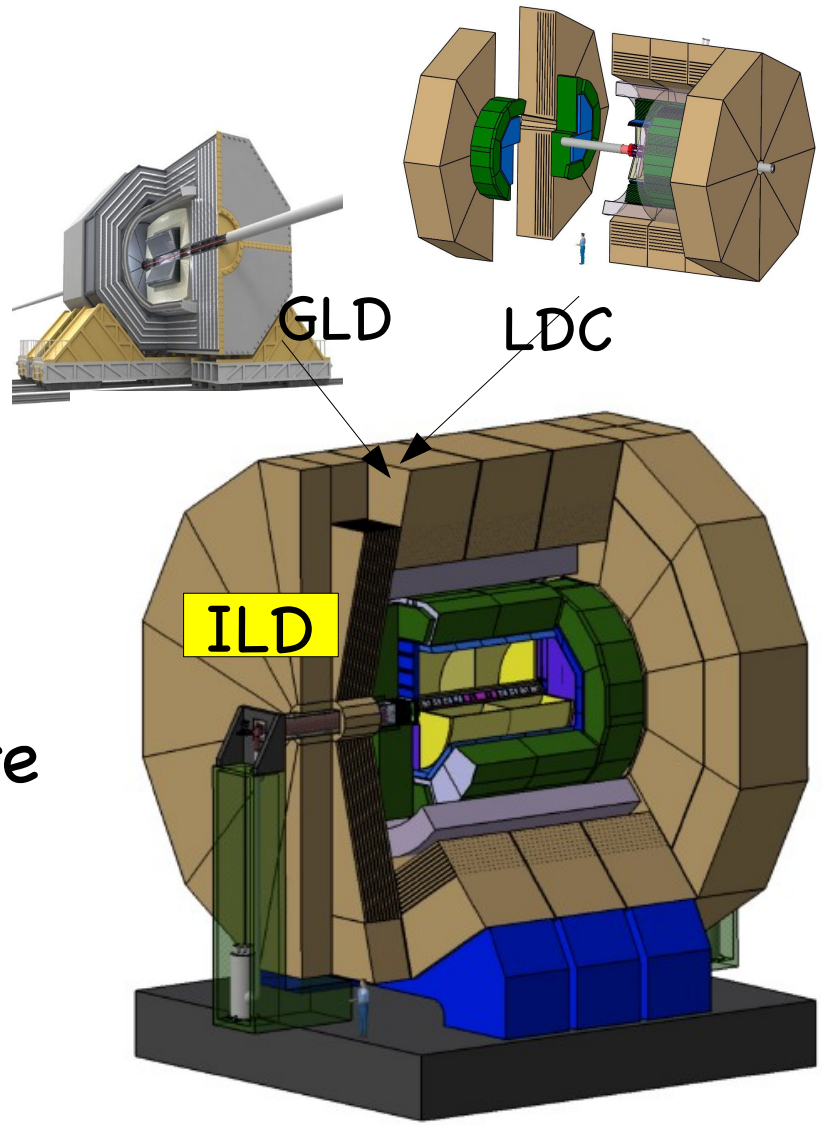
The ILD software framework - LDC flavor

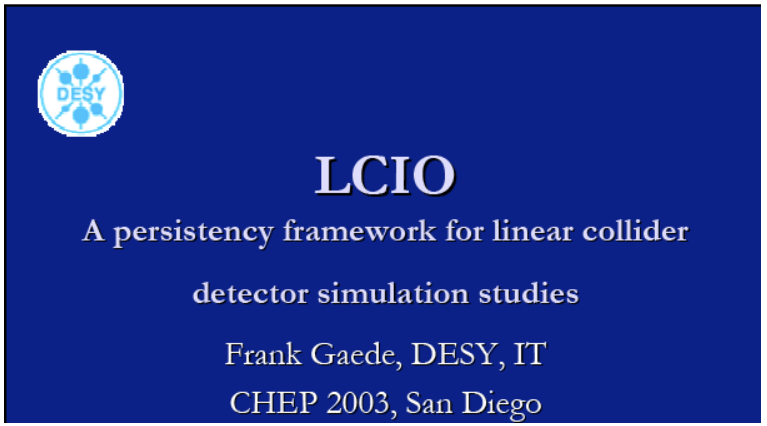
Frank Gaede
DESY
ILD Software Workshop
KEK, Tsukuba, 16.04.2009

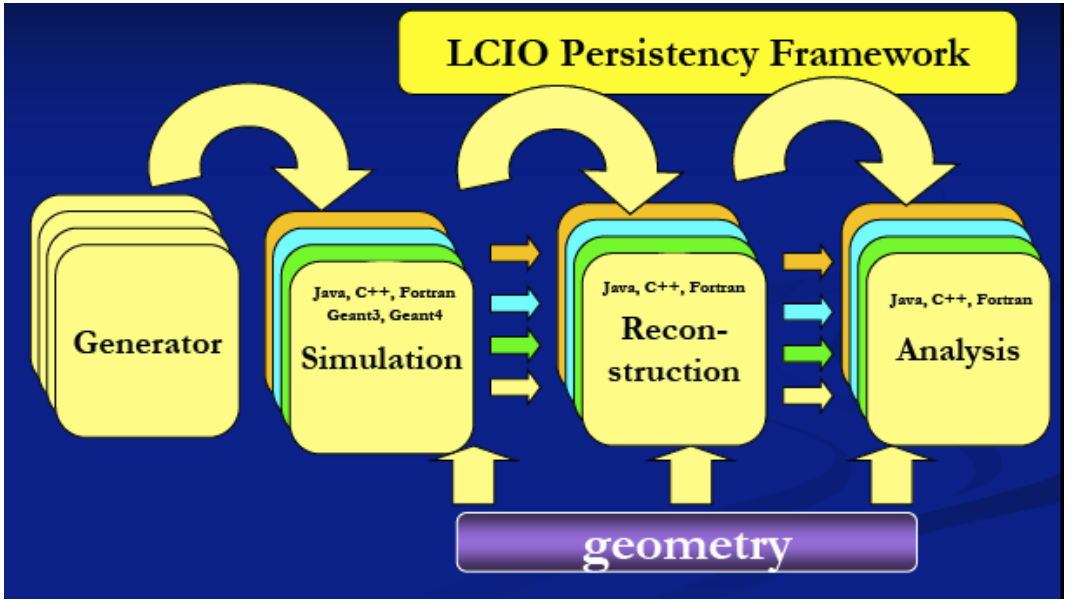
Outline

- Introduction
  - or a brief history of ILC software
- The framework and its users
- Possible improvements - the developers view
- Towards a common ILD software framework
- Summary/Outlook

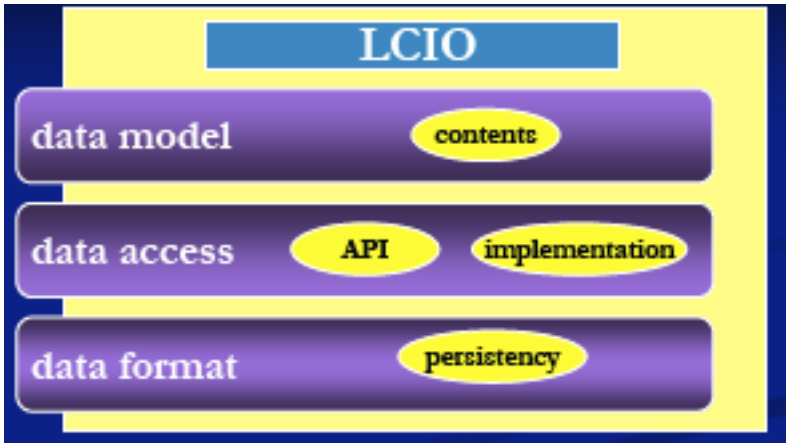


LCIO introduced at CHEP- 2003



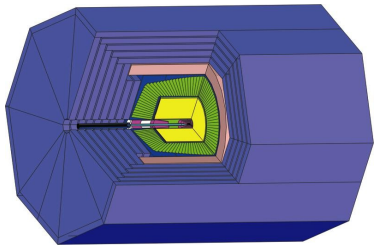


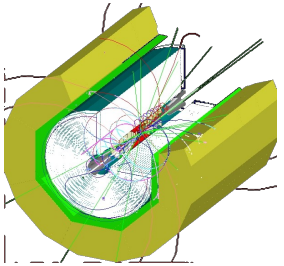
- idea for LCIO from ECFA – persistency task force
- started as DESY-SLAC common project in 2002 to
- provide data model and persistency to international ILC detector R&D community
- provide basis for collaboration and safe manpower

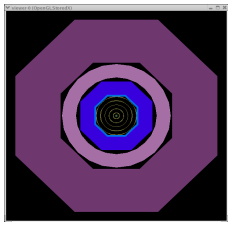


ILC Simulation Frameworks - 2005

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







GLD,...

LDC,...

SiD,...

A.Miyamoto @LCWS 2005

Parameter List

MySQL+CGA

CGD

LCDD

StdHep

Jupiter

Mokka

SLIC

root

LCIO: SimTrackerHits/SimCalorimeterHits

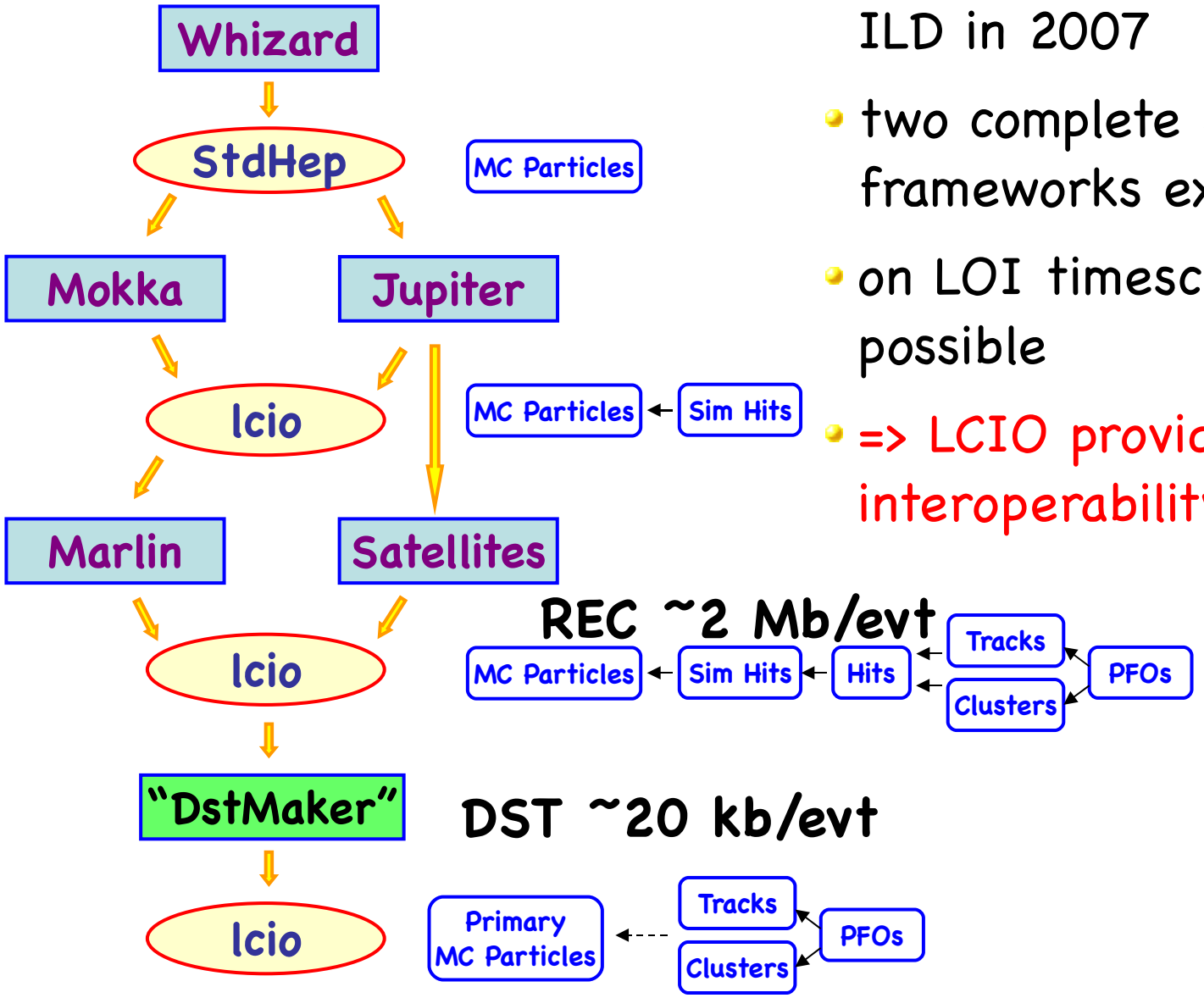
Note: 4th did not yet exist - and has not adopted LCIO since

ILC tools/frameworks (LCWS07)

	Description	Detector	Language	IO-Format	users
Simdet	fast Monte Carlo	TeslaTDR	Fortran	**LCIO**	LDC,SID
SGV	fast Monte Carlo	simple Geometry, flexible	Fortran	None/LCIO	LDC,SiLC
LiCToy	fast Monte Carlo	simple trk. Geometry	C++	**LCIO**	LDC,SiLC
Lelaps	fast Monte Carlo	SiD, flexible	C++	SIO, **LCIO**	SID
Mokka	full simulation – Geant4	LDC, flexible	C++	**LCIO**	LDC
SLIC	full simulation – Geant4	SiD, flexible	C++	**LCIO**	SID
Jupiter	full simulation – Geant4	GLD	C++	Root/**LCIO**	GLD
ILCroot	full sim. – Geant4/Flukka/g3	4th	C++	Root	4th
Marlin	reconstruction and analysis application framework	Flexible	C++	**LCIO**	LDC
org.lcsim	reconstruction framework	SiD (flexible)	Java	**LCIO**	SID
Jupiter-Satelites	reconstruction and analysis	GLD	C++	Root	GLD
ILCroot	reconstruction and analysis	4th	C++	Root	4th
LCCD	Conditions Data Toolkit	All	C++	**LCIO**	LDC,Calice,...
GEAR	Geometry description	Flexible	C++	XML	LDC,Calice,...
LCIO	Persistency and datamodel	All	Java, C++, Fortran	**LCIO**	LDC,SID, GLD,Calice,...
JAS3/WIRED	Analysis Tool / Event Display	All	Java	xml,stdhep, eprep,LCIO,.	SID
root	Analysis Tool / Event Display	All	C++	Root	LDC,GLD,4th

LCIO: basis for 'horizontal' collaboration

ILD frameworks interoperability 2008



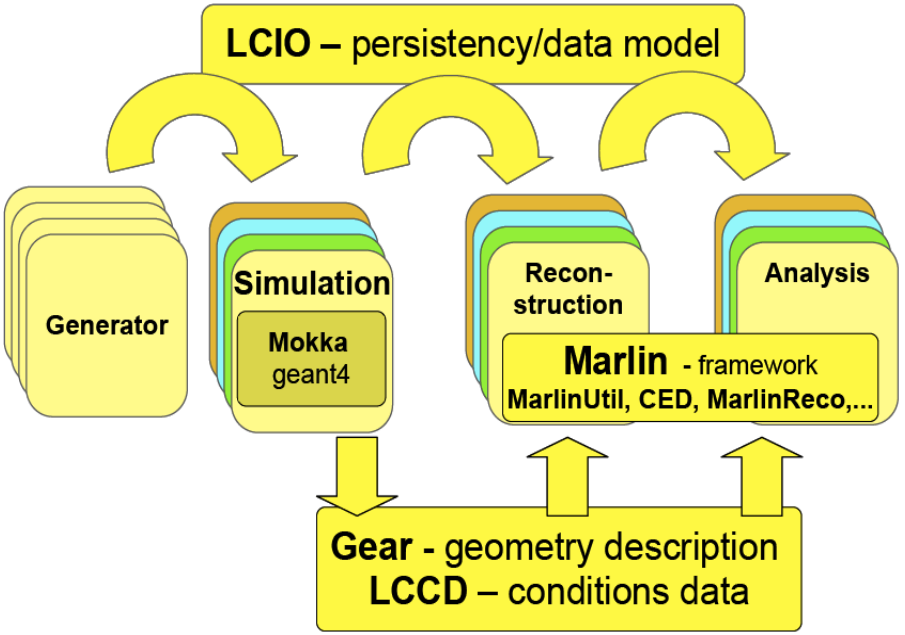
- merger of GLD and LDC into ILD in 2007
- two complete and independent frameworks existed
- on LOI timescale no 'unification' possible
- => LCIO provided the needed interoperability

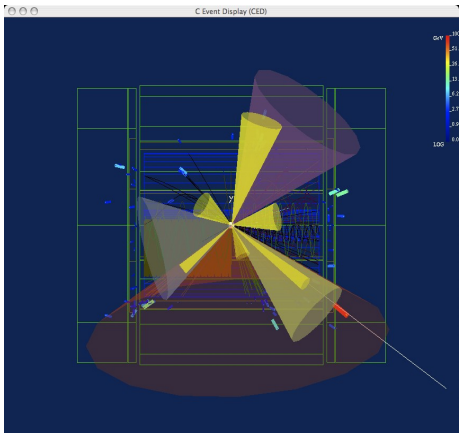
Common ILD framework 2009-...

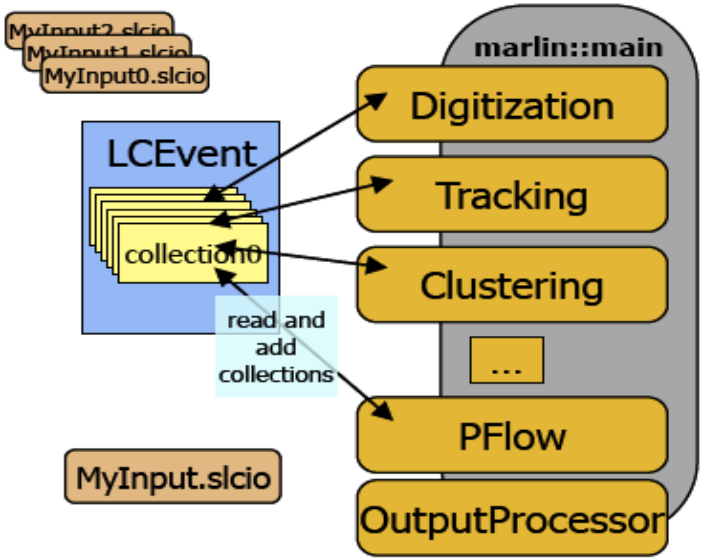
- ILD software working group (A.Miyamoto, F.G.) charged to work towards a unified ILD software framework with joined responsibilities
- LCIO is a good starting point for this as already used in both worlds
- this workshop:
  - see where we are after the LOI
  - get user feedback -> Requirements for software
  - develop ideas on how to continue
  - - within ILD
  - - in collaboration with other concepts/groups

The ILD software framework - LDC flavor

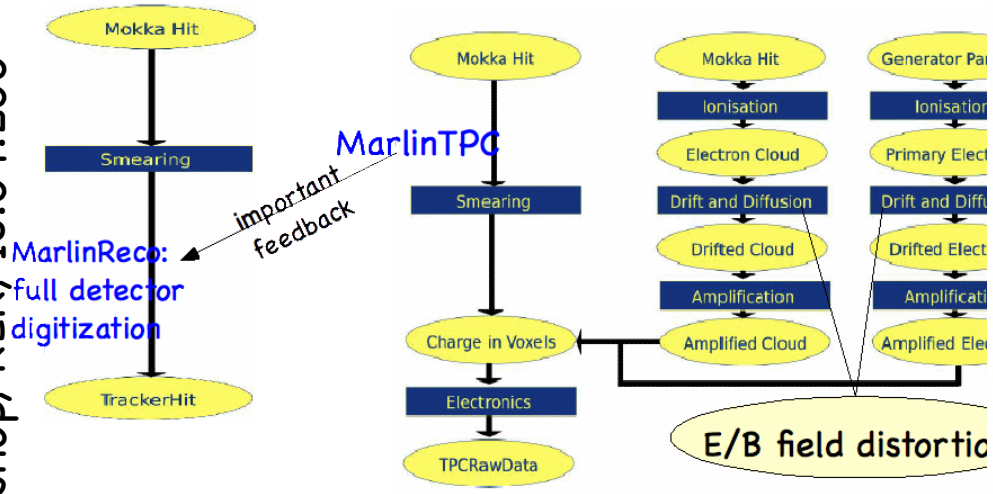
- **Mokka** (LLR)
  - geant4 simulation application
- **LCIO** (DESY/SLAC)
  - international standard for persistency format / event data model
- **Marlin**
  - core application framework for reconstruction & data analysis
- **GEAR**
  - geometry package f. reconstruction
- **LCCD**
  - conditions
  - data toolkit (DB)
- **CED**
  - 3d event display

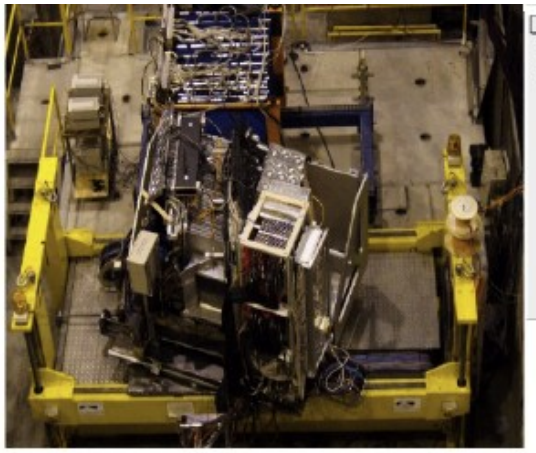


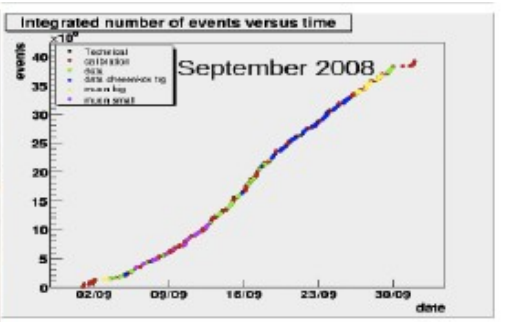




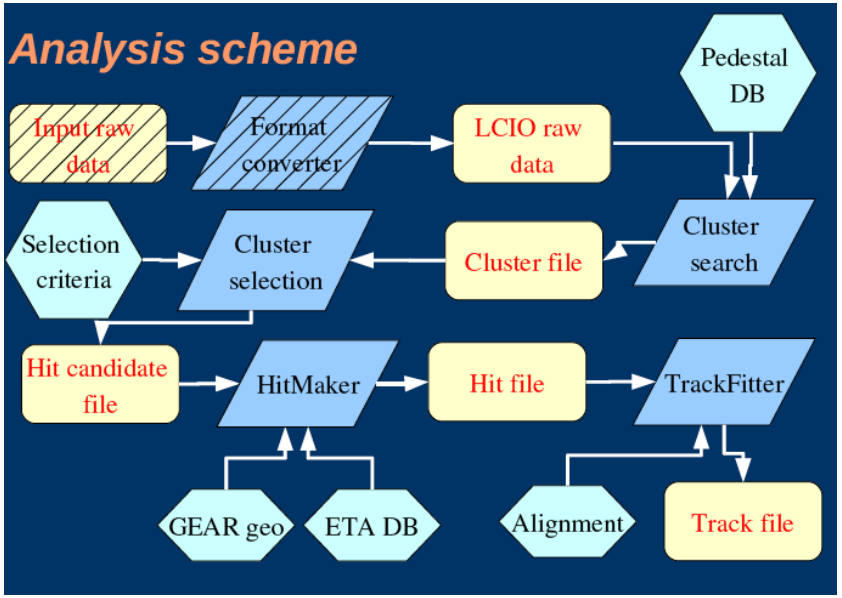
framework users







>300 Mio events
~40 TB (incl.MC/processed)



- framework not only used for ILD detector optimization - also for ILC testbeams:
- **CALICE**
- **MarlinTPC**
- **EUTelescope**
- -> need to further fully support and maintain the tools

Improving the software framework

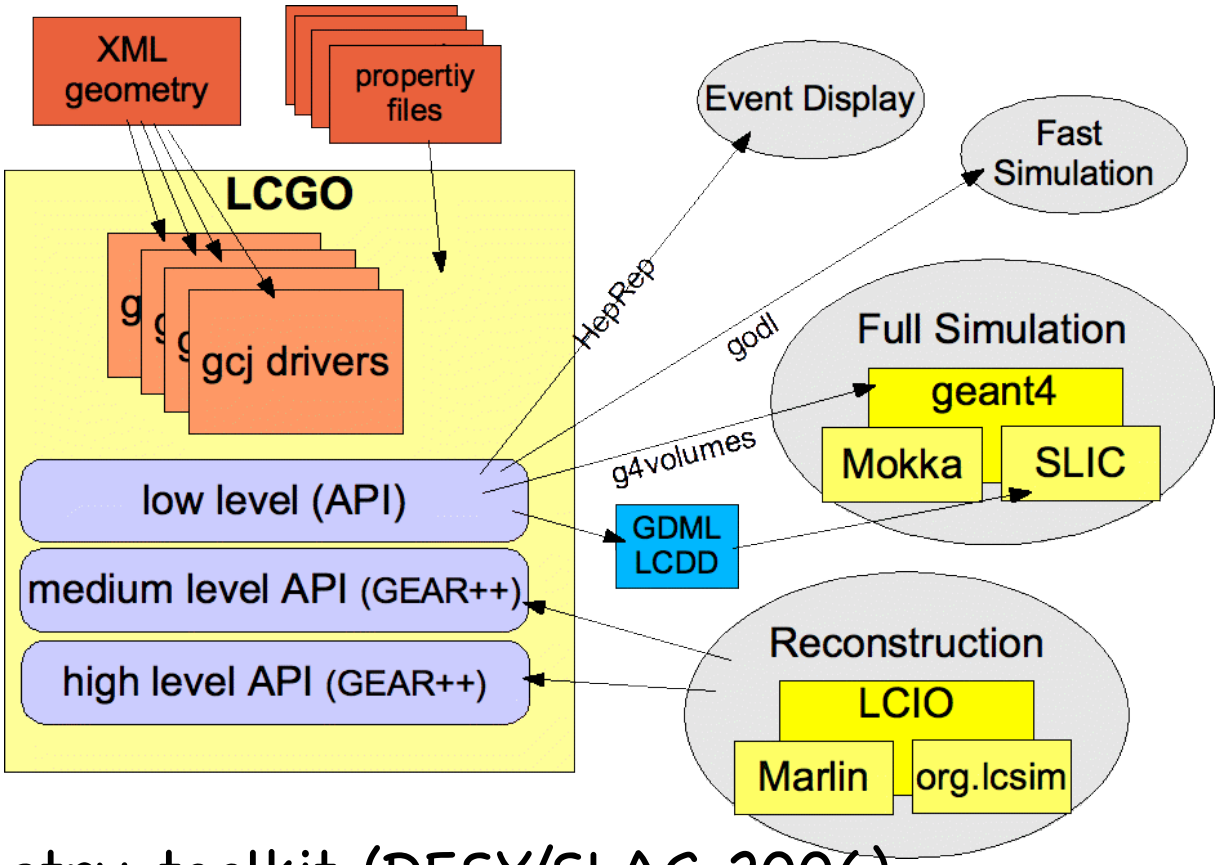
- the Mokka/Marlin framework provides a lot of the need core functionality for HEP data processing
- a complete set of full reconstruction exists: MarlinReco, PandoraPFA, LCFIVertex
- of course, there is always room for improvements
- a number of improvements/new developments where put on hold for the LOI Monte Carlo production
- now after the LOI has been handed in might be a good time to revisit some of these ideas
- -> see next slides

Improving the geometry description

- geometry description
  - Mokka MySQL being the leading system not optimal
  - ideally have standalone geometry system – for
    - simulation, reconstruction, analysis, event displays
    - provide interfaces with the appropriate level of detail at the various stages
  - allow for smooth transition from existing tools (e.g. extend existing GEAR interfaces)
  - unified/combined with conditions data base !?
  - request from CALICE to extend GEAR...

LCGO geometry tool - proposal

- driver based approach a la Mokka
- MySQL DB replaced by xml files



- LCGO - a planned geometry toolkit (DESY/SLAC 2006)
  - based on geometry drivers - written in JAVA !
  - use gcj-compiler to compile to binary & interface with C++
- issues with performance - 4 times slower than C++ (2007)
- -> could look into implementing similar concept in C++
- also investigate existing packages TGeo, VGeometry,...

Improving LCIO -> LCIOv2

- LCIO is persistency and data model - separated through abstract interface
  - -> both can be developed independently !

- improving the persistency
  - I/O performance
  - direct acces
  - splitting&merging files
  - allow for simple streaming of user defined classes (testbeam hardware)
  - evaluate using ROOT-I/O

- improving the data model:
  - allow multiple fits for Tracks
  - provide specialized TrackerHits, e.g. for strips
  - improved relations !?
  - learn from JSF data model...
  - user feedback needed !

time to abandon F77 interface ?

ROOT: to use it or not to use it ?

- decision to not use ROOT for ILC software in meeting at CERN (~2000 ?)
  - based on status of ROOT at the time
  - now with all LHC experiments using ROOT it might be time to revise that decision
- possible integration/usage of ROOT
  - for I/O: using ROOT I/O in LCIO
  - for histograms and trees
    - -> fast interactive user analysis - (e.g.: a la JSF macros)
  - for application framework
  - details to be evaluated/studied ....

Testing and Validation

- testing of software tools has been left to the judgement of the developer
- ideally we should have an automated test system (run at nightly build/on cvs commits)
  - possibly unit tests?
  - integration tests
- also validation of 'physics performance':
  - checkplots, resolutions , overlaps?,....
- -> would have saved us some work in the past

Towards a common ILD software framework

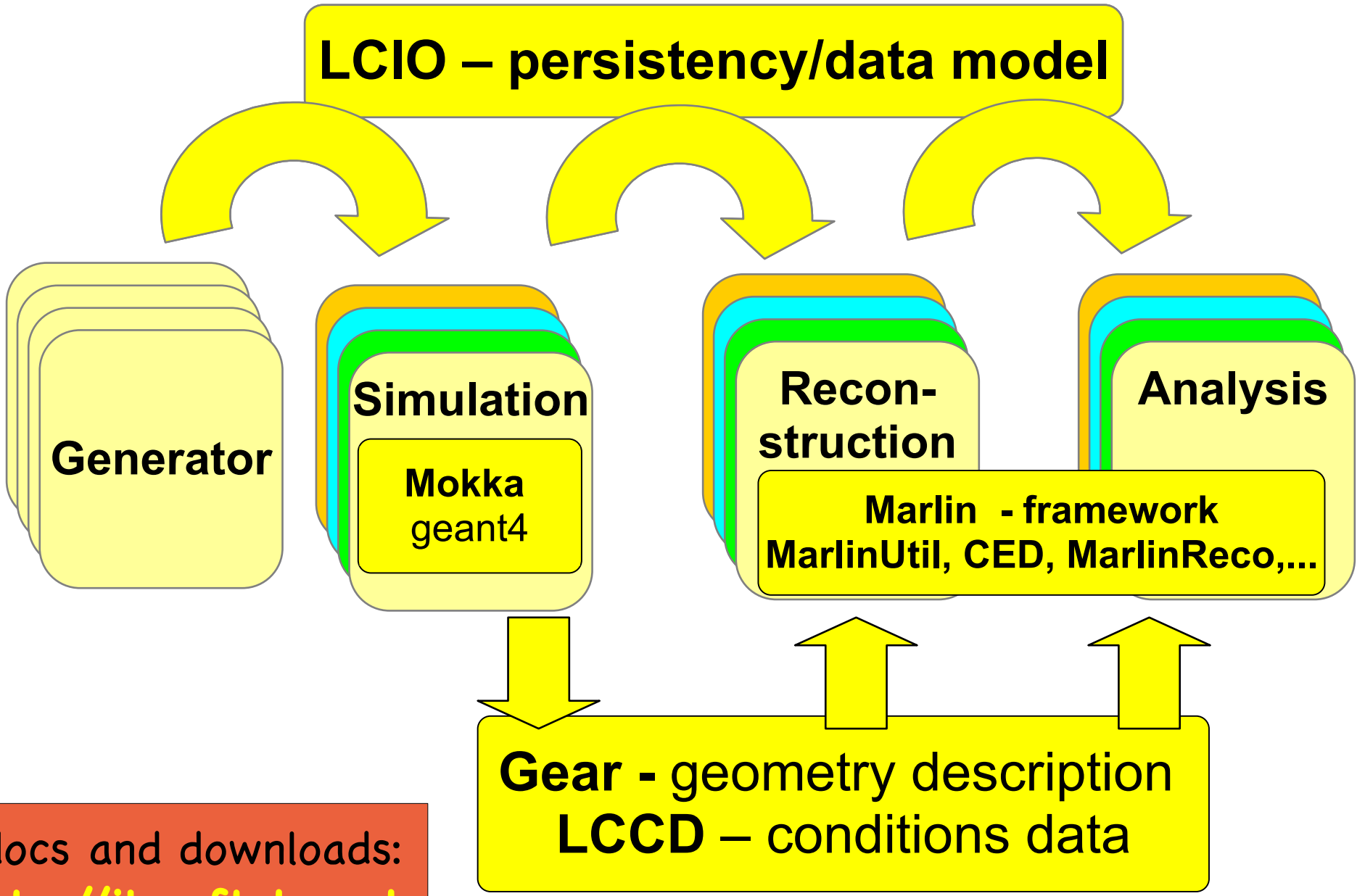
- general agreement that we want to move towards on framework - a possible roadmap :
  - create ILDsoft framework - based on improved versions of LCIO, Marlin with 'goodies' from JSF framework, e.g.
  - investigate tighter coupling to ROOT
    - for user analysis (macros)
    - at I/O level - user defined object serialization
  - easy network access to data - unified for data, configuration, log and histograms
  - JSFEnv user parameters
  - have common ilcinstall build tool
  - work on common geometry system (mid term!?)
  - details will have to be worked out...
  - start in this workshop

Summary & Outlook

- ILD should move towards one software framework - LCIO, Marlin probably good basis together with goodies from JSF
- -> need improvements and new features, e.g.
  - geometry
  - LCIOv2 - data mode and persistency
  - automated test and validation system
  - investigate usage of ROOT
    - interactive user analysis
    - persistency, geometry,.....
- + requirements from user communities - today
- talk to other concepts & groups to seek collaboration on common software tools (TILC09, CERN meeting,...)

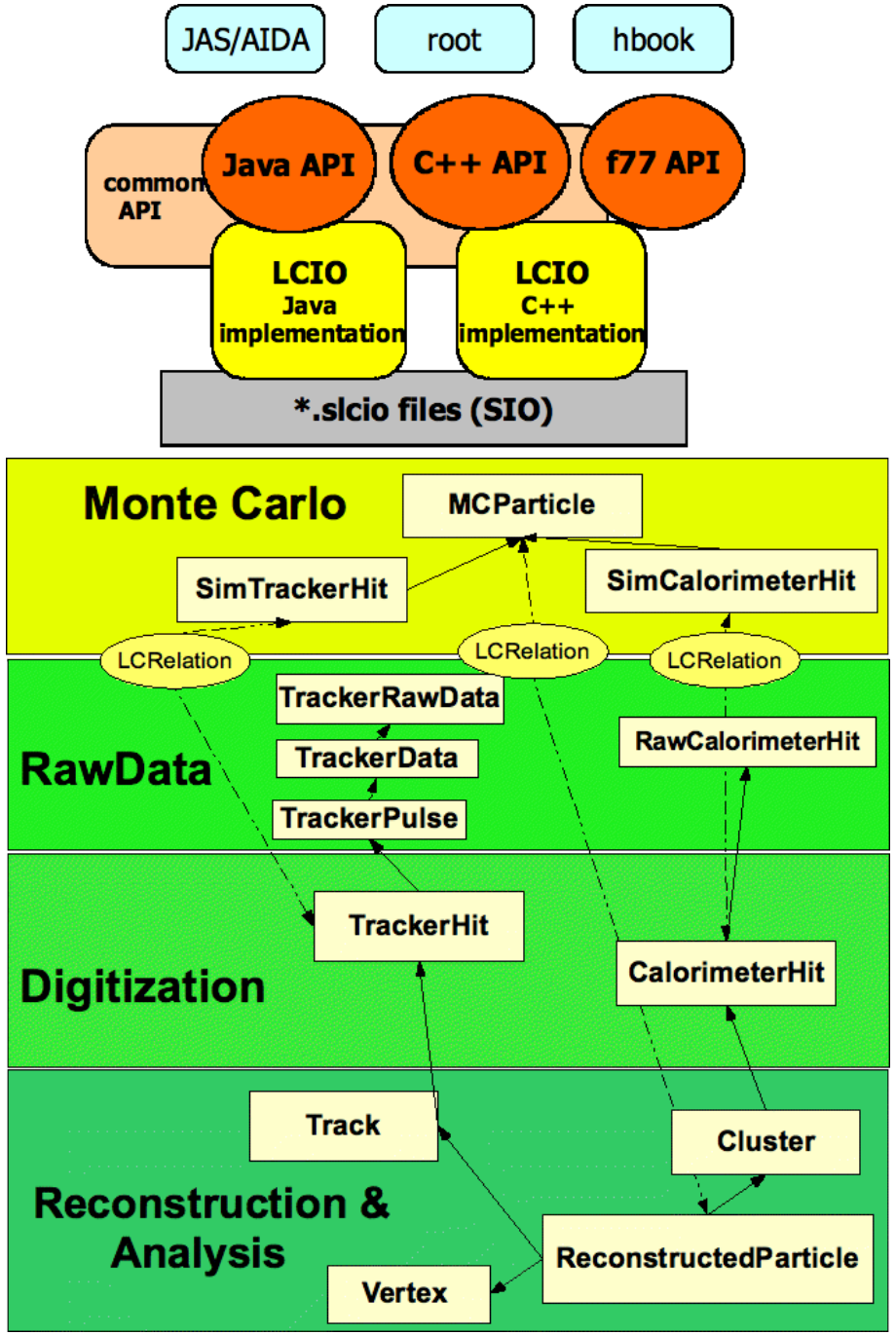
additional material

The ILD software framework (LDC flavor)



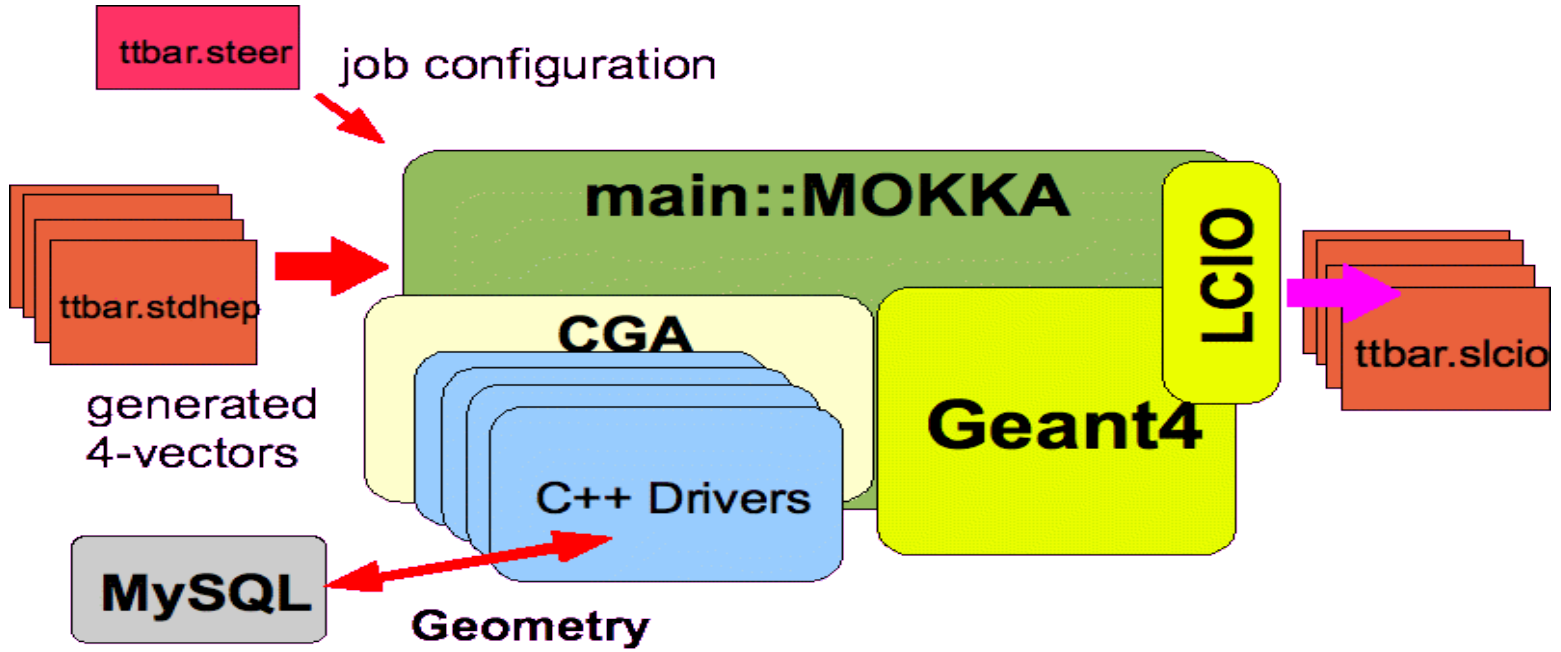
docs and downloads:
http://ilcsoft.desy.de

LCIO: persistency & event data model



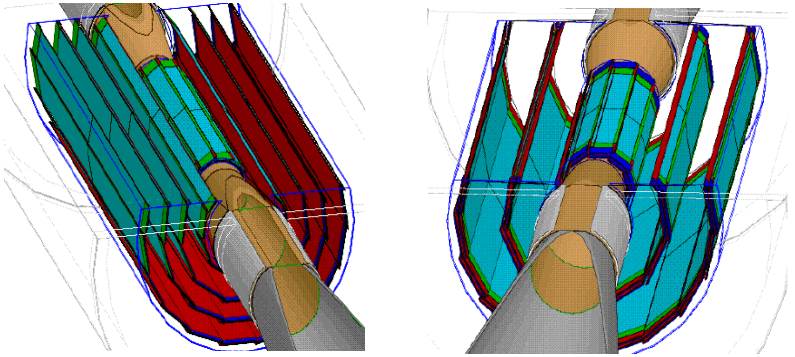
- joined DESY and SLAC project - first presented @ CHEP 2003
- provides persistency (I/O) and an event data model to ILD detector R&D community
- features:
  - Object I/O (w/ pointer chasing)
  - schema evolution
  - compressed records
  - hierarchical data model
  - decoupled from I/O by interfaces
    - C++, Java (and Fortran)
  - some generic user object I/O

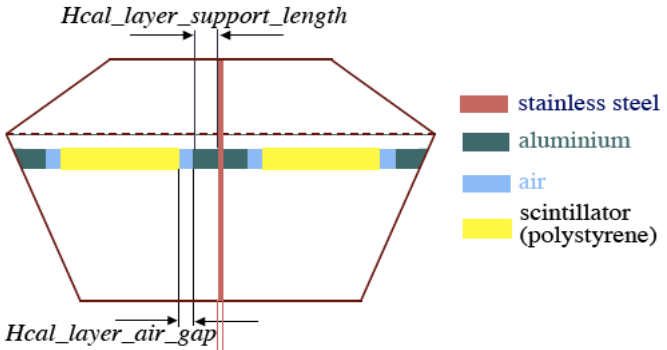
Mokka - geant4 simulation

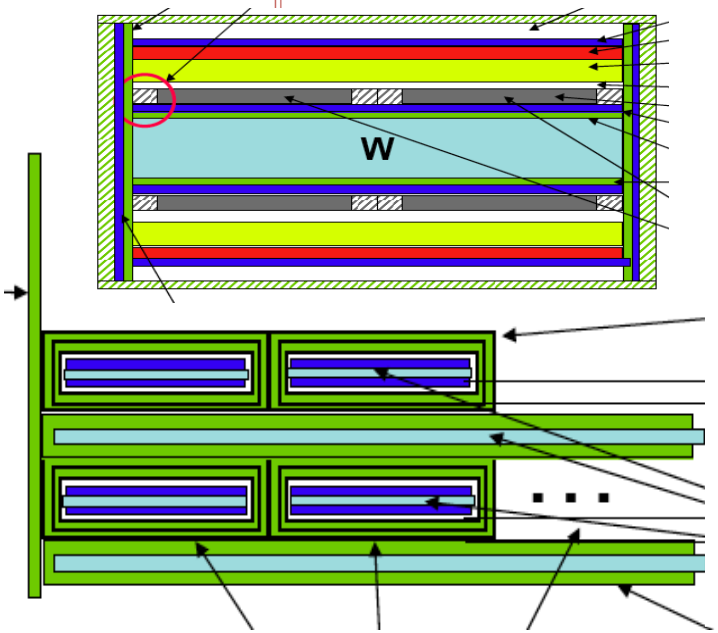


- flexible geant4 based application for simulating the detector response for ILC detector R&D
- developed at LLR (Ecole Polytechnique) with contributions from all detector R&D groups
- flexible geometry on subdetector basis (MySQL & C++ drivers)
- used in ILD detector concept studies
- used in ILC subdetector testbeams: Calice, LCTPC, EUPixelTelescope

ILD detector geometry in Mokka





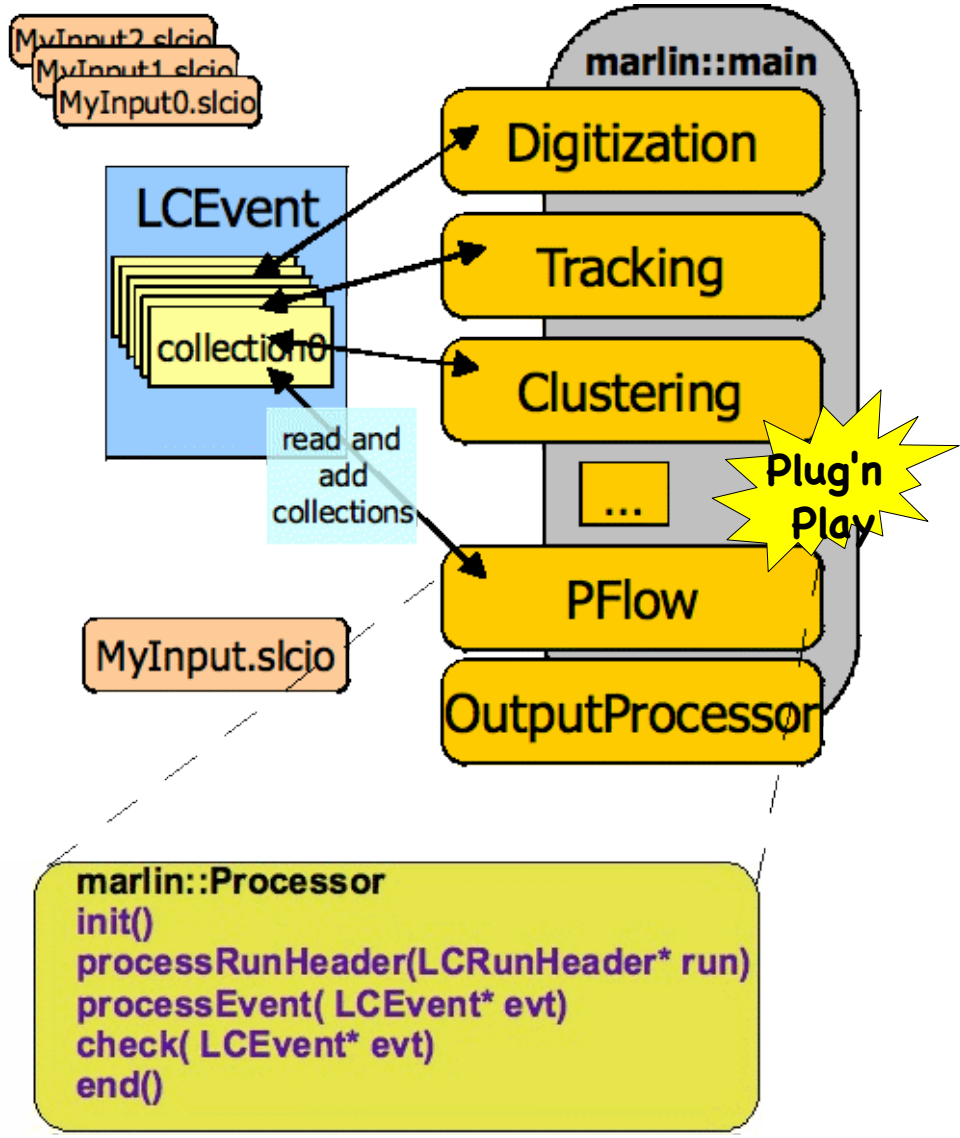


- Mokka allows various levels of detail when defining the detector geometry
- simple and coarse detector models for fast studies
- engineering level of detail as done for ILD_00 model :
  - get material budget right
  - get (in)efficiencies right
  - important for estimating PFA performance

MARLIN application framework

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

- modular C++ application framework for ILC detector R&D
- component based
- shared library **plugins**
- **LCIO** as transient data model
- xml steering files
  - configure application @ runtime
  - processor parameters
- self documenting
- consistency check of input/output collection types



GEAR geometry description

- detailed geometry for simulation with Mokka/geant4:
  - MySQL data base with parameters
  - C++ drivers per subdetector
- at reconstruction:
  - high level abstract interface:
  - per subdetector type (Hcal,TPC,...) parameters/quantities for reconstruction
  - geometry + some navigation
  - implementation uses xml files
  - abstract interface for detailed geometry &materials:
    - point properties