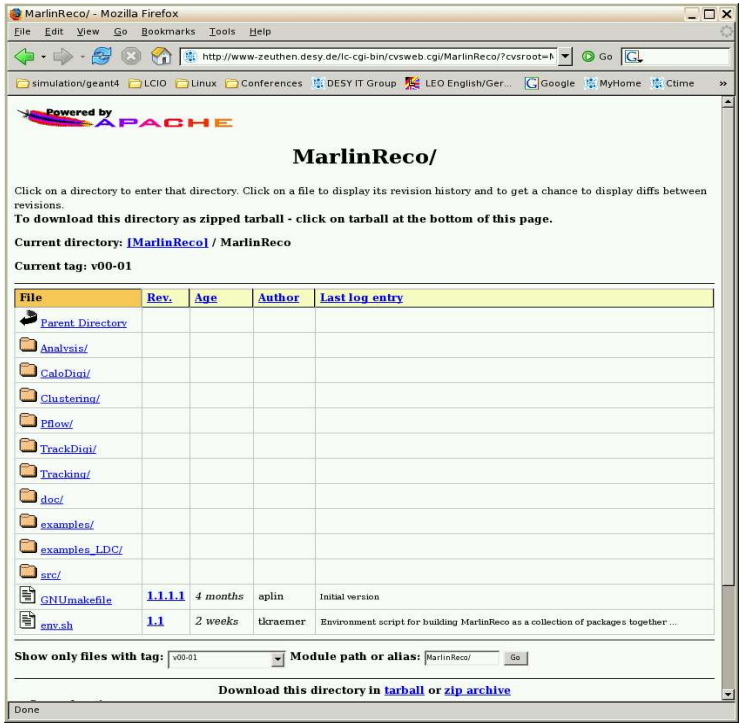
"Download, install and run the software tools !"

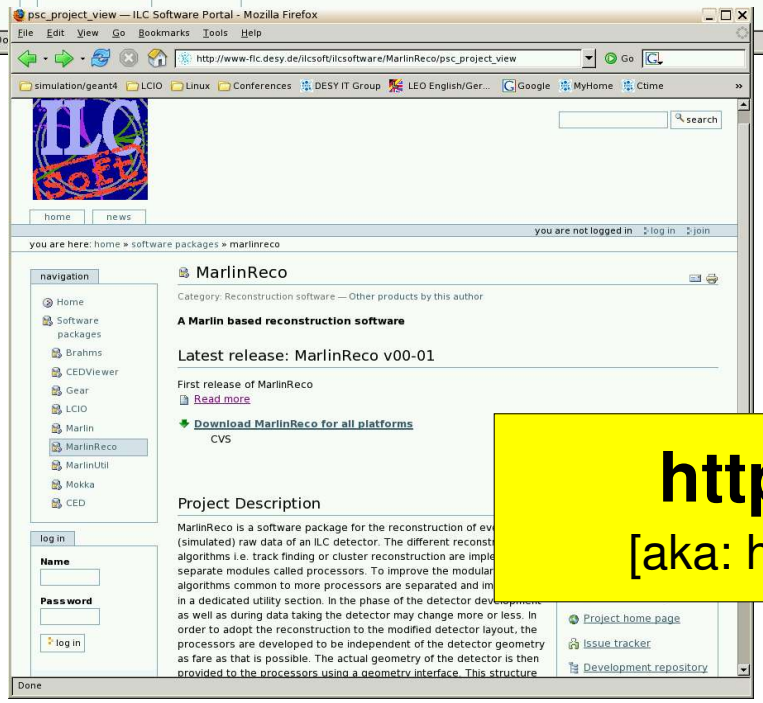
"Where should I start?"

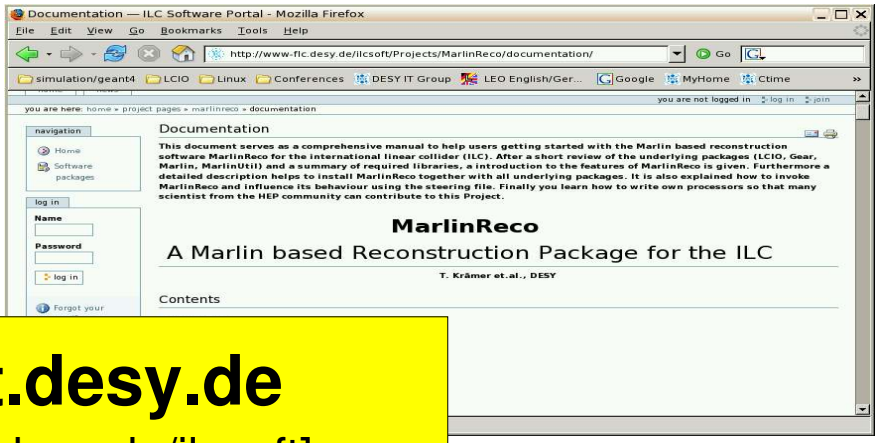
"At the software portal !"

ILC software portal







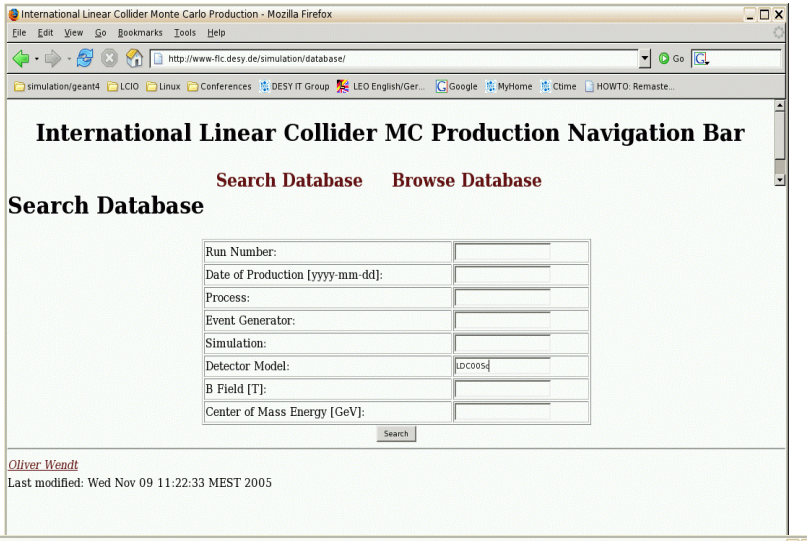


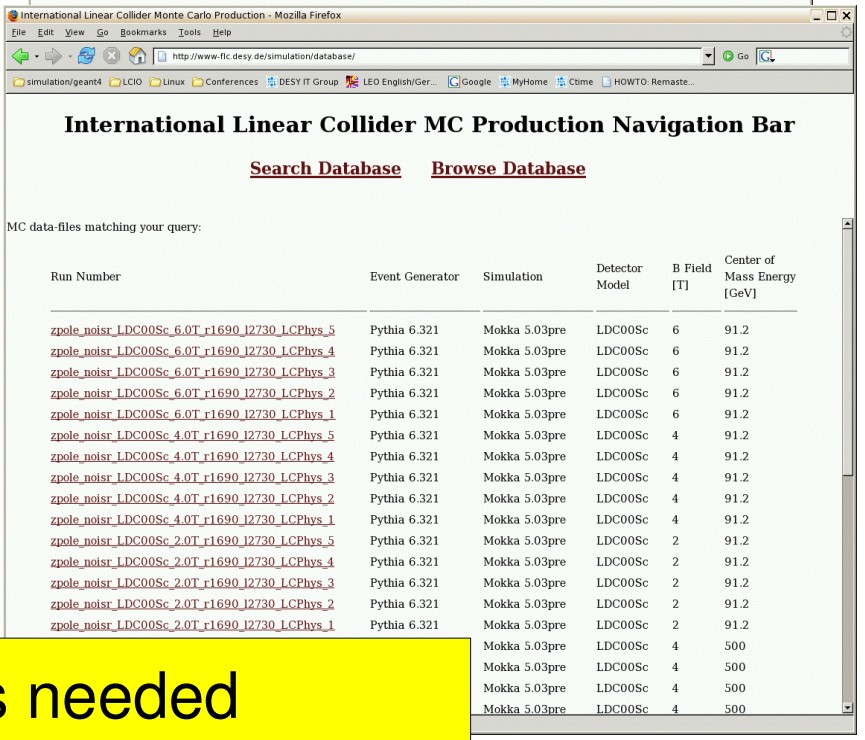
http://ilcsoft.desy.de

[aka: http://www-flc.desy.de/ilcsoft]

LDC simulation “mass production”

- detector optimization – vary
  - B,R_TPC,L_TPC,...
- need considerable number of events with detector parameter variations for benchmark reactions
- @ DESY:
  - started to produce these files on the **grid** for VO ILC
  - resolved some issues with Mokka and Geant4
  - will provide database with available data files
  - use **grid tools** to distribute the data !





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

Summary & Outlook

- a fairly complete OO-software framework exists for the LDC study based on Mokka, Marlin, LCIO, LCCD and GEAR
- can already be used for detector concept study !
  - -> started Monte Carlo production on the grid

details @ software portal:
http://ilcsoft.desy.de/

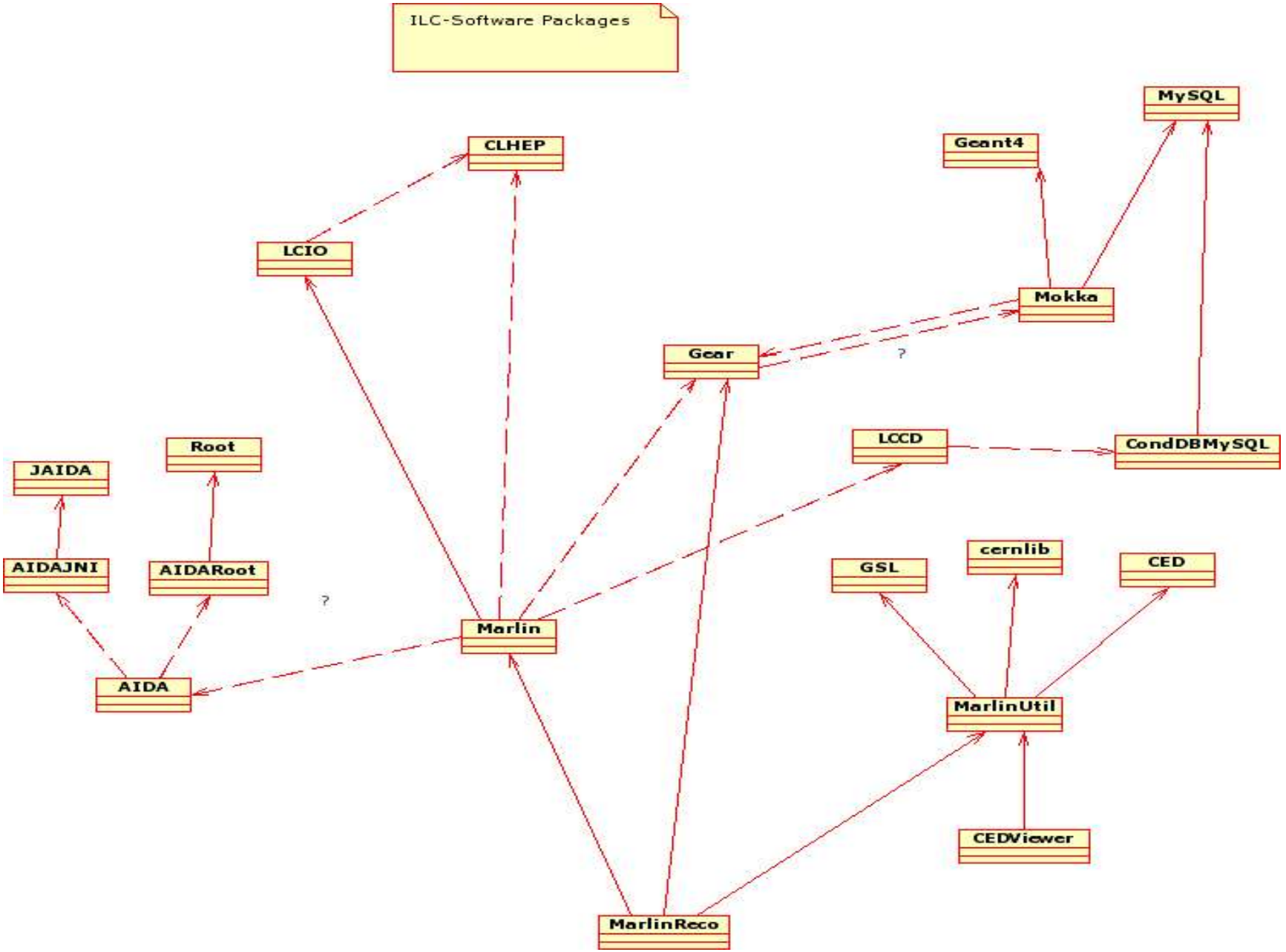
To Do:

- investigate interoperability with other frameworks (ongoing !)
- apply software to other concepts (ongoing !)
- combine with other software developments (Magic, DigiSim,...)
- improve software ...

Please consider using common software tools for your ILC study and provide feedback and contribute to the effort !

Backup slides ...

software package dependencies



Marlin – example XML steering

```
- <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>
```

- ActiveProcessors replaced by <execute>...</execute> section
- Reconstruct only events that pass the event selection

- Parameters defined as content of <parameter/> tag or as its value attribute

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

Marlin features

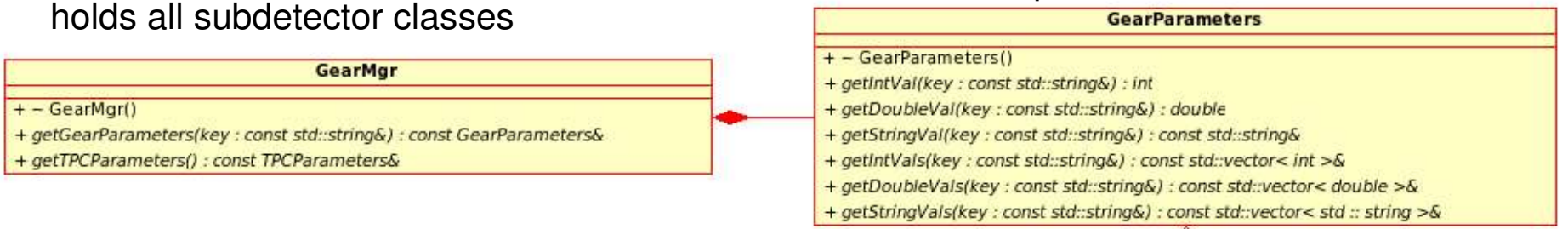
- core processors
  - AIDAProcessor
    - for easy creation of histograms, clouds, ntuples
  - OutputProcessor
    - writes current event or subset thereof
  - MyProcessor
    - simple example – serves as template for user code
  - ConditionsProcessor New
    - read conditions transparently with LCCD
  - DataSourceProcessor New
    - read non LCIO input, e.g. StdHepReader
  - SimpleFastMCProcessor New
    - fast smearing Monte Carlo
    - needs testing
- fully configurable through steering files:
  - program flow
  - input parameters
    - processor based and global
- self-documenting:
  - MyApplication -l

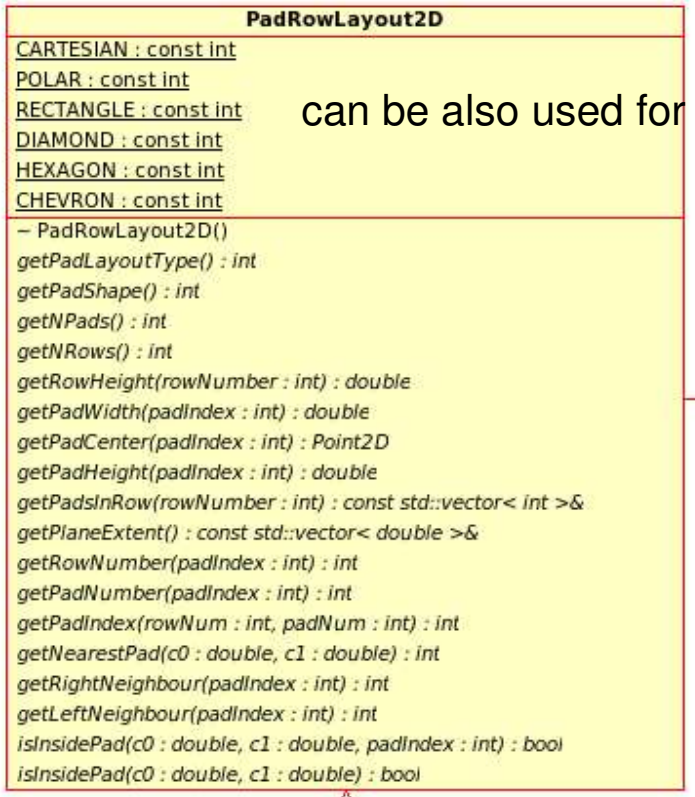
    will print all available processors with their parameters and example/default values

GEAR – TPC description

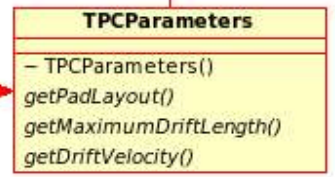
holds all subdetector classes

named parameters for additional attributes



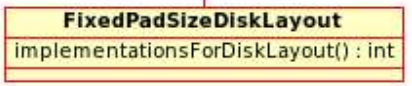


can be also used for FTD, CaloEndcap,...



TPC specific parameters

based on discussion with TPC experts at LCWS 2005



implementation for disk with pad rings

GEAR – material properties

GearDistanceProperties

- GearDistanceProperties()
getMaterialNames(p0 : const Point3D&, p1 : const Point3D&) : const std::vector< std :: string >&
getMaterialThicknesses(p0 : const Point3D&, p1 : const Point3D&) : const std::vector< double >&
getNRadlen(p0 : const Point3D&, p1 : const Point3D&) : double
getNIntlen(p0 : const Point3D&, p1 : const Point3D&) : double
getBdL(pos : const Point3D&) : double
getEdL(pos : const Point3D&) : double

integrated along path:
- straight line or
- true path in B-Field ?

GearPointProperties

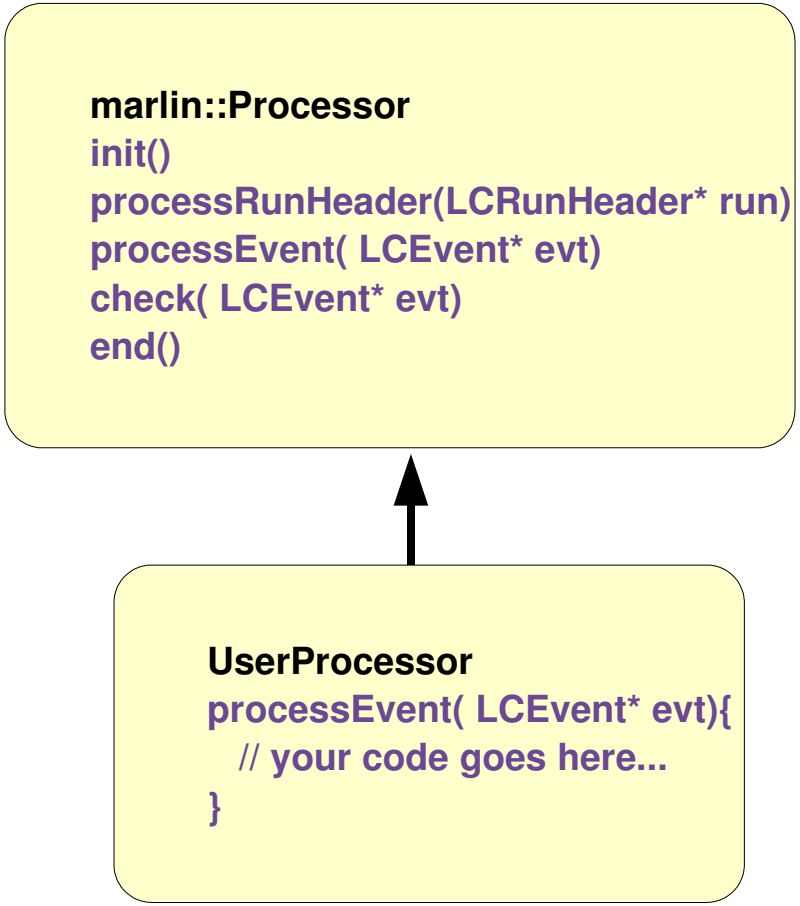
- GearPointProperties()
getCellID(pos : const Point3D&) : int
getMaterialName(pos : const Point3D&) : const std::string&
getDensity(pos : const Point3D&) : double
getTemperature(pos : const Point3D&) : double
getPressure(pos : const Point3D&) : double
getRadlen(pos : const Point3D&) : double
getIntlen(pos : const Point3D&) : double
getLocalPosition(pos : const Point3D&) : Point3D
getB(pos : const Point3D&) : double
getE(pos : const Point3D&) : double
getListOfLogicalVolumes(pos : const Point3D&) : std::vector< std :: string >
getListOfPhysicalVolumes(pos : const Point3D&) : std::vector< std :: string >
getRegion(pos : const Point3D&) : std::string
isTracker(pos : const Point3D&) : bool
isCalorimeter(pos : const Point3D&) : bool

properties at point from geant4 (CGA)

based on discussions at Argonne Simulation Meeting 2004

Marlin Processor

- provides main user callbacks
- has own set of input parameters
  - int, float, string (single and arrays)
  - parameter description
- naturally modularizes the application
- order of processors is defined via steering file:
  - easy to exchange one or several modules w/o recompiling
  - can run the same processor with different parameter set in one job
- processor task can be as simple as creating one histogram or as complex as track finding and fitting in the central tracker



LCIO status [v01-05]

- changed return values of E, P, m to double for MCParticle and ReconstructedParticle
  - stored are still floats
  - requires some trivial changes (float->double) in code where indicated by the compiler !
- new template UTIL::LCTypedVector<T> for creating std::vector<LCObjectType> from LCCollections
  - allows to use iterators and STL algorithms, e.g. std::for_each()
- added first implementations of generic tracker raw data classes (TPC, VTX, SiliconStrip,...)
  - TrackerRawData, TrackerData, TrackerPulse
- files are downward compatible with LCIO 1.4
- bug fixes
- see $LCIO/doc/versions.readme for more