Status of Core Software Tools

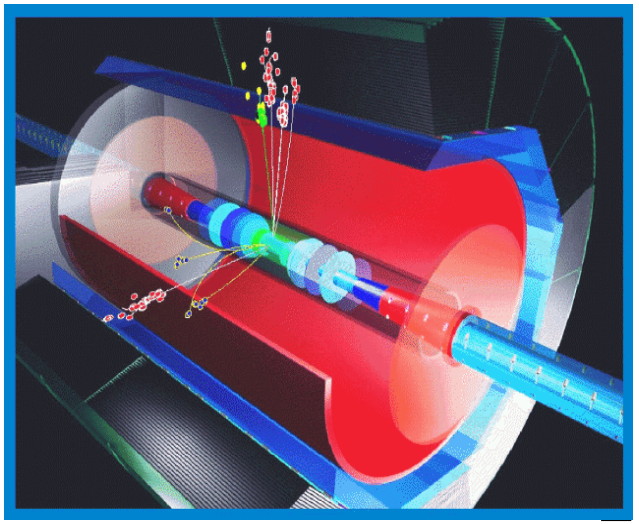
LDC Simulation and Reconstruction

Frank Gaede
DESY

ILC Software Workshop
Cambridge, April 4-6, 2006

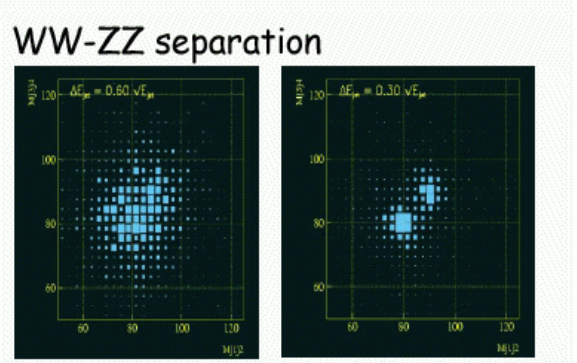
Outline

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



why full sim & rec @ ILC now ?

- precision tracking and vertexing
- high granularity in calorimeters
- need very high jet-mass resolution ~30%/sqrt(E/GeV):

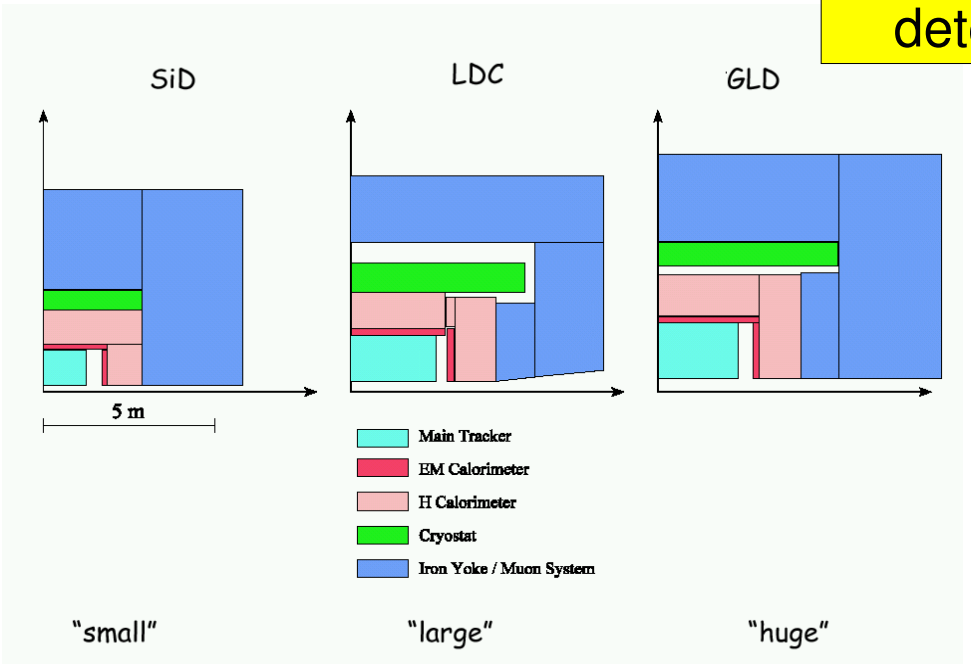


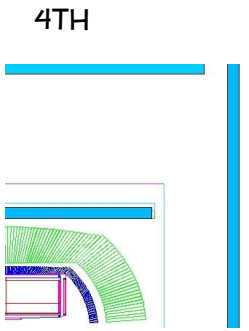
Particle Flow

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

why develop reconstruction sw now ?

- reconstruction algorithms (pflow) are part of the overall detector performance
- need full reconstruction to optimize detector concepts





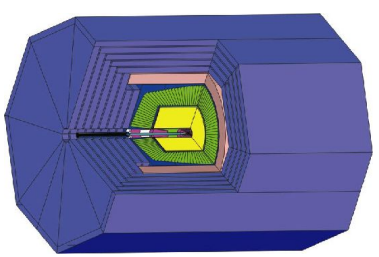
four interregional detector concept studies ongoing

ILC software packages

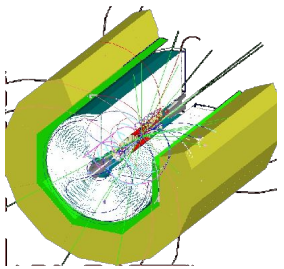
	Description	Detector	Language	IO-Format	Region
Simdet	fast Monte Carlo	TeslaTDR	Fortran	StdHep/LCIO	EU
SGV	fast Monte Carlo	simple Geometry, flexible	Fortran	None (LCIO)	EU
Lelaps	fast Monte Carlo	SiD, flexible	C++	SIO, LCIO	US
Mokka	full simulation – Geant4	TeslaTDR, LDC, flexible	C++	ASCI, LCIO	EU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

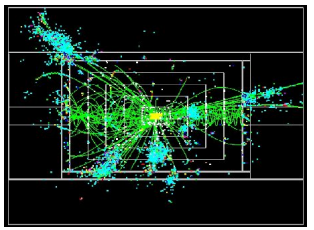
- *Geant4, StdHep and LCIO are common feature*
- *Each trying to be generic with different approach*
  different ways to define geometries



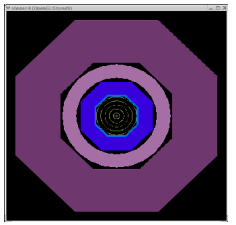
GLD,…



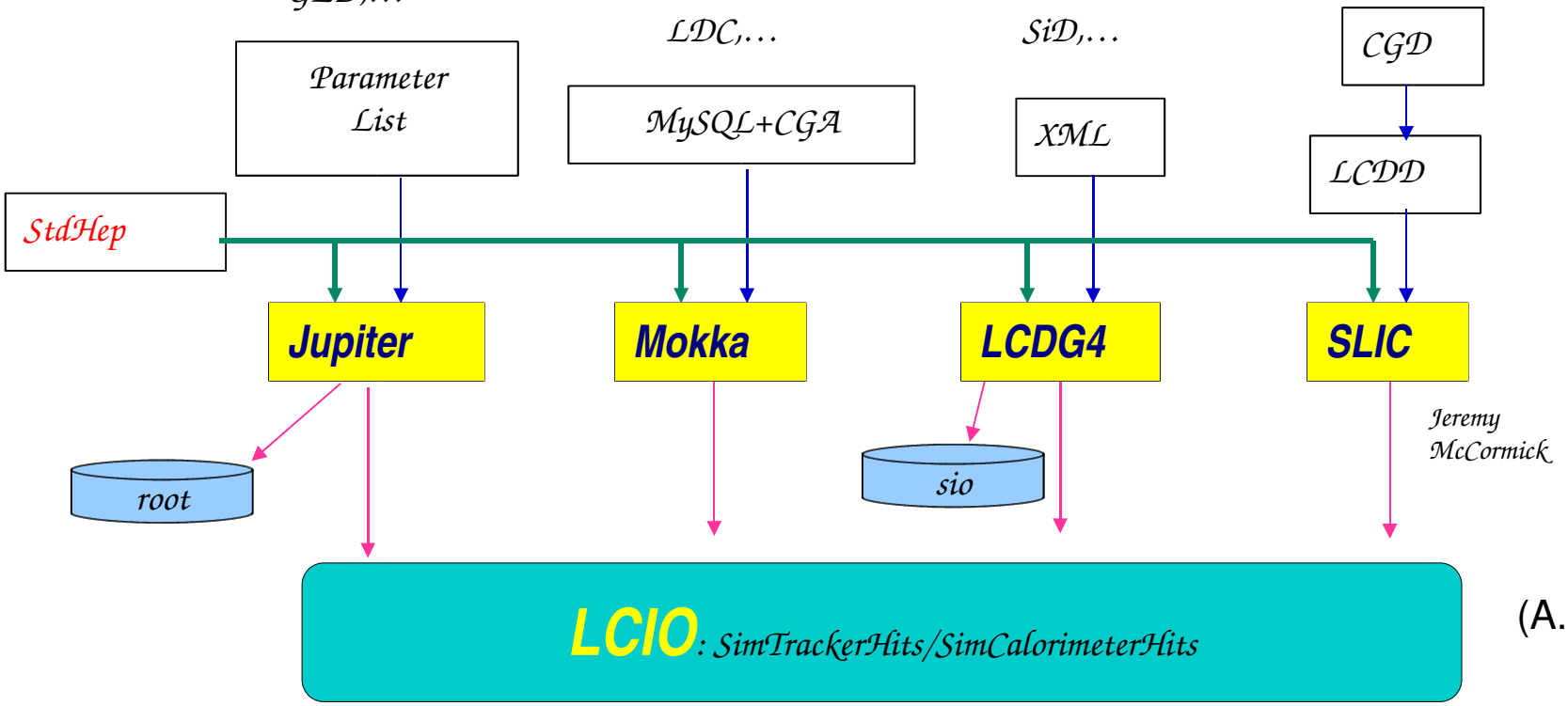
LDC,…



SiD,…

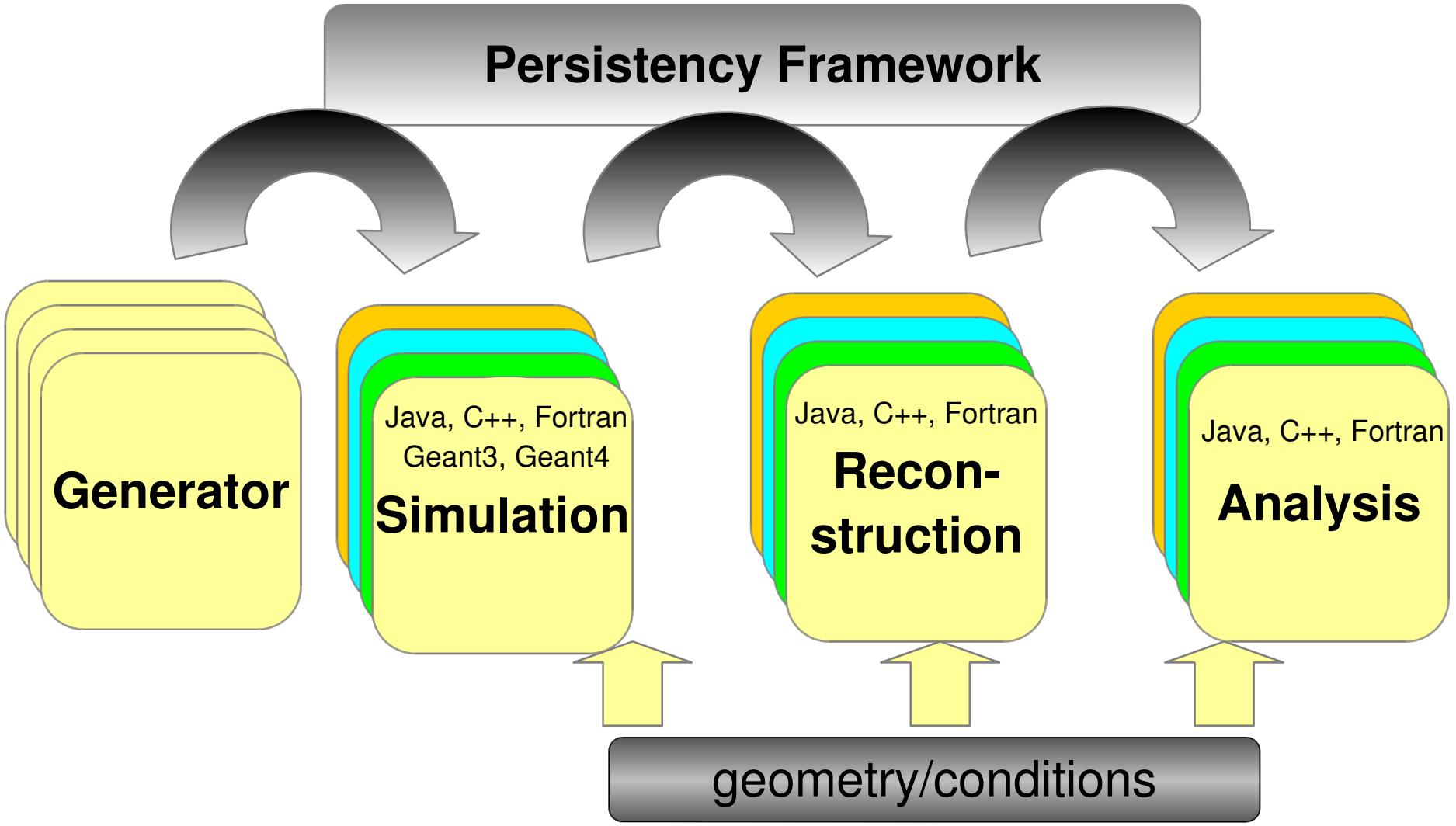


SiD,…

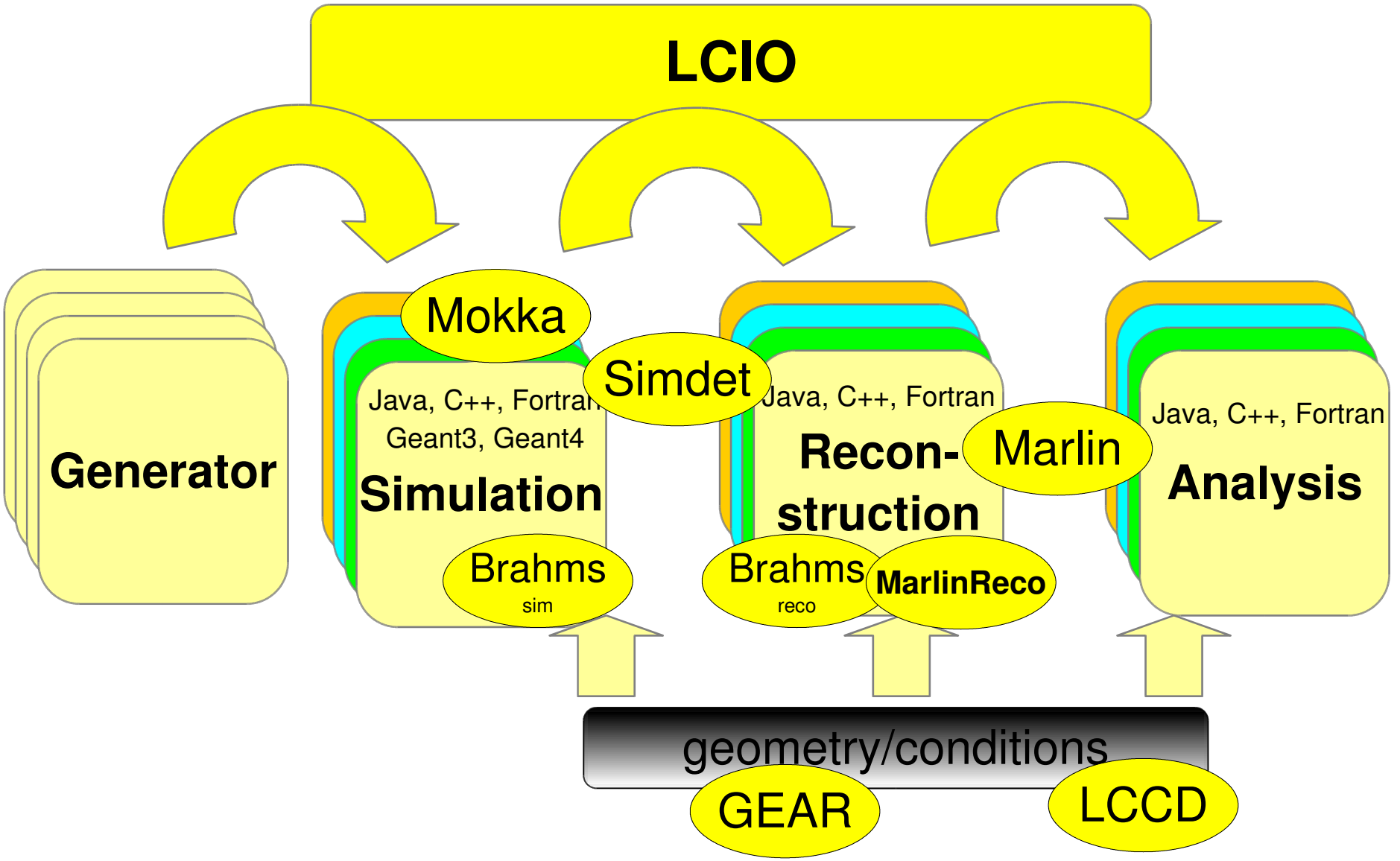


(A.Miyamoto)

ILC software chain



ILC software tools used for LDC



LCIO overview

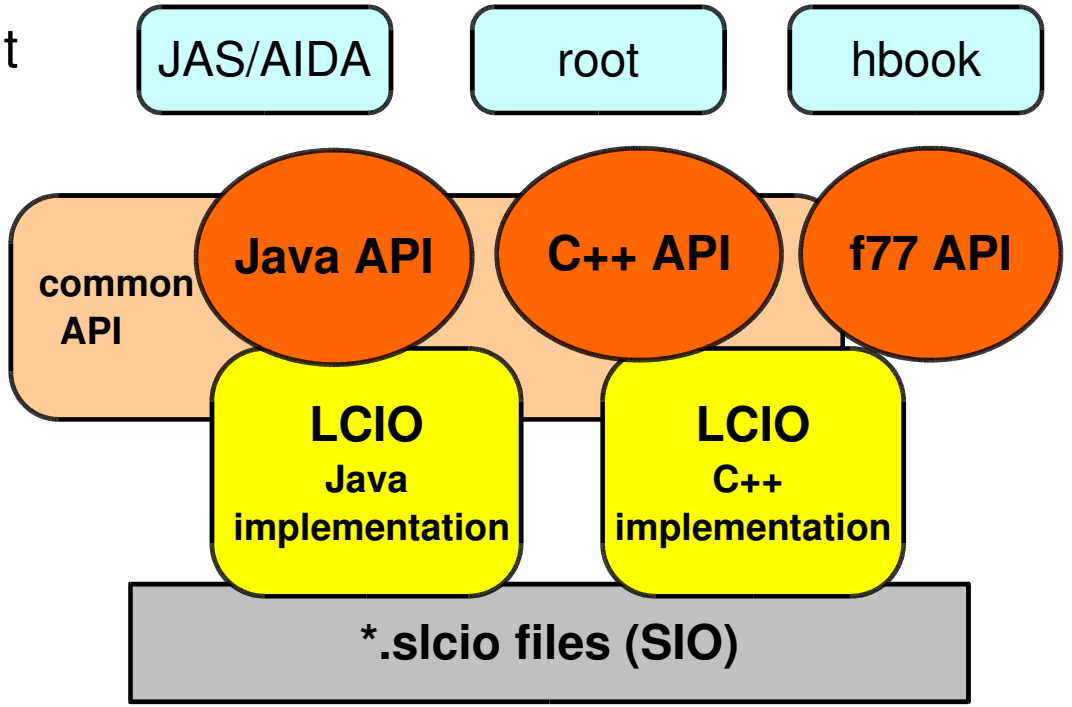
- DESY and SLAC joined project:
  - provide common basis for ILC software
- Features:
  - Java, C++ and f77 (!) API
  - extensible data model for current and future simulation and testbeam studies
  - user code separated from concrete data format
  - no dependency on other frameworks

simple & lightweight

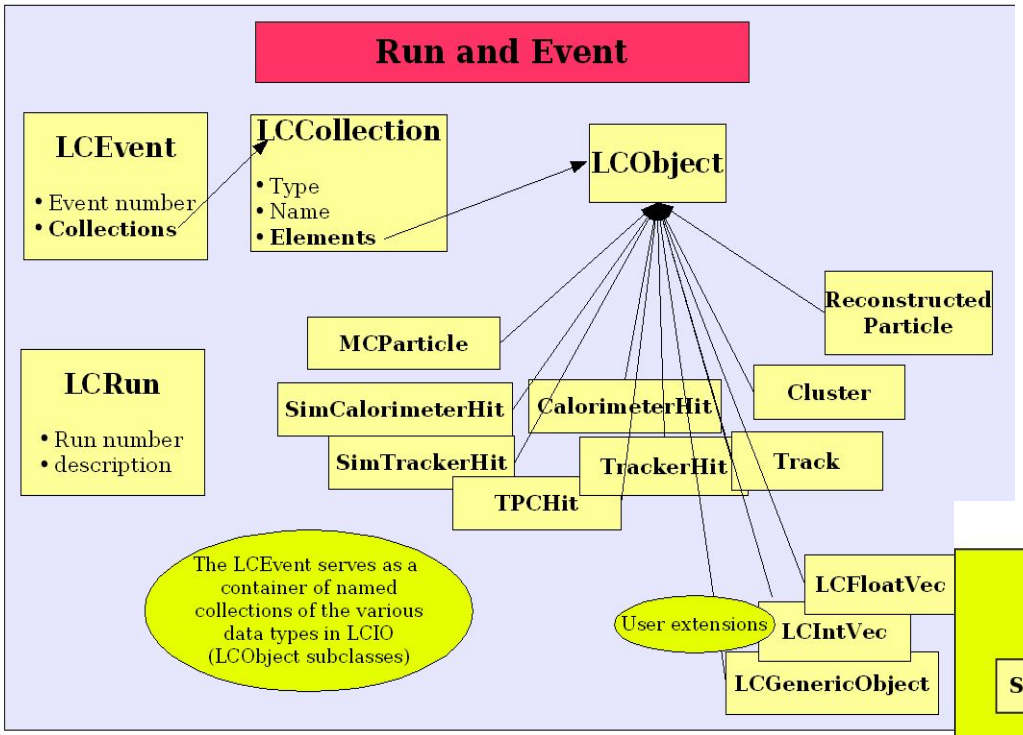
new release: **v01-06**

now de facto standard persistency & datamodel for ILC software

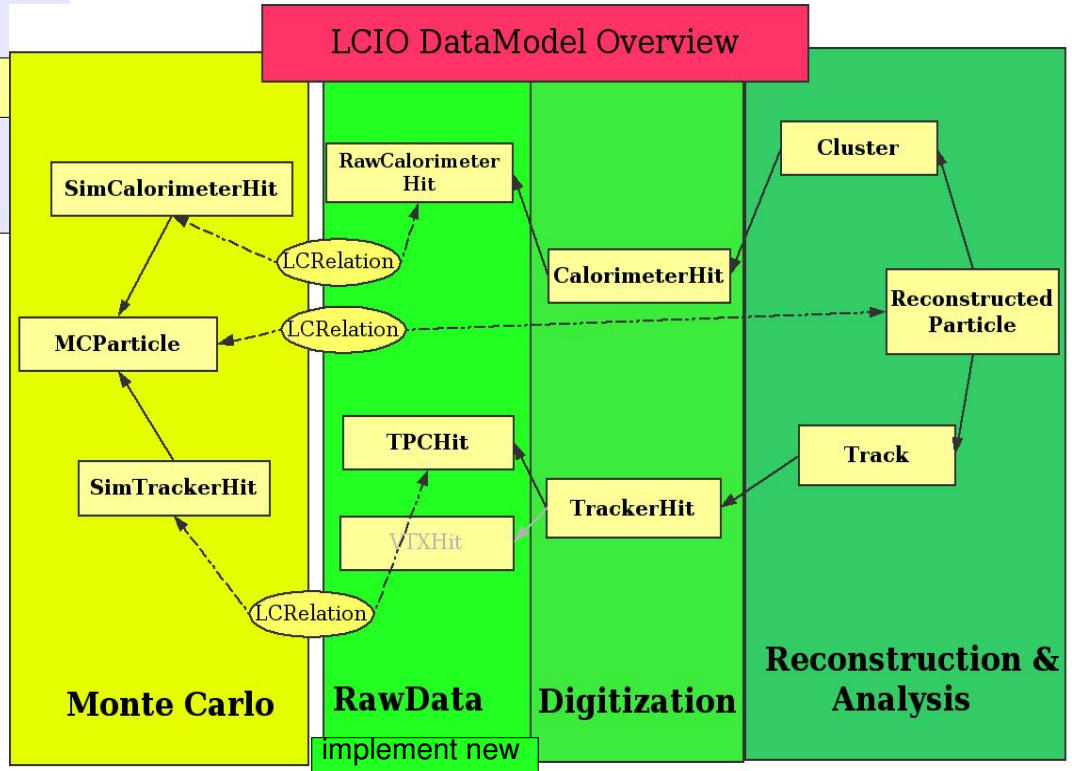
SW-Architecture



LCIO data model



event serves as container of untyped collections



hierarchy of data objects in the event

LCIO status [v01-07(06)]

- added optional momentum[3] and pathLength to SimTrackerHit (silicon digitization)
- LCReader::skipNEvents()
  - dump the n-th event in $LCIO/bin/dumpevent
- C++ utility code for encoding/decoding of cellids in hit classes (32bit and 64bit)
- support for large files (>2GB) and UTIL::LCSplitWriter
- optional python binding (J.McCormick)
  - files are downward compatible with LCIO 1.6
  - see $LCIO/doc/versions.readme for more

Simulation tools: Simdet, Brahms

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

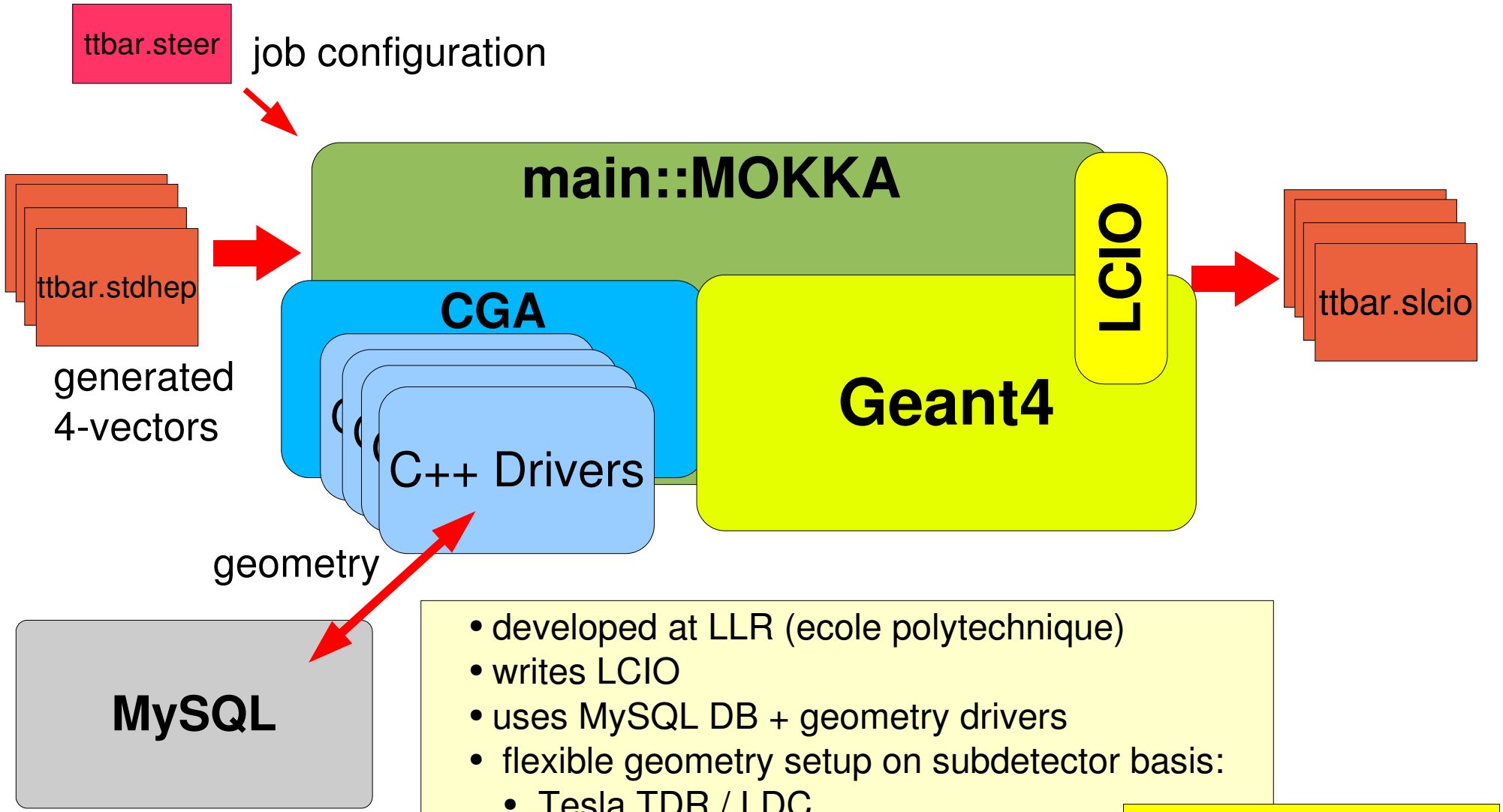
writes LCIO

- **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



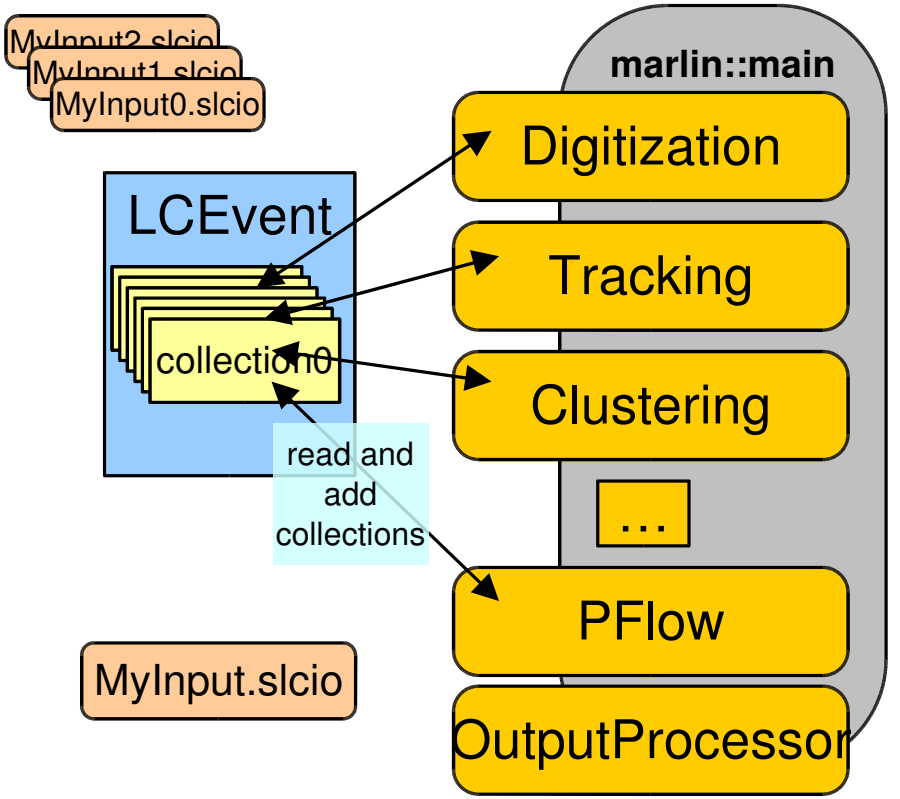
- developed at LLR (ecole polytechnique)
- writes LCIO
- uses MySQL DB + geometry drivers
- flexible geometry setup on subdetector basis:
  - Tesla TDR / LDC
  - SiD
  - Calice testbeam prototypes
- used in European/ LDC study

- see talk: G.Musat

Marlin

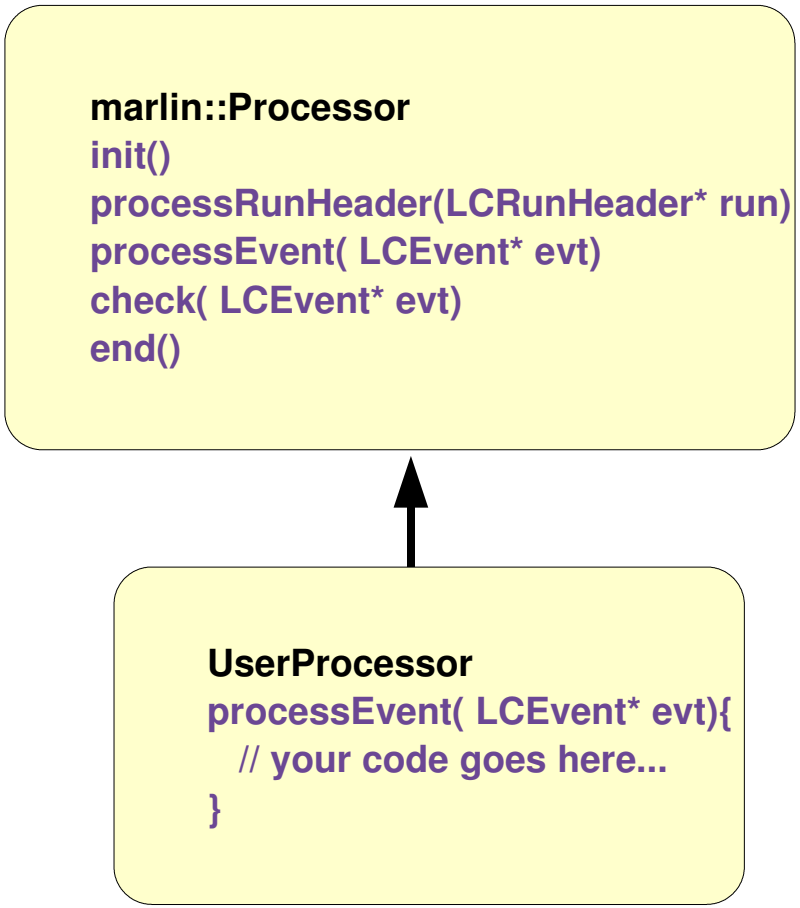
Modular**A**nalysis & **R**econstruction for the **L I N**ear Collider

- 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



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



Marlin core features

- fully configurable through steering files:
  - program flow
  - input parameters (processor based and global)
- self-documenting:
  - ./bin/Marlin -x
    prints example steering file with
    all available processors with their parameters and example/default values
- AIDA interface for histogramming
  - easy creation of histograms through absract 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) !!

Marlin Status and Plans

- current version: v00-09-04 (tagged last week):
- some new features:
  - made compatible with CLHEP 2.x
  - made compatible with RAIDA
  - split outputfiles wrt. size: ttbar_000.slcio, ttbar_001.slcio,...
  - bug fixes
- plans:
  - improve dealing with and creation of steering files
    - show parameters in order specfied
    - provide minimal steering files
    - provide help for one single processor
  - user suggestions/ needs ?

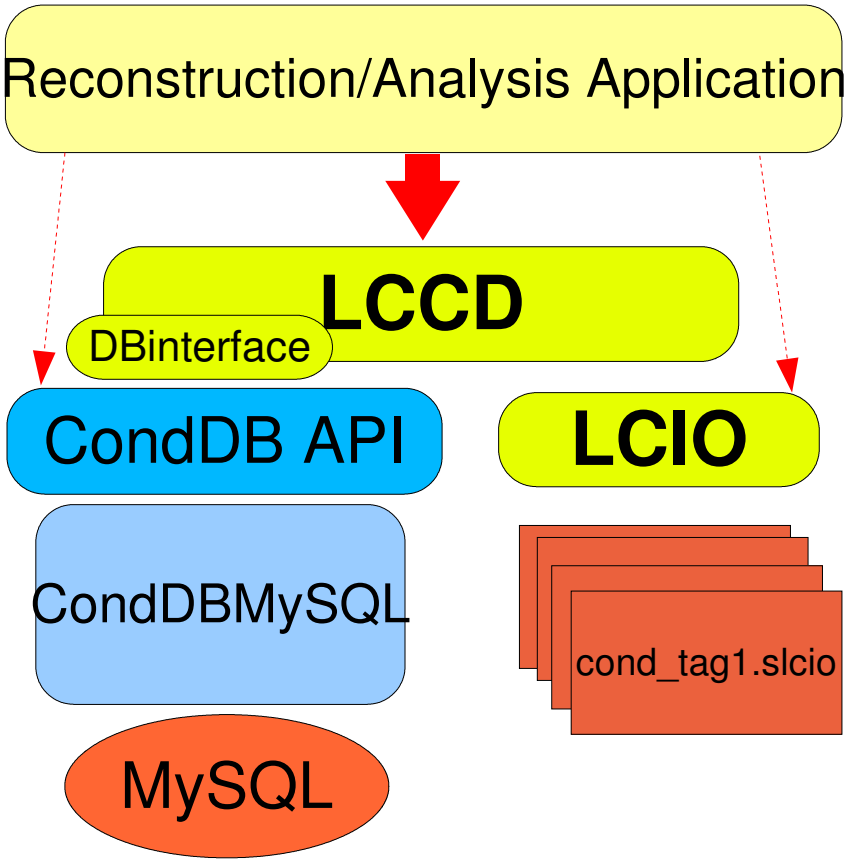
Marlin for reconstruction

- Marlin is a generic framework for processing of ILC data (LCIO)
  - simple and lightweight, depends on LCIO only
- some ingredients for typical reconstruction are missing:
  - description of detector geometry
  - conditions data (testbeam)
  - utility software, math libraries, ...
- optionally Marlin supports additional tools:
  - **GEAR** – geometry description
  - **LCCD** – conditions data
  - MarlinUtil – utility library

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

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>
```

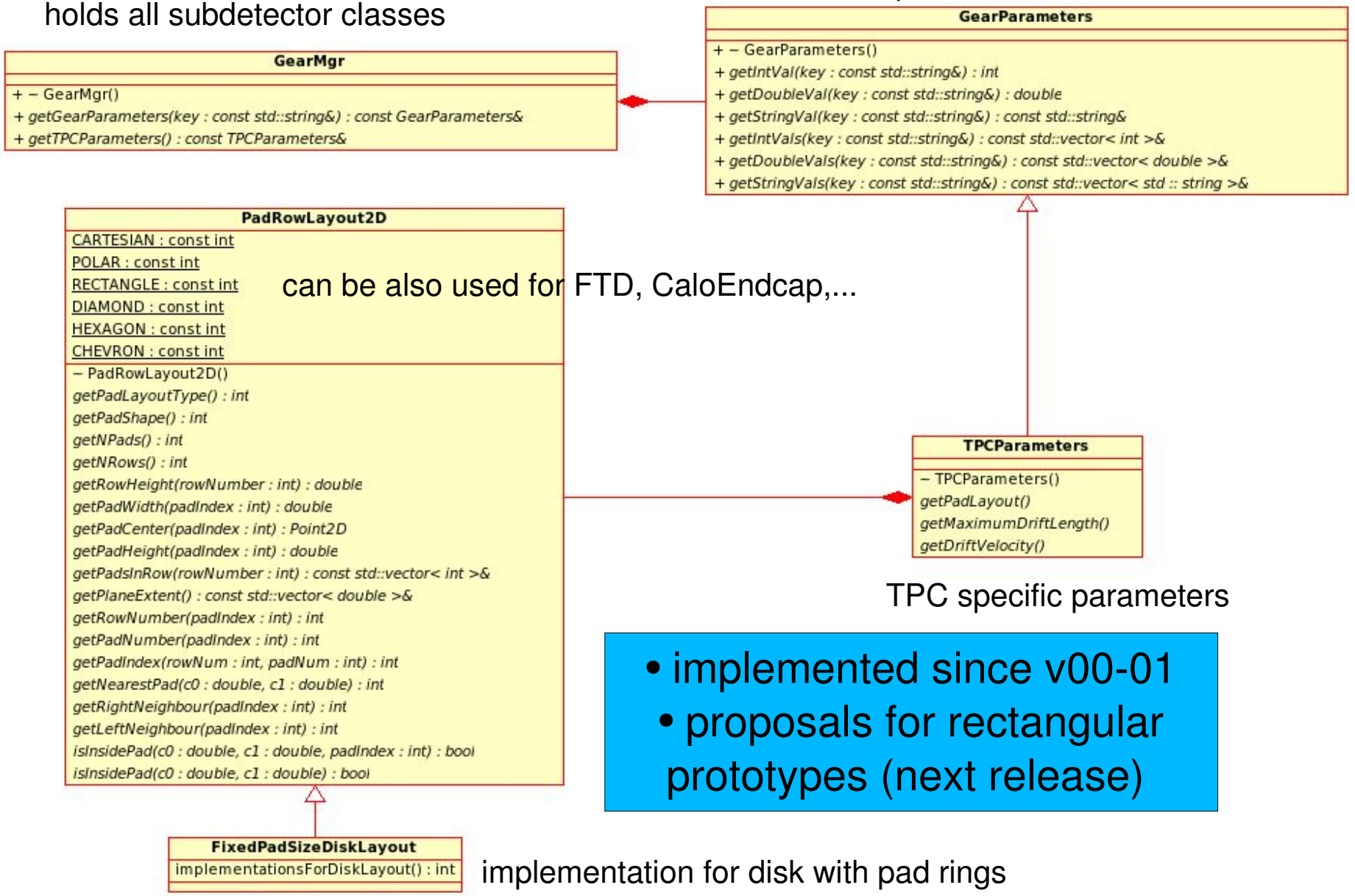
compatible with US – compact format

- 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

GEAR example:TPC description

holds all subdetector classes

named parameters for additional attributes



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

TPC specific parameters

- implemented since v00-01
- proposals for rectangular prototypes (next release)

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

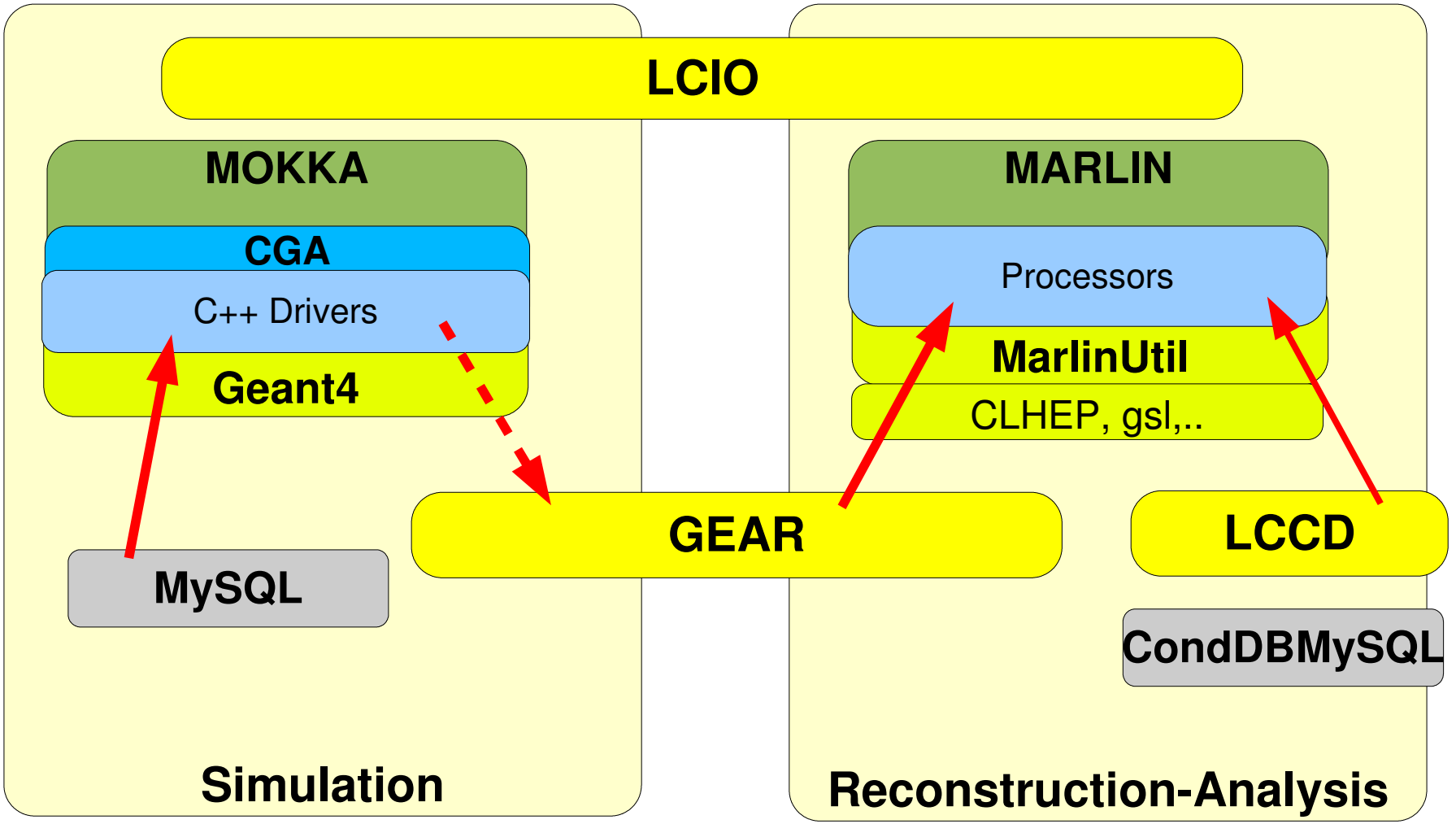
properties at point from geant4 (CGA)

- proposal since Argonne Simulation Meeting 2004
- no implementation yet
- could use Mokka CGA
- importance ?

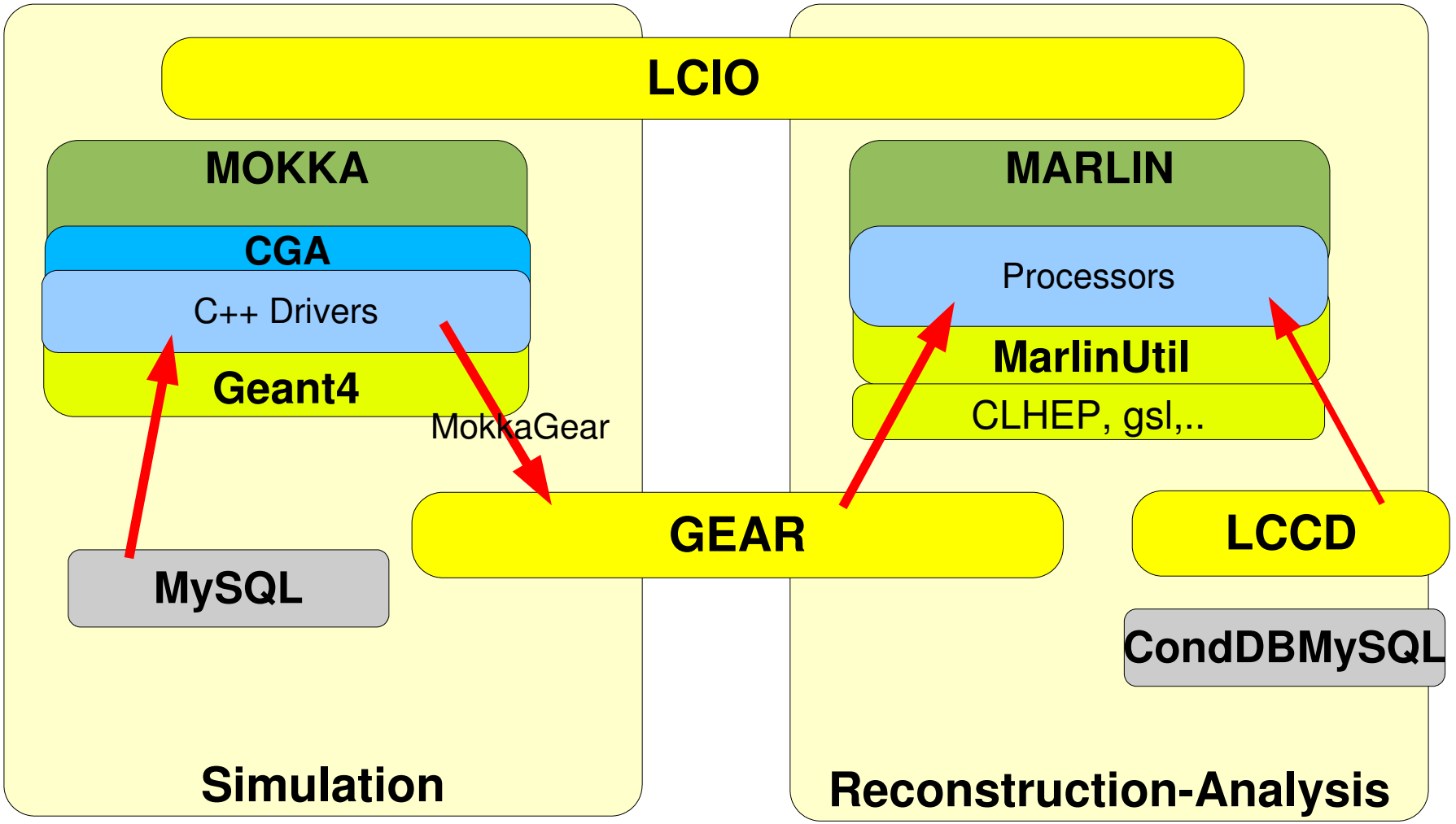
Gear status

- Gear proposed at DESY simws 2005
- first version v00-01 for snowmass
- current version v00-02 (since vienna 2005):
  - TPC, Hcal, Ecal interfaces defined and implemented
  - user parameters
- plans for next release (soon):
  - also write xml files from parameters in memory
  - tool to merge files: gearmerge

LDC simulation framework



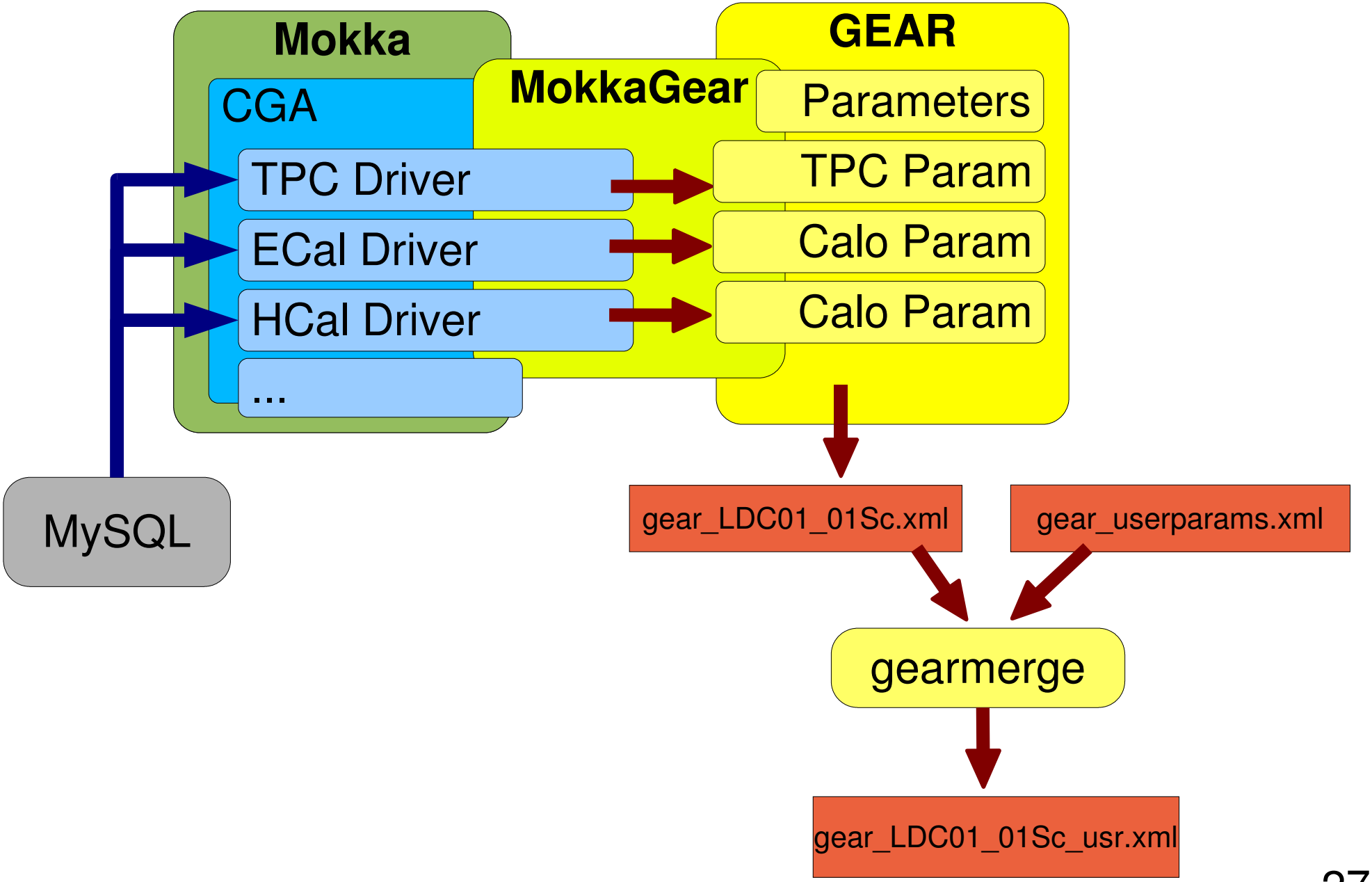
LDC simulation framework



MokkaGear

- extension to Mokka (R.Lippe – Diploma Thesis)
- extract geometry information in drivers when detector is built
- use Gear to create XML files for reconstruction
- currently implemented:
  - TPC (tpc04), Ecal (ecal02) and Hcal (hcal04)
- to be released soon (next Mokka release)
- optional feature
  - only if Gear is installed and included

MokkaGear


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

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, PandoraPFA,...

MarlinReco packages

- **TrackDigi**
  - TPCDigi
  - new VTXDigi
- **CaloDigi**
  - LDCCaloDigi
- **Tracking**
  - LEPTracking
  - new VTXTracking
  - TrackCheater

talks: S.Aplin O.Wendt

talk: A.Raspereza

- **Clustering**
  - TrackwiseClustering
  - ClusterCheater
- **Pflow**
  - Wolf

talk: P.Krstonosic

- **Analysis**
  - EventShapes
  - SatoruJetFinder

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

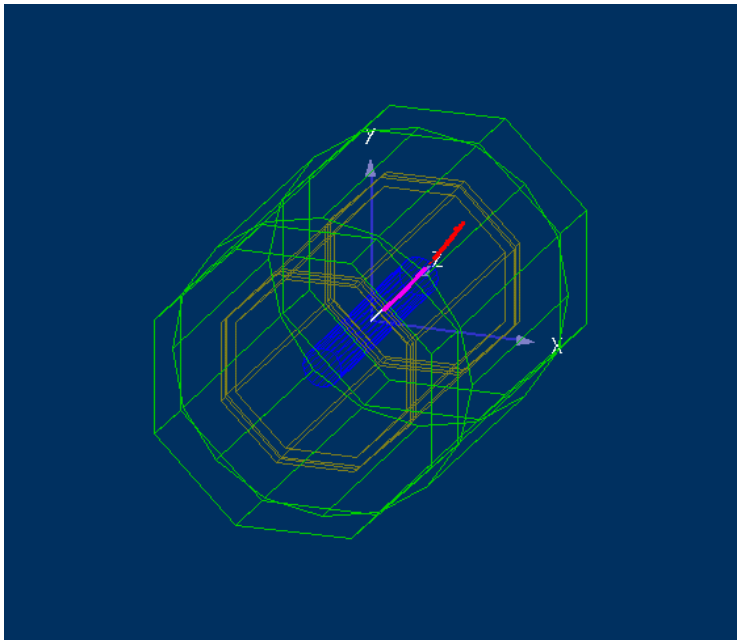
MarlinReco support packages

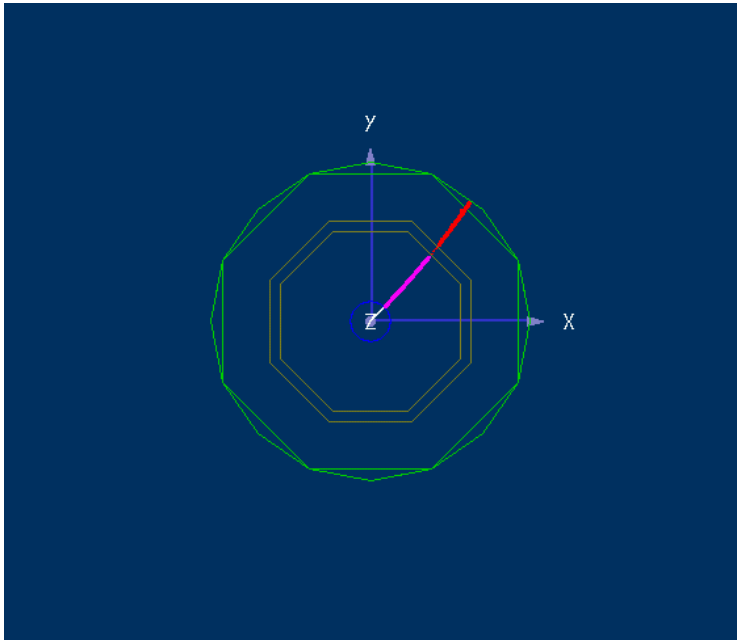
- MarlinUtil (O. Wendt)
  - Utility and Helper classes
    - helix fitter, cluster shapes,...
    - common code for CED
- RAIDA

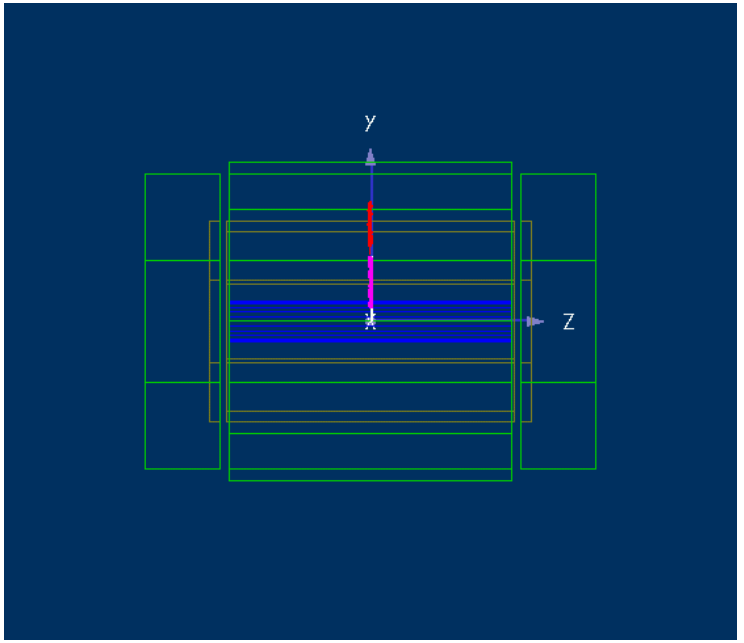
  talk: T.Kraemer
  - AIDA root implementation
- CED (A. Zhelezov)
  - event display based on GLUT/ OpenGL
  - client server architecture
- CEDViewer
  - event display client processors
    - CEDViewer. GenericViewer

CEDViewer

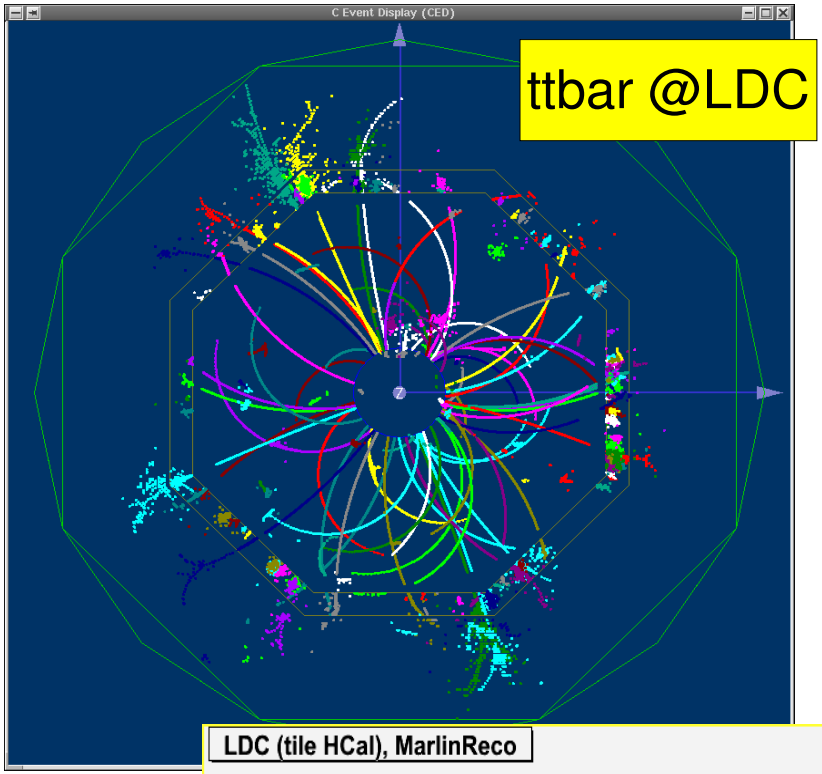
simple example taken from
$MarlinReco/examples/LDC
steer_ldc.xml
gear_ldc.xml



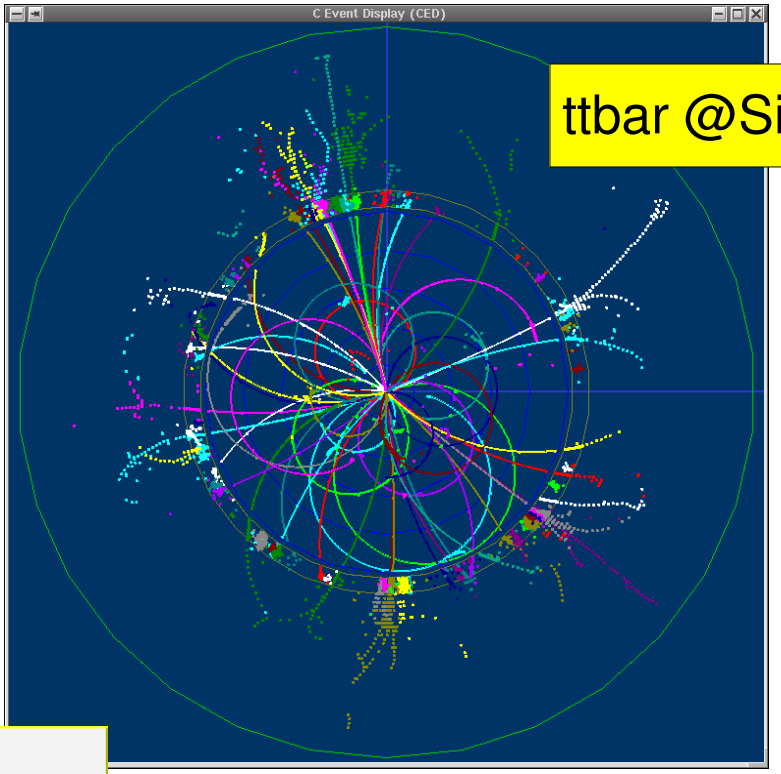




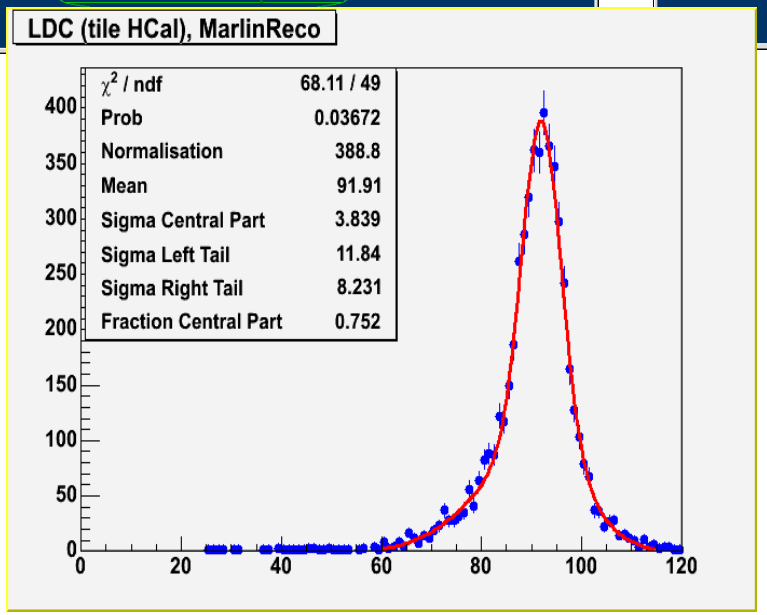
MarlinReco @work



ttbar @LDC

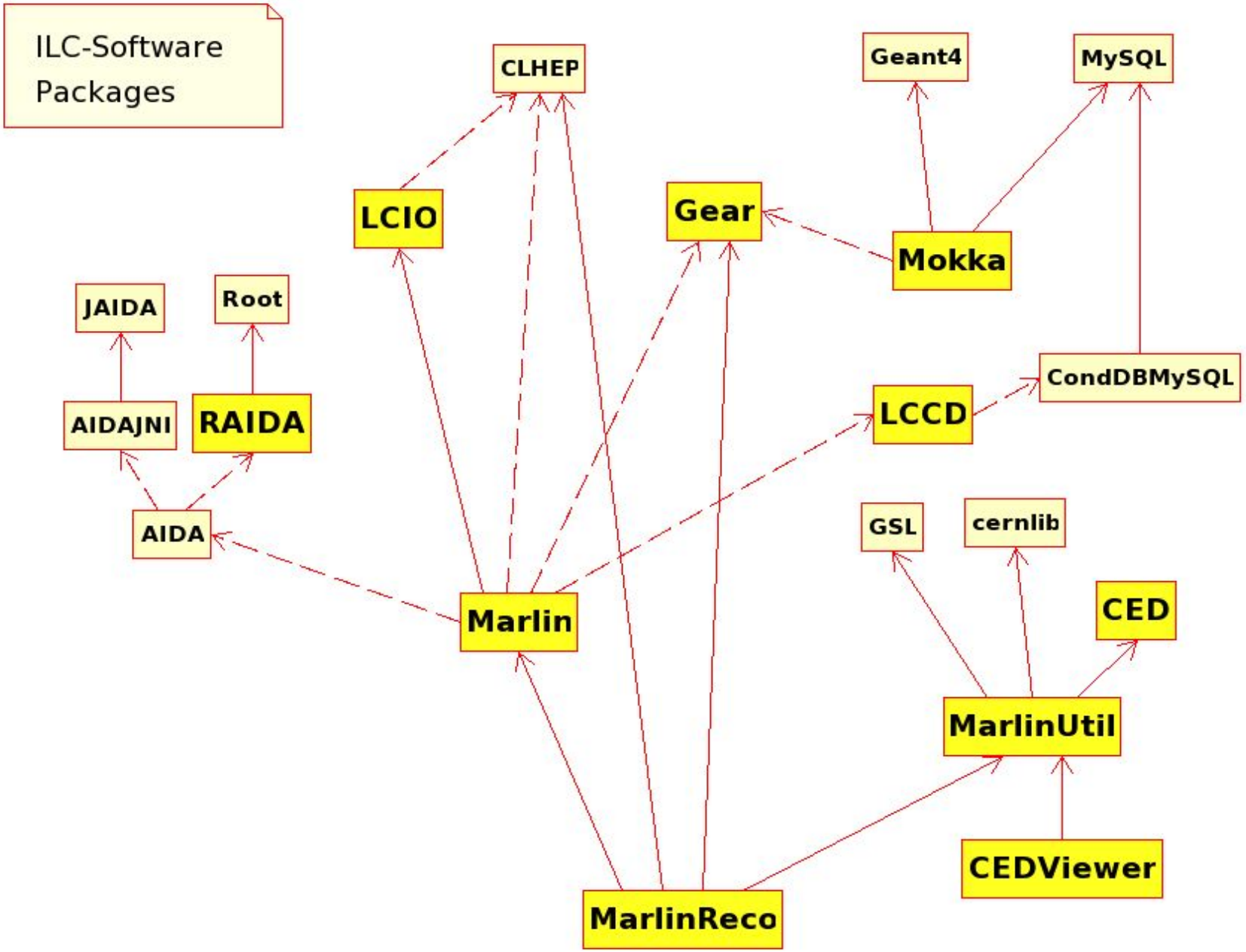


ttbar @SiD

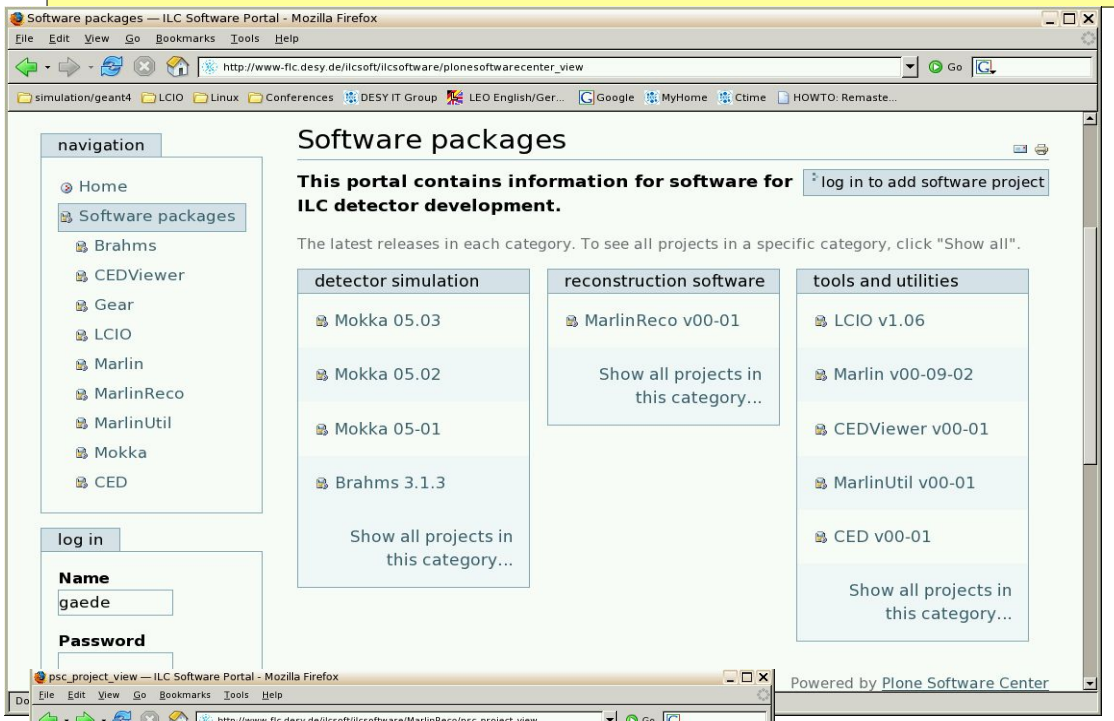


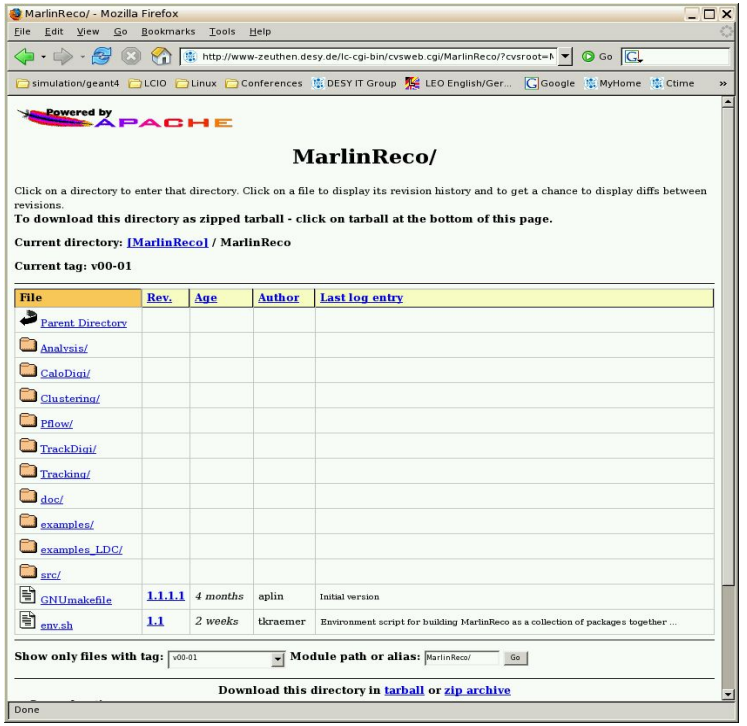
see talks:
- S.Aplin
- A.Raspereza
- P.Krstonosic
- O.Wendt

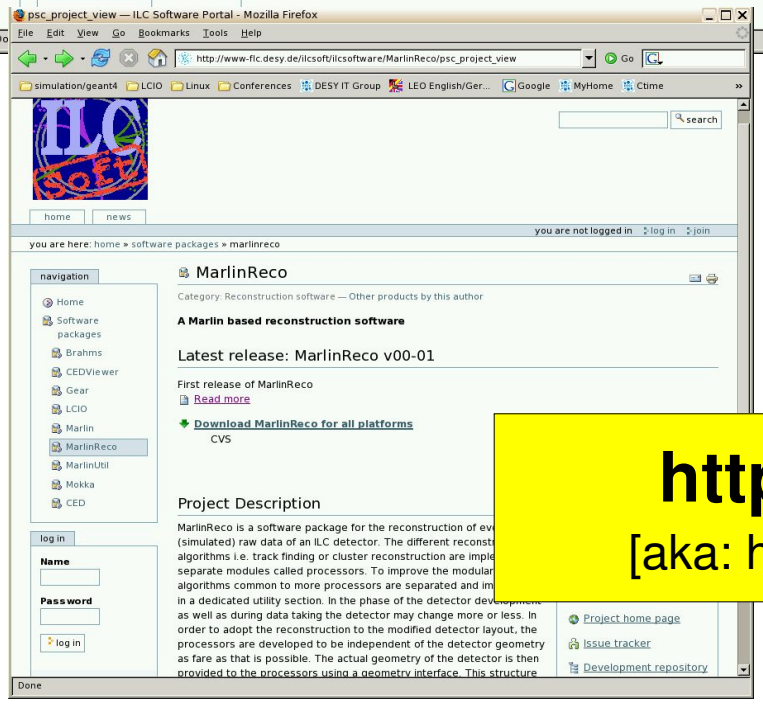
software package dependencies

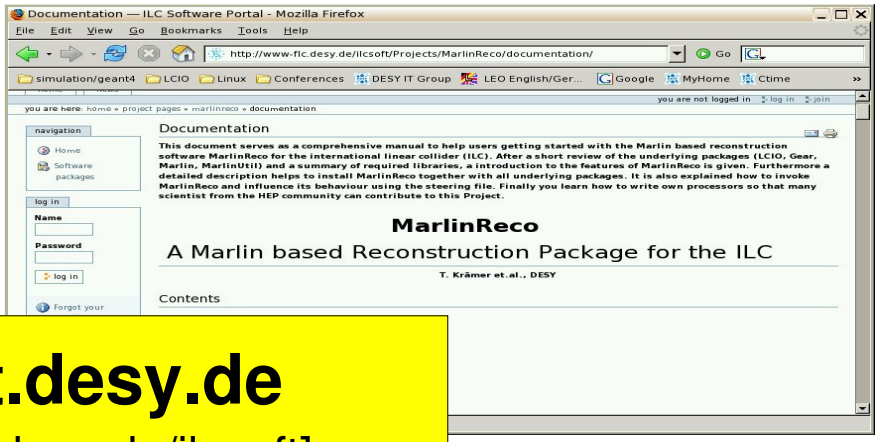


ILC software portal









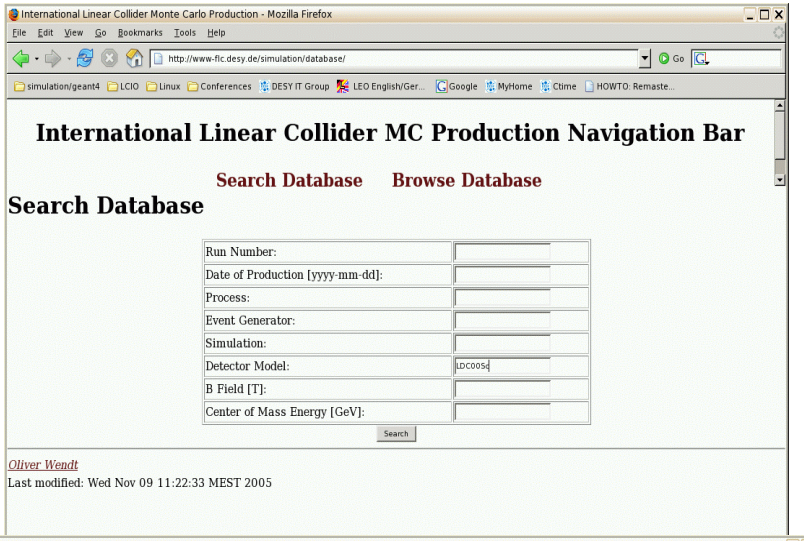
http://ilcsoft.desy.de

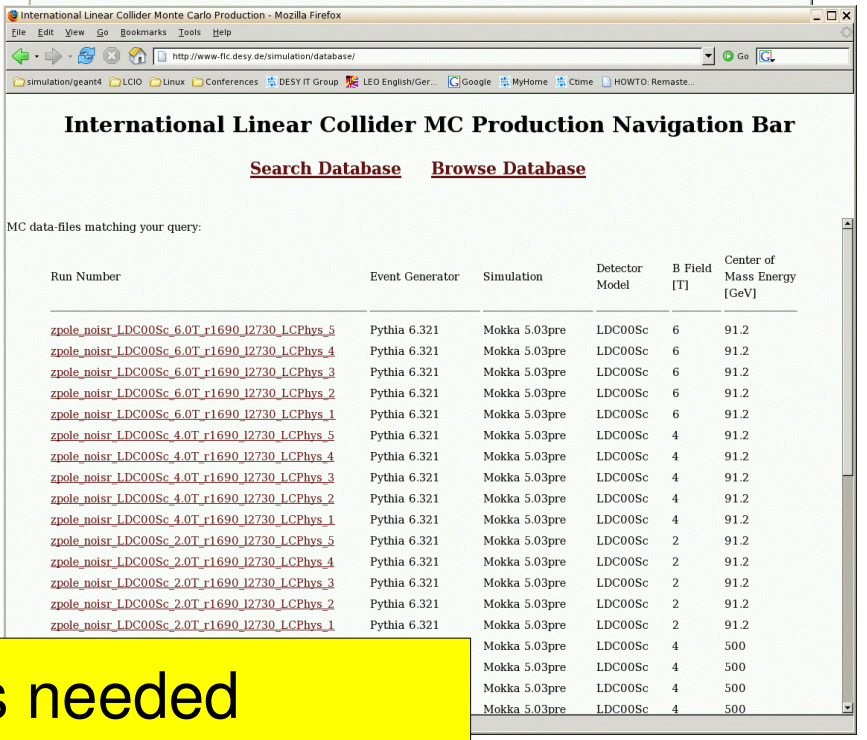
[aka: http://www-flc.desy.de/ilcsoft]

LDC simulation “mass production”

- detector optimization – vary
  - B,R_TPC,L_TPC,...
- need considerable number of events with detector parameter variations for benchmark reactions
- 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 !

talk: D.Martsch





- 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
  - used for detector concept study !
  - used Monte Carlo production on the grid

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

To Do:

- investigate interoperability with other frameworks
- improve software ...
- development needs to be driven by user (reconstruction) needs

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

Backup slides ...

MarlinReco Digitization

- Tracker (S.Aplin)
  - TPC: Gaussian smearing of position in z and r-phi
  - Silicon: exact position from simulation –
    - more meaningful clustering under development (A.Raspereza)
- Calorimeter (A.Raspereza)
  - calibration only - no smearing
  - optional energy cut
  - ganging (investigate different granularity)

MarlinReco Tracking

- Central Tracks: (S.Aplin)
  - algorithms taken from LEP (ALEPH and DELPHI)
  - f77 code from Brahms
    - track finding based on out-in search using circle fit
    - fit with Kalman Filter takes material into account
    - start with TPC tracks and then include VTX hits (next slide)

- Track Cheater (A.Raspereza)
  - uses Monte Carlo for track finding
  - fitted with a helix hypothesis

MarlinReco Clustering

- Trackwise Clustering (A.Raspereza)
  - algorithm needs spatial information only
    - -> applicable to both digital and analogue calorimeters
  - minimal dependence on detector geometry
    - -> can be used for other detector concepts

- Cluster Cheater (A.Raspereza)
  - uses Monte Carlo to combine hits into clusters
  - proximity criterion for 'realistic' clusters

PFlow

- Track-Cluster matching (A.Raspereza)
  - extrapolate tracks into the calorimeter
    - use only outer track hits and apply helix fit
    - proximity criterion to assign cluster to track
  - charged objects (E,p) from track
  - neutral particles (E,p) from cluster
- Particle ID (A.Raspereza)
  - cluster shape analysis
    - fraction of energy in ECAL
    - longitudinal and transverse profile
    - test of MIP hypothesis

Analysis

- Event Shapes (T. Kraemer, P. Krstonic)
  - ThrustReconstruction: Tasso & Jetnet algorithms
  - Sphere: sphericity, aplanarity, ...
- SatoruJetFinder (J. Samson)
  - originaly developed by Satoru Yamashita for OPAL
- BenchmarkPlots
  - not yet -> input needed