Simulation and Reconstruction Software for the ILC

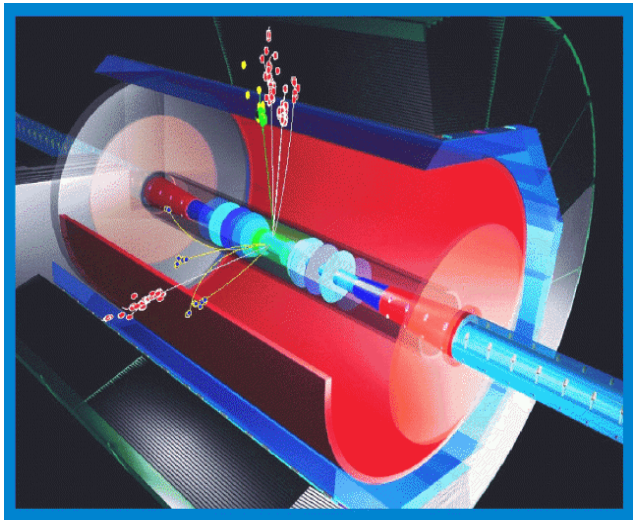
Frank Gaede, DESY

ACAT05, DESY Zeuthen

May 25, 2005

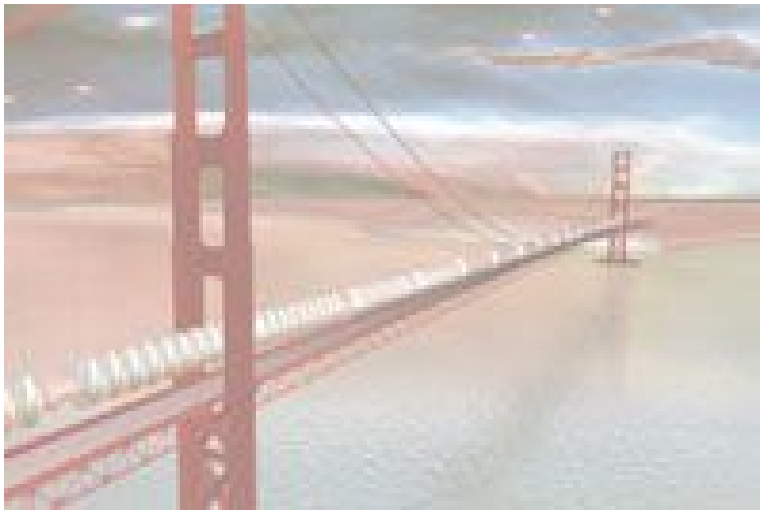
Outline

- ILC - brief overview
- Overview international software
- Software developed at DESY:
  - LCIO – data model & persistency
  - MARLIN – C++ reconstruction framework
  - LCCD - conditions data toolkit
- Summary & Outlook

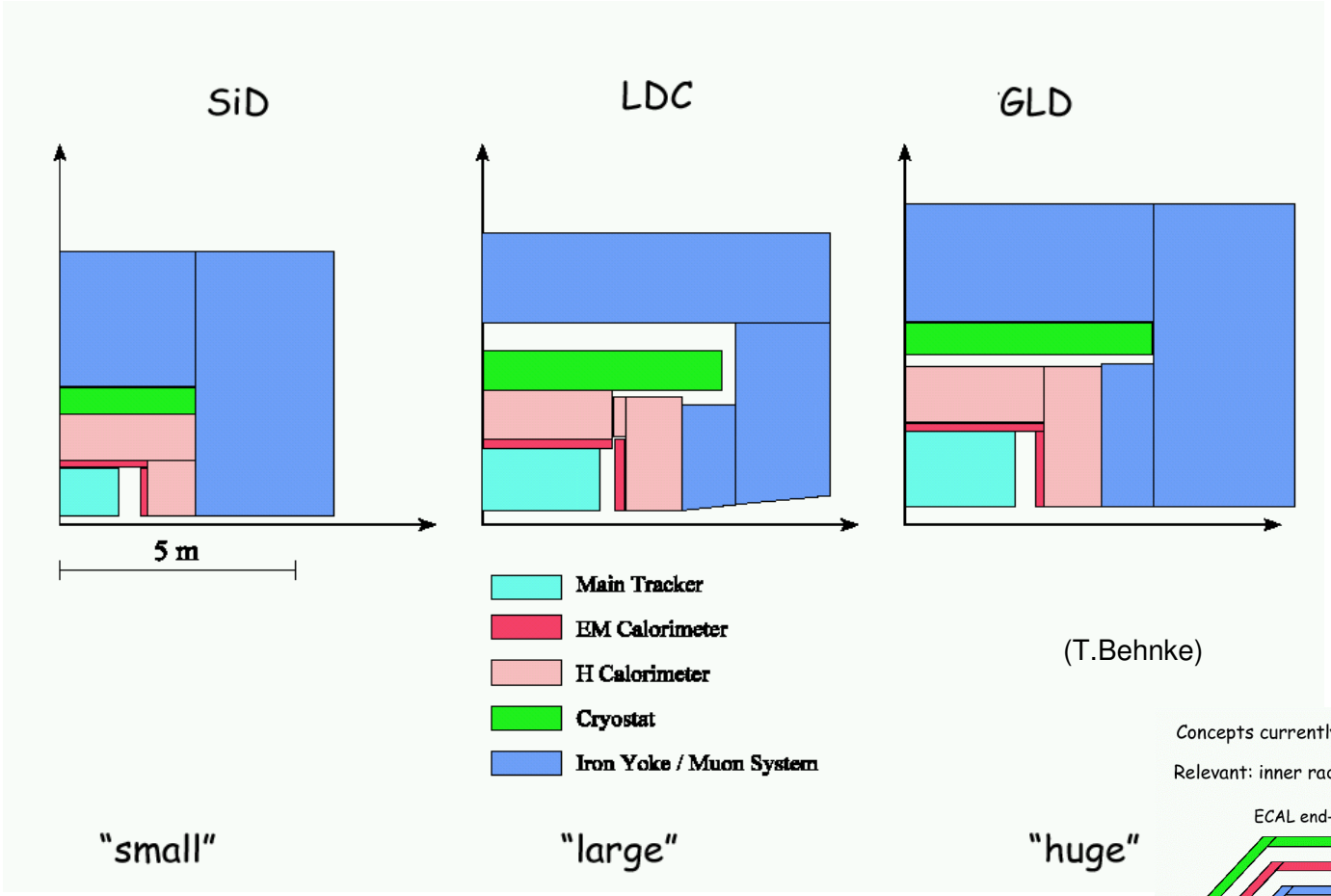


ILC Overview

- future linear e+/e- collider (l~30km)
  - super-conducting RF technology
  - stage 1: E=200→500 GeV
  - stage 2: upgrade E~1TeV
- options: gama/gamma, gamma/e-, e-/e-, Giga-Z, 2 IRs for 2 experiments ?
- rough timeline (planned):
  - (2005) Accelerator CDR
  - (2007) Accelerator TDR – Detector CDRs
  - (2008) LC site selection
  - (2009) Global lab selects experiments
  - (?) Start construction
  - (??) Data
  - **-> operating simultaneously with LHC !**
- currently ongoing: R&D, detector concept studies



Detector Concept Study

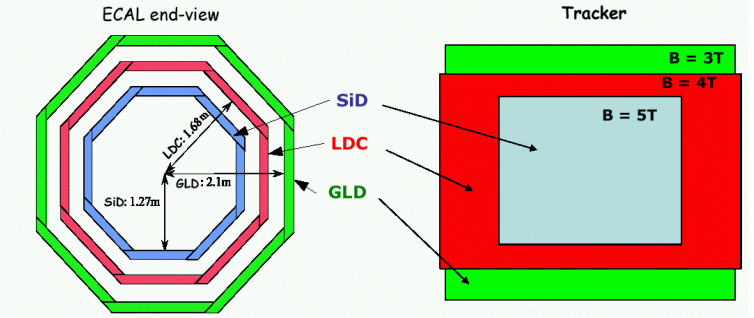


three interregional detector concept studies ongoing

(T.Behnke)

Concepts currently studies differ mainly in SIZE and aspect ratio

Relevant: inner radius of ECAL: defines the overall scale



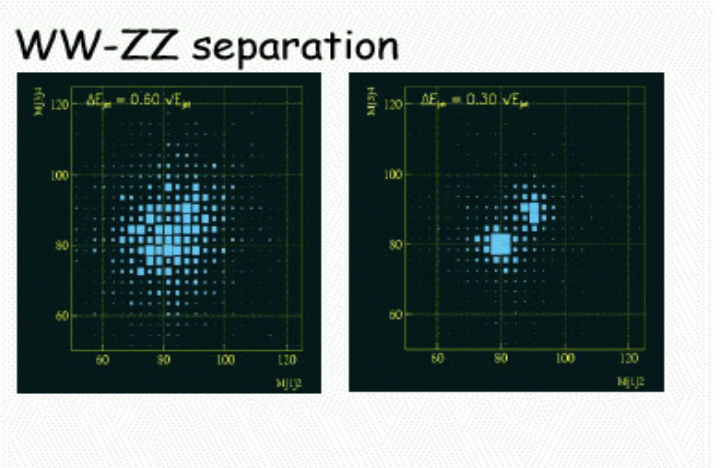
SiD: Silicon based concept

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

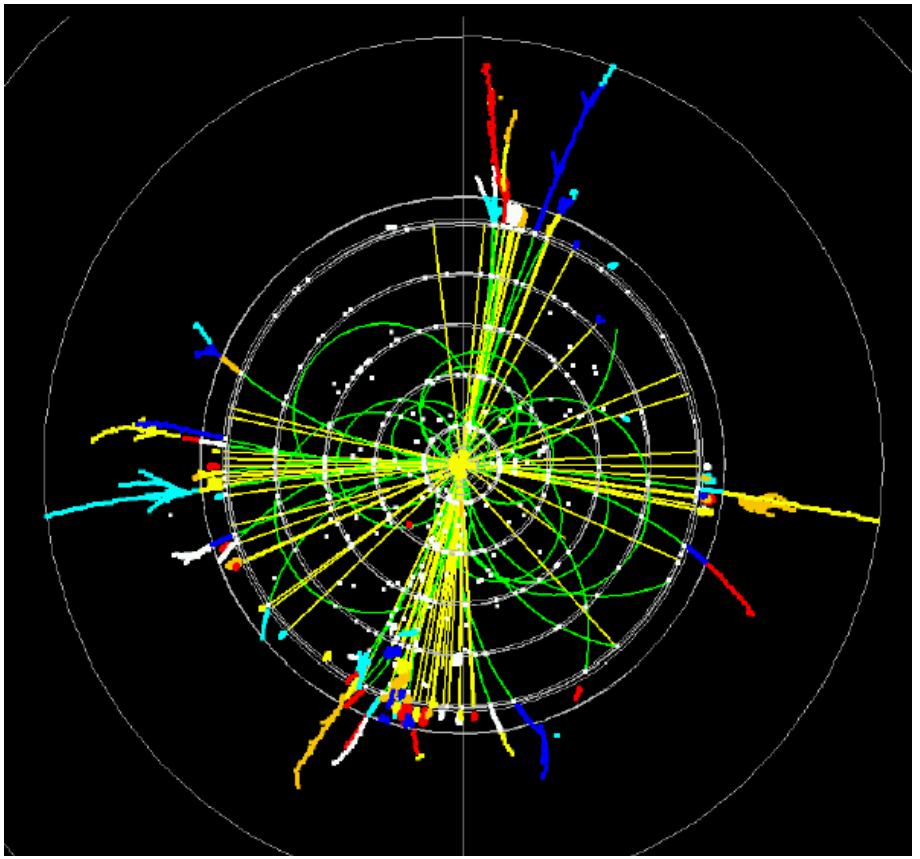
dominant contribution (E<50 GeV):

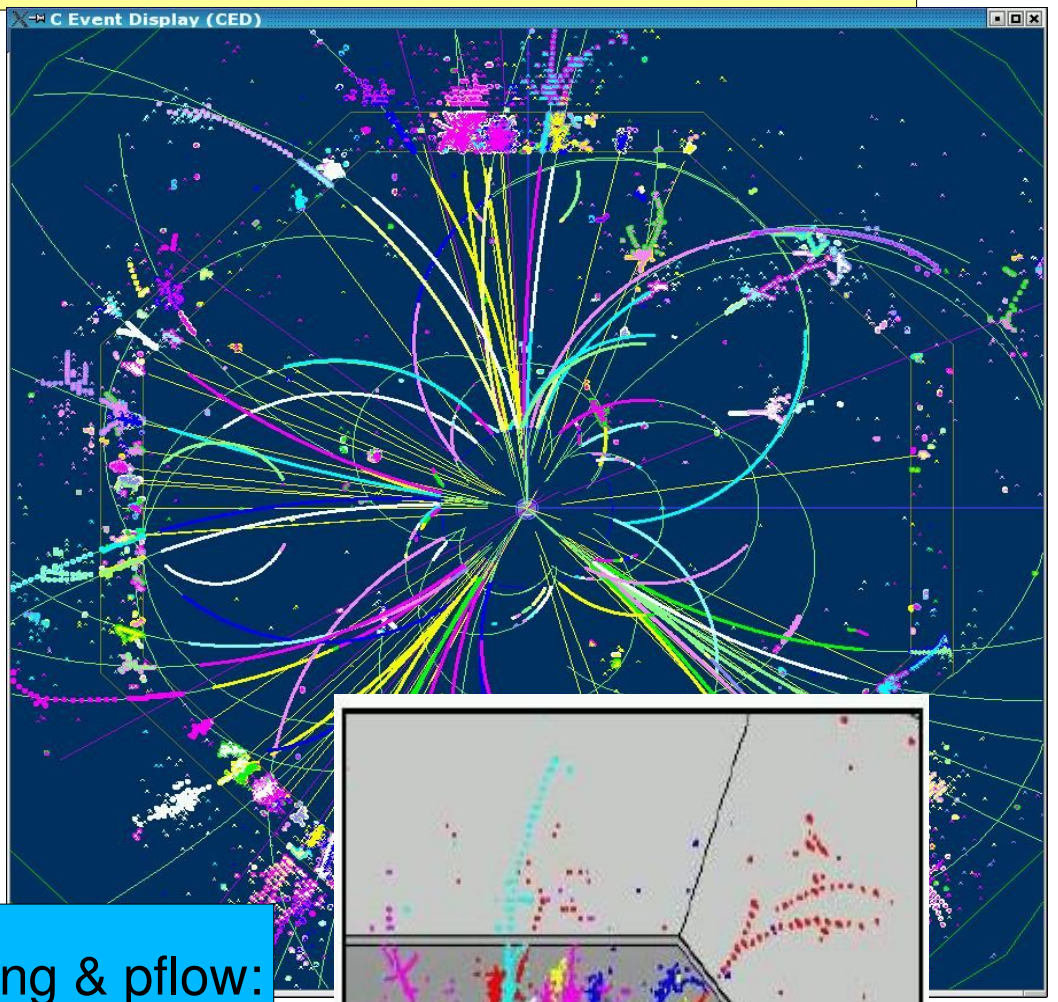
- Hcal resolution
- confusion term

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

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

Imaging calorimeter & ParticleFlow





Need software tools to improve clustering & pflow:

- detailed hadronic shower simulation (**Geant4**)
- framework for developing and comparing pflow algorithms

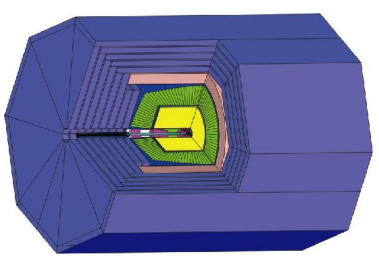
Note: precision tracking and vertexing also very challenging and important !

ILC software packages

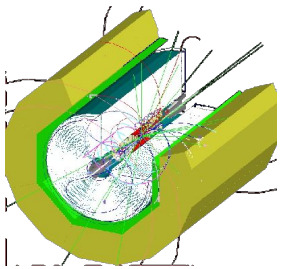
	Description	Detector	Language	IO-Format	Region
Simdet	fast Monte Carlo	TeslaTDR	Fortran	StdHep/LCIO	EU
SGV	fast Monte Carlo	simple Geometry, flexible	Fortran	None (LCIO)	EU
Lelaps	fast Monte Carlo	SiD, flexible	C++	SIO, LCIO	US
Mokka	full simulation – Geant4	TeslaTDR, LDC, flexible	C++	ASCI, LCIO	EU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

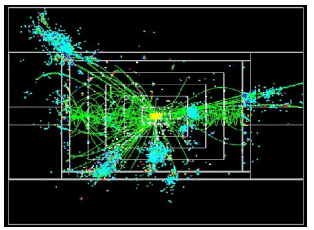
- *Geant4, StdHep and LCIO are common feature*
- *Each trying to be generic with different approach*
  different ways to define geometries



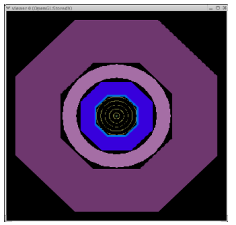
GLD,…



LDC,…



SiD,…



SiD,…

Parameter List

MySQL+CGA

XML

CGD

LCDD

StdHep

Jupiter

Mokka

LCDG4

SLIC

root

sio

Jeremy McCormick

LCIO: *SimTrackerHits/SimCalorimeterHits*

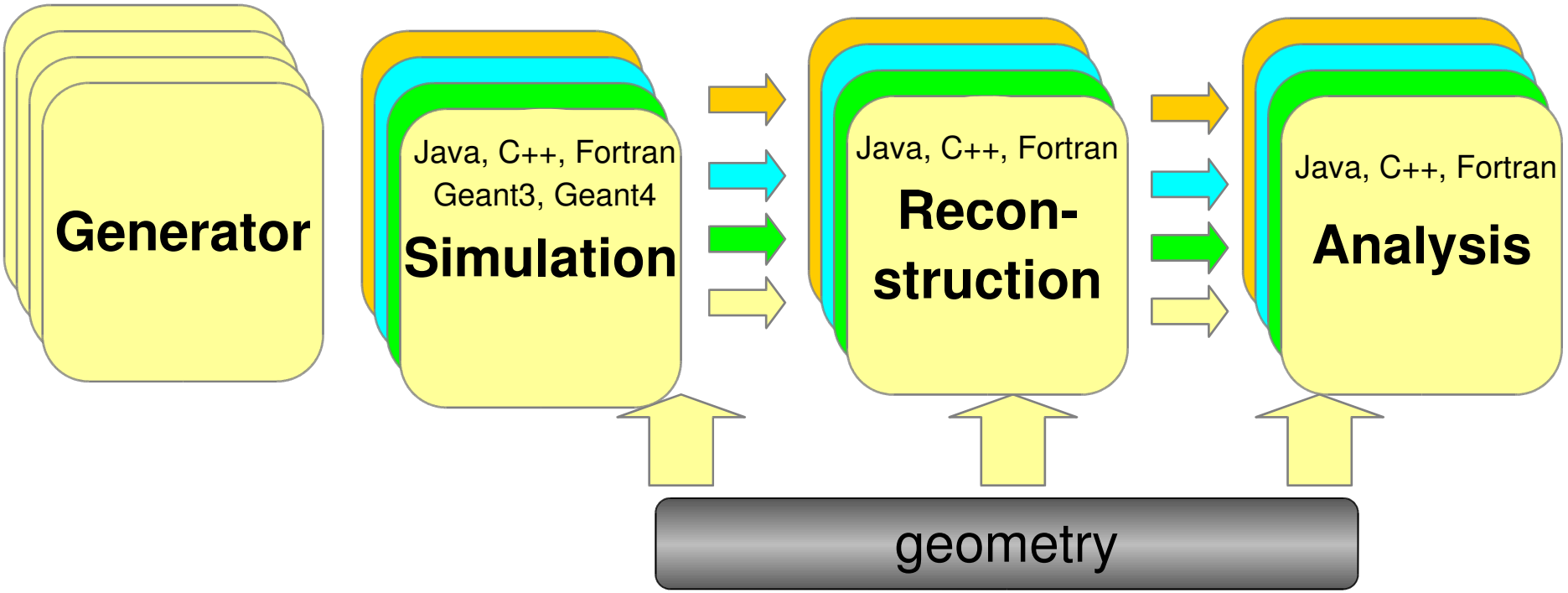
(A.Miyamoto)

A Common Software Framework for the ILC ?

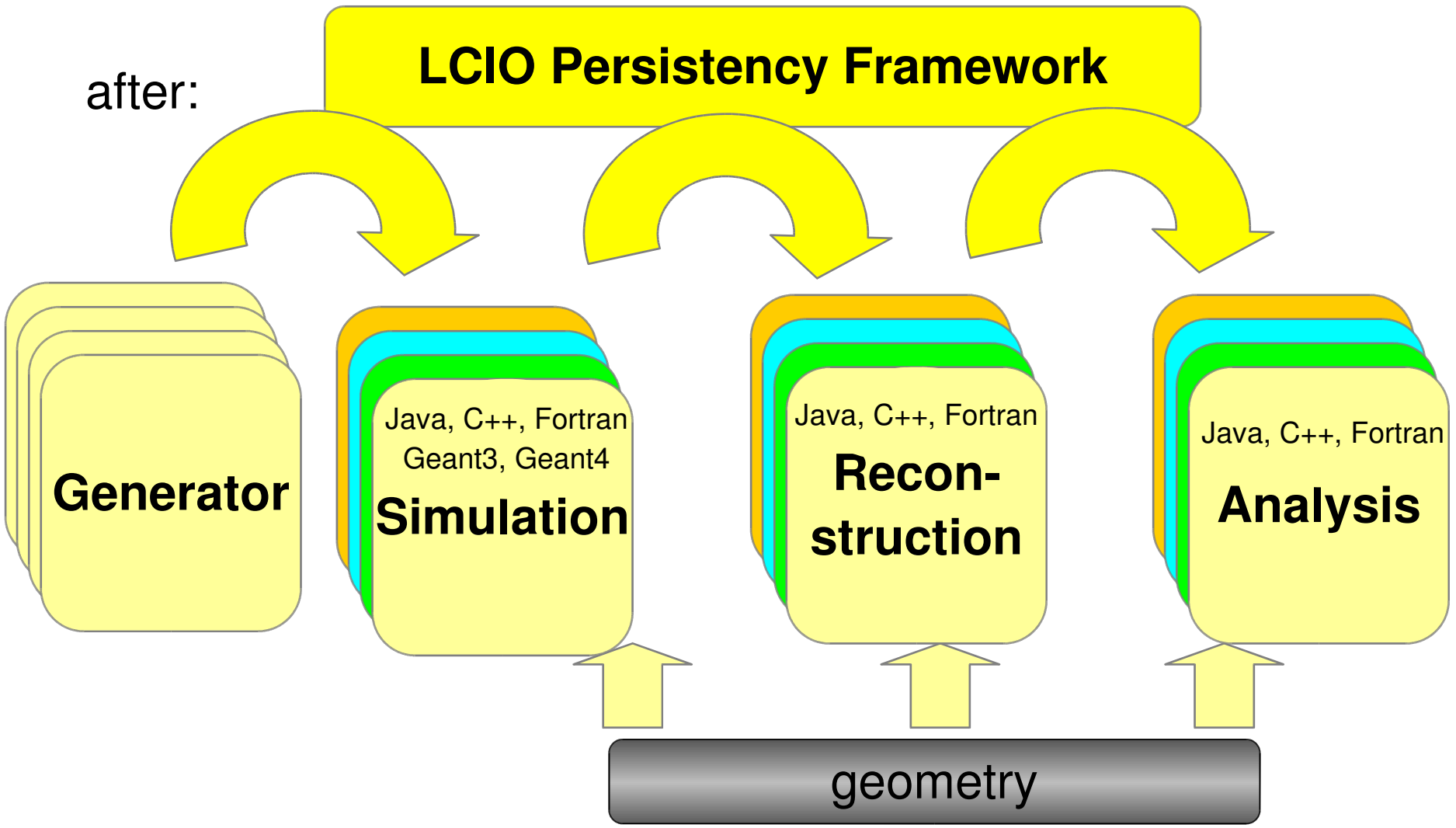
- a common software framework for the ILC that is flexible and easy to use would be highly desirable !
- but:
  - ILC emerged out of three regional studies
  - all groups have developed their own software as needed for the R&D
  - different languages used: C++, Java, f77
  - ...
- aim:
  - develop modular software and define interfaces so that packages can coexist/cooperate - and eventually converge
- example: **LCIO**

Motivation for LCIO

before:



Motivation for LCIO



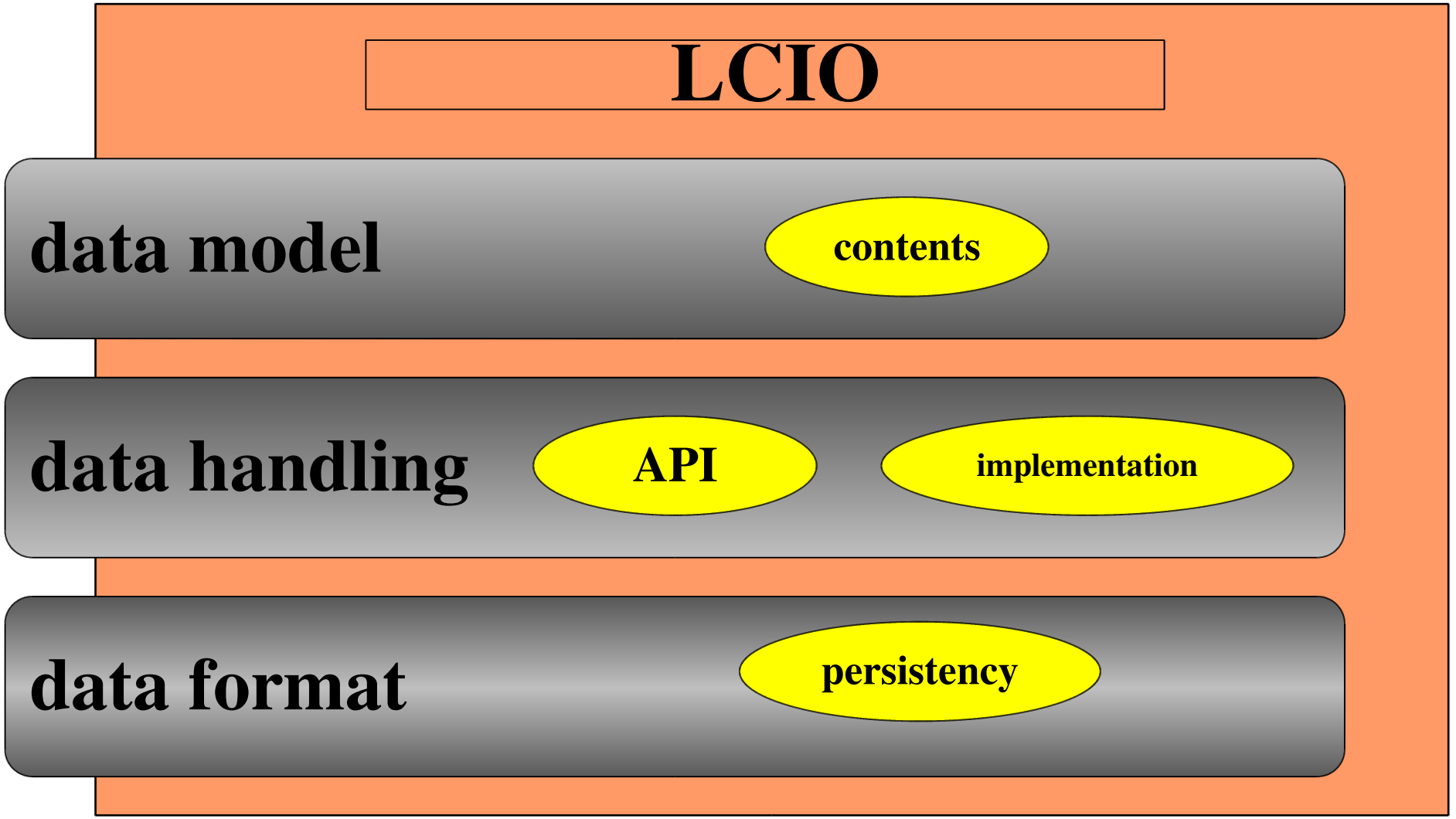
LCIO project overview

Linear **C**ollider **I**nput **O**utput

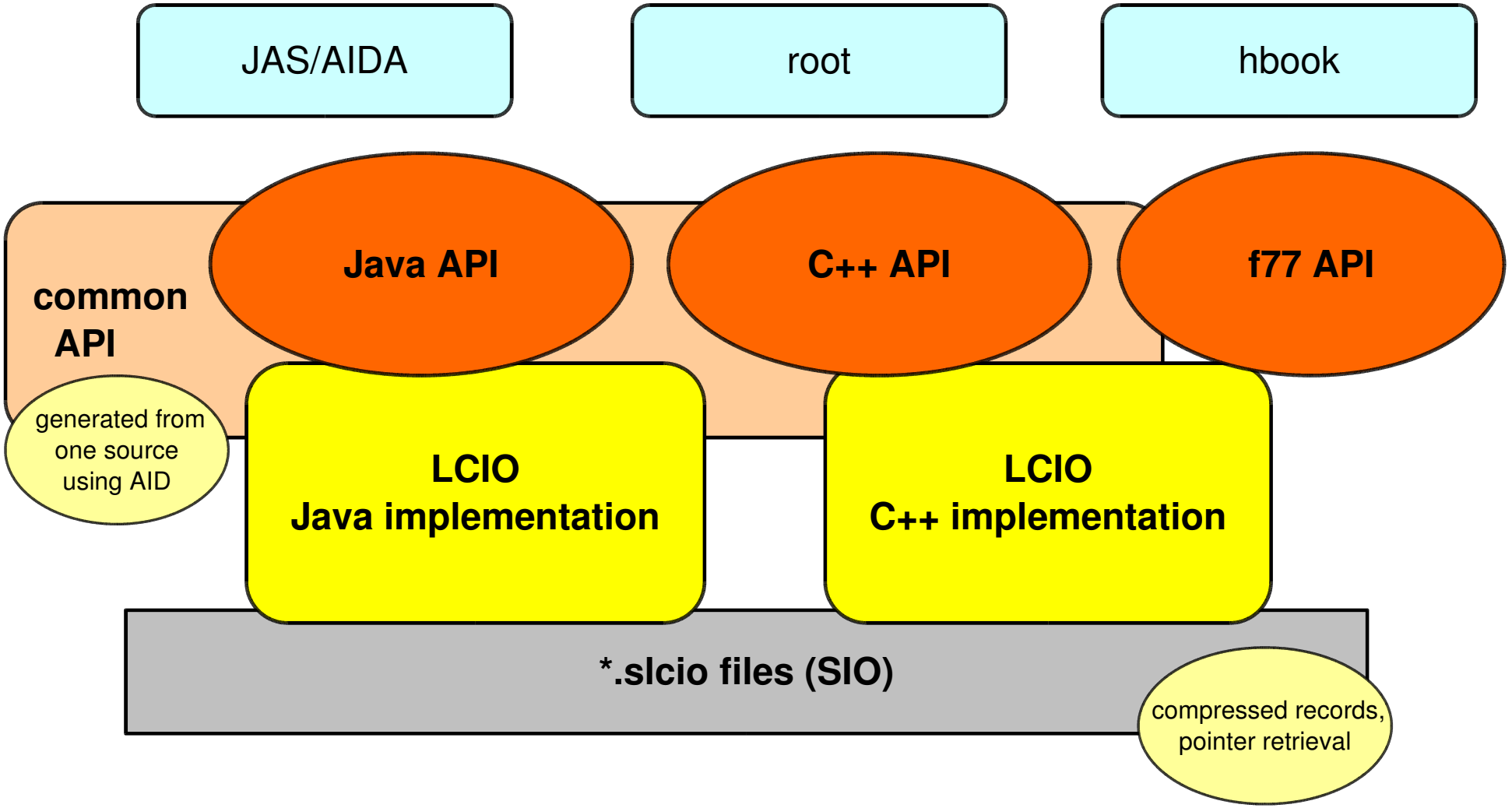
- DESY and SLAC joined project:
  - provide common basis for ILC software
  - started end of 2002
- Requirements
  - need Java, C++ and f77 (!) API
  - extensible data model for current and future simulation and testbeam studies
  - user code separated from concrete data format
  - easy to adapt LCIO in existing applications
  - no dependency on other frameworks

-> keep it simple & lightweight

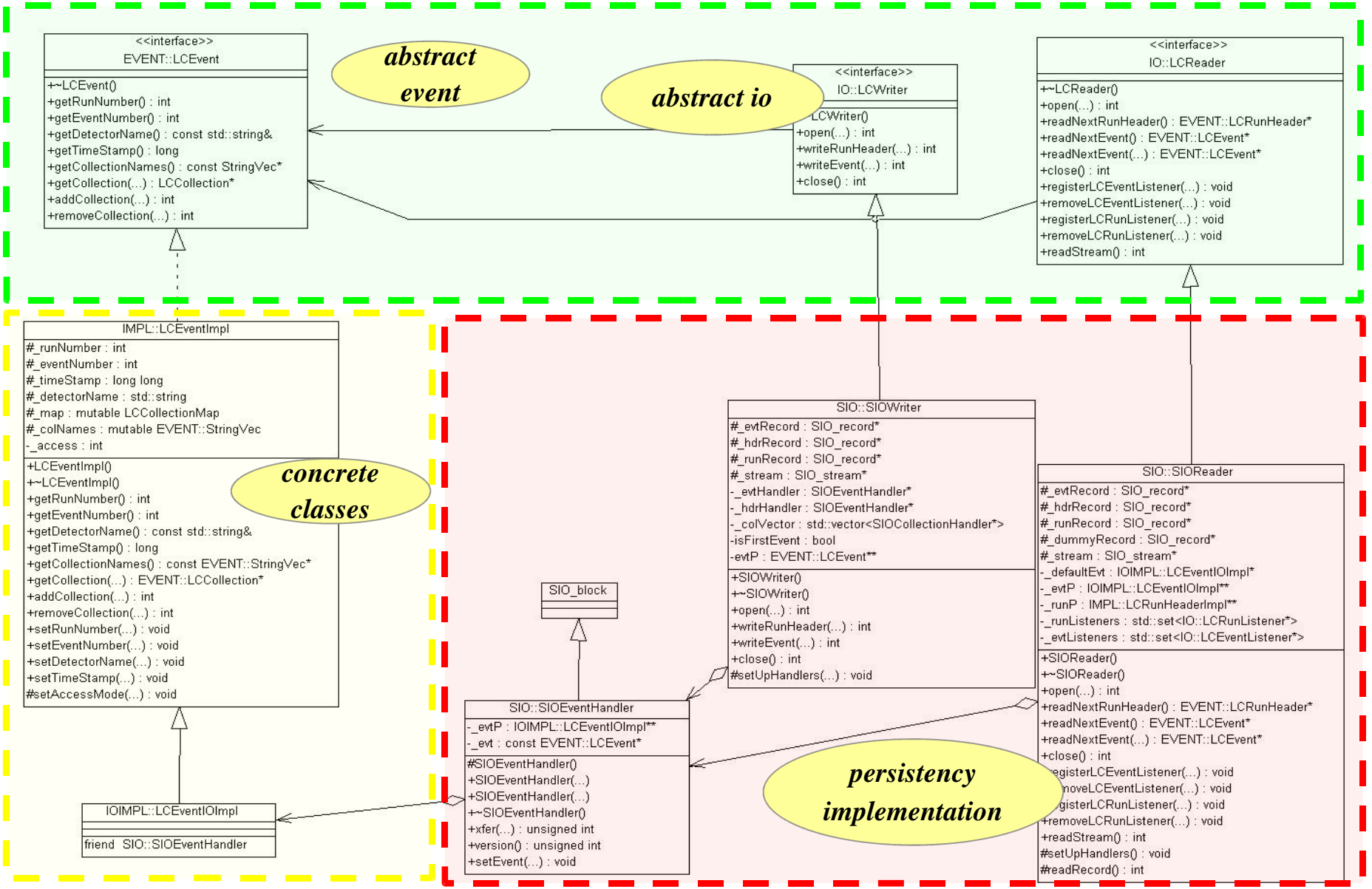
LCIO Building Blocks



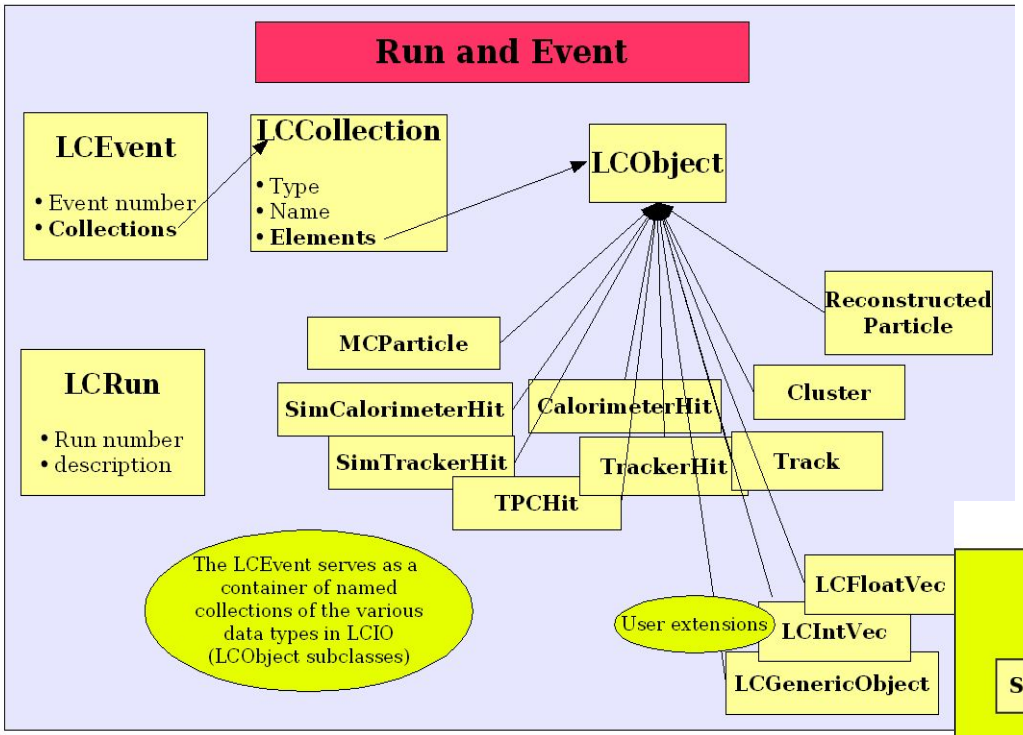
LCIO SW-Architecture



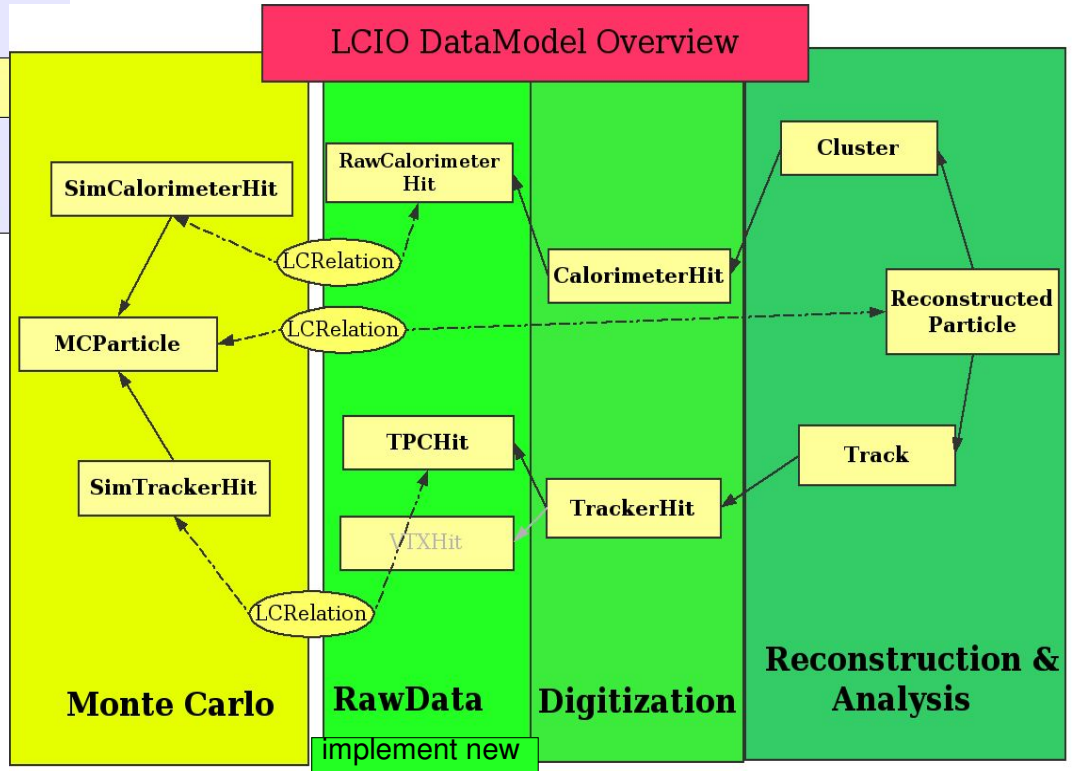
LCIO class design



LCIO data model



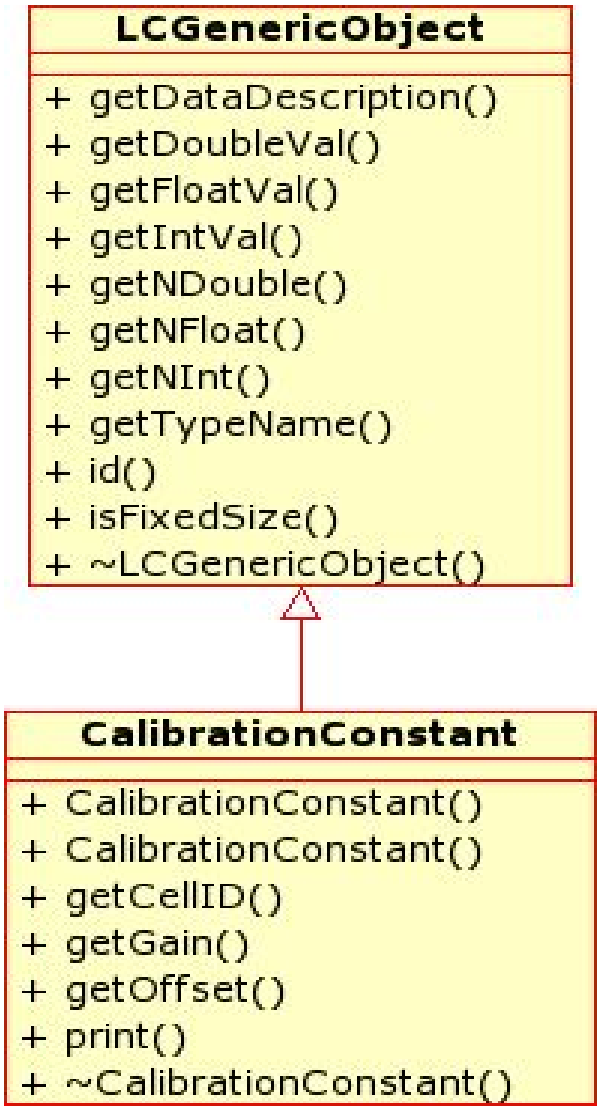
event serves as container of untyped collections



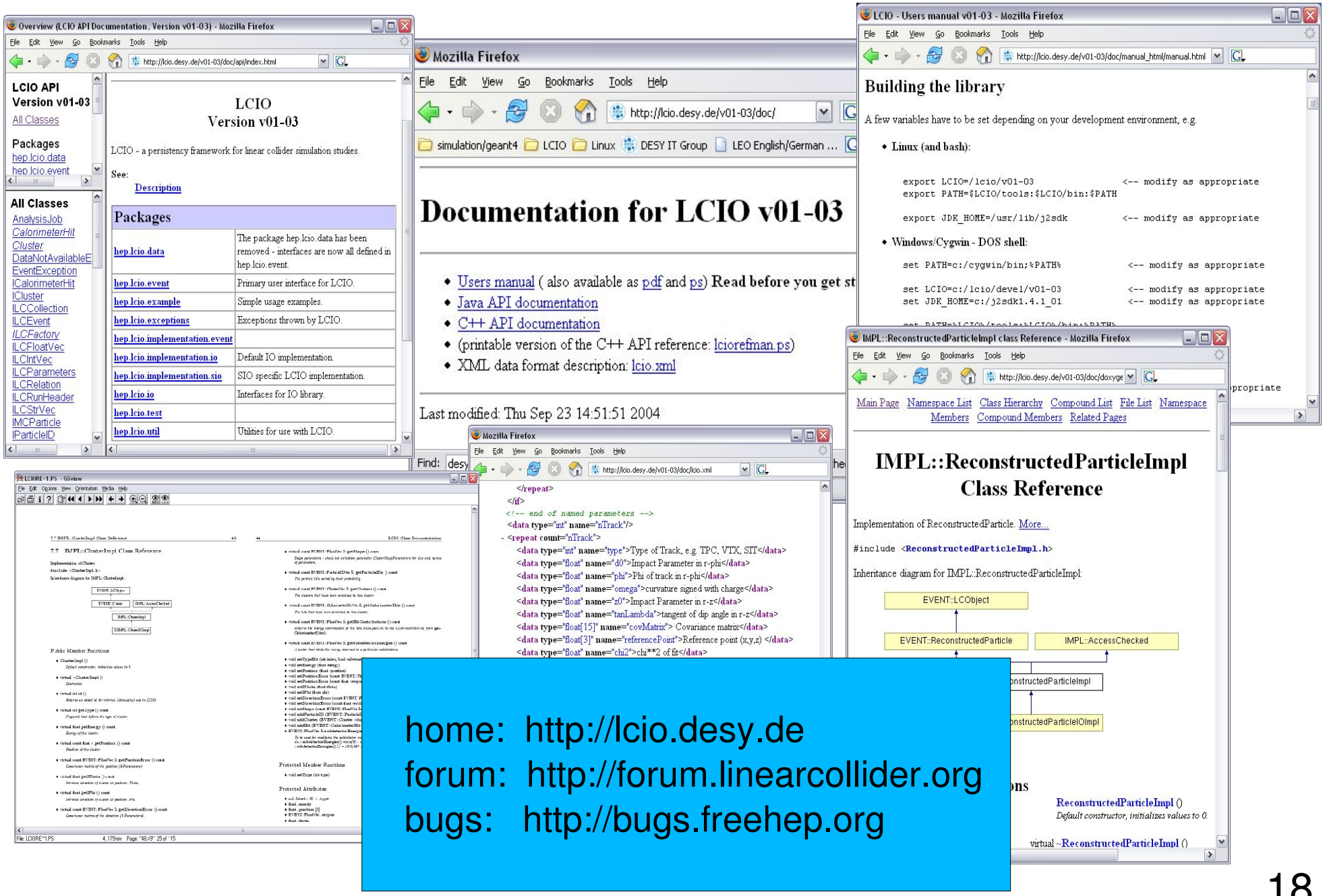
hierarchy of data objects in the event

LCIO – Generic User Information

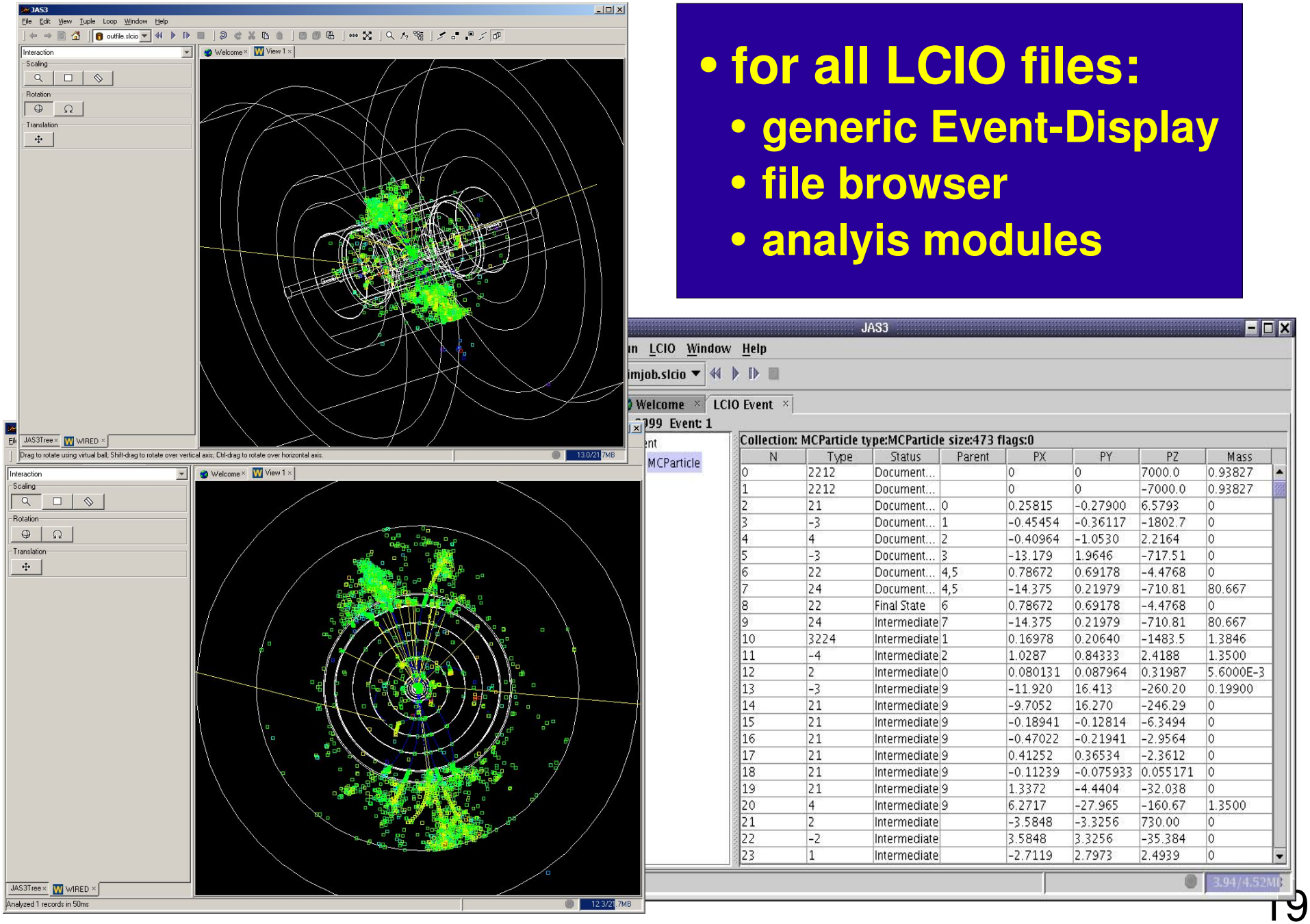
- LCIO data model defines the object needed for ILC simulation studies, but
  - users want additional information in files for specific studies
  - can't create new classes within LCIO for all requests and purposes
  - need generic user class: **LCGenericObject**
    - almost arbitrary data objects
    - typically access provided through user subclass - but not needed:
    - has description string for reading the data without need to have access to data dictionary (library)



LCIO on the web



JAS3 and LCIO



- **for all LCIO files:**
  - **generic Event-Display**
  - **file browser**
  - **analyis modules**

LCIO - Summary

- LCIO provides a common data model and file format for ILC studies
- has become a defacto standard for ILC software packages
- provides Java, C++ and Fortran interface
  - **could serve as a basis for a common ILC software framework !**

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

LCIO can also be used as transient data model in analysis and reconstruction software -> **MARLIN**

Marlin - Motivation

- currently the most complete ILC reconstruction software implementing the pflow algorithm is **Brahms**
  - written in Fortran
  - code recycled from LEP and other experiments
  - based on geant3
  - hard coded geometry definition
  - inflexible and hard to maintain/extend
- so there is need for something new which
  - is written in OO-language
  - is flexible in terms of the detector geometry
  - is easy to use and maintain
  - allows for collaborative development and exchange of code and algorithms

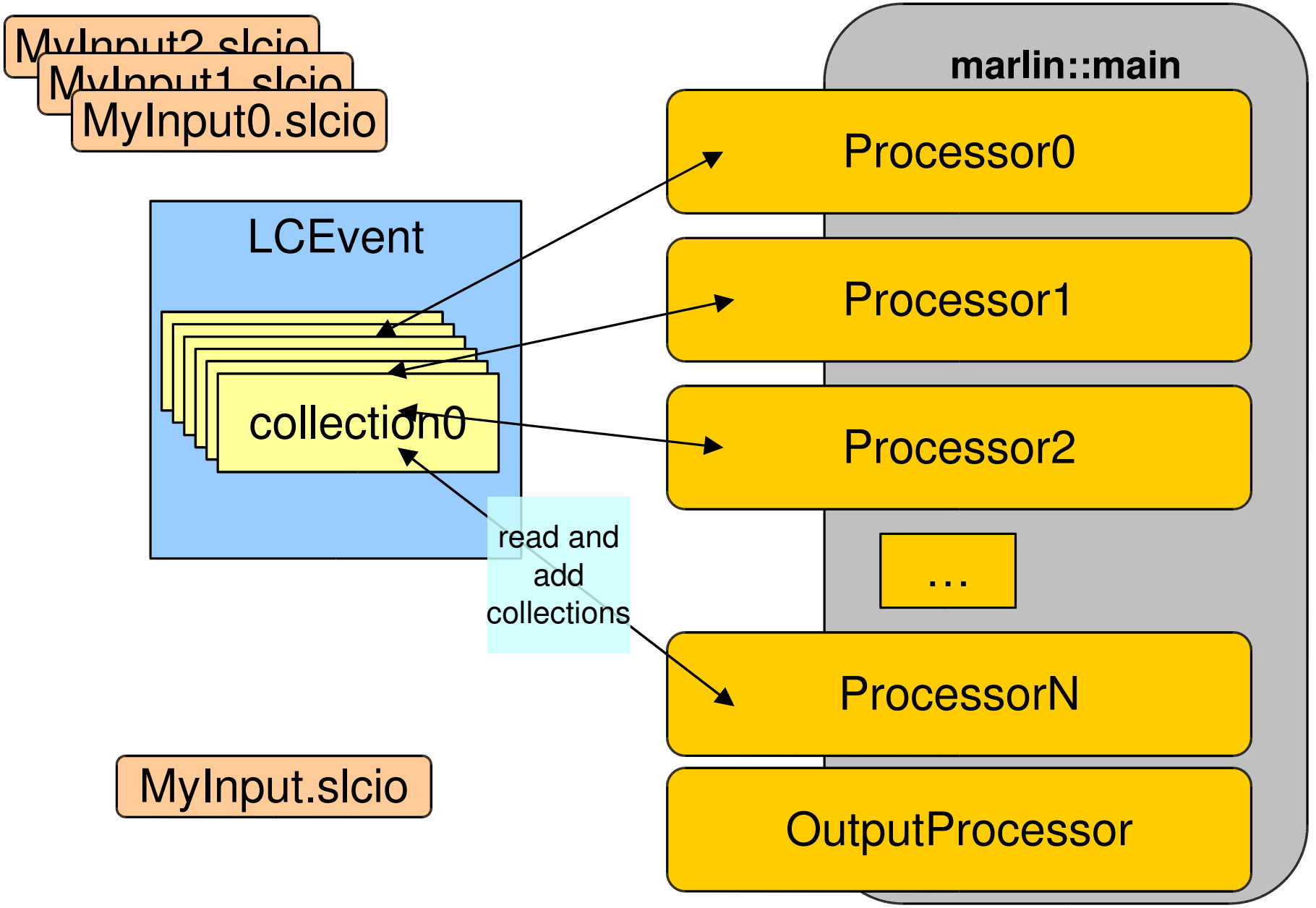
Marlin - Introduction

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
- similar to US org.lcsim Java framework

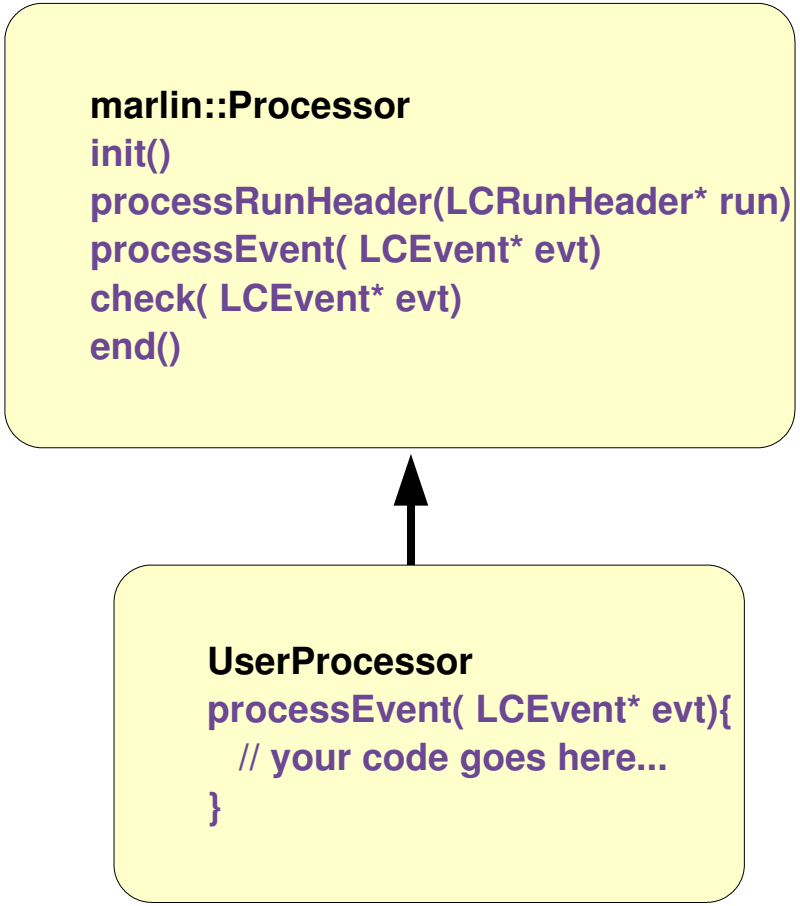
- Application framework:
  - set of classes that provide the core functionality needed in problem domain and provide hooks (callbacks) for specific user code
  - provides main program
  - note: most current experiments that use OO (C++) have application frameworks

Marlin – schematic overview



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 features

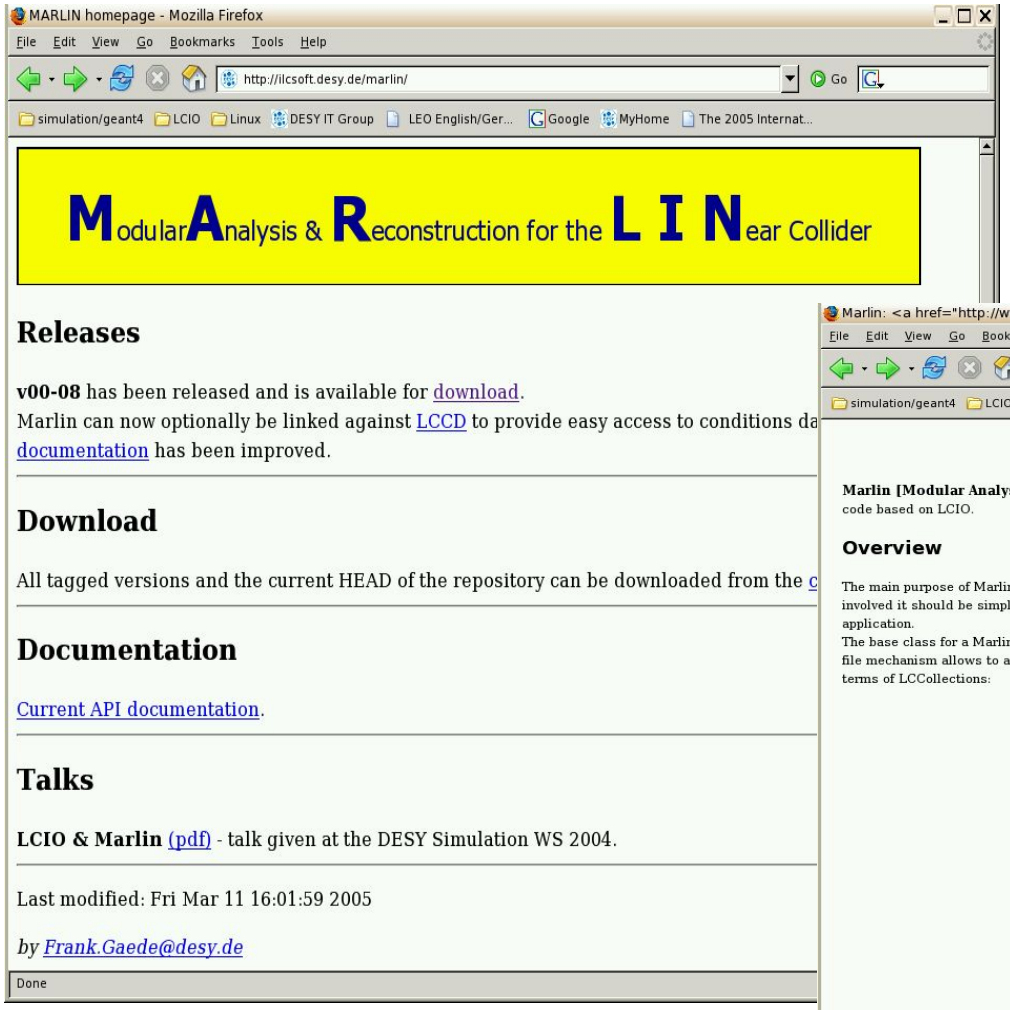
- core processors
  - AIDAProcessor
    - for easy creation of histograms, clouds, ntuples
  - OutputProcessor
    - writes current event or subset thereof
  - ConditionsProcessor
    - read conditions transparently with LCCD
  - OverlayProcessor
    - event mixing (under development)
  - SimpleFastMCProcessor
    - fast smearing Monte Carlo (under development)
  - MyProcessor
    - simple example – serves as template for user code
- fully configurable through steering files
- self-documenting:
  - MyApplication -l

    will print all available processors with their parameters and example/default values
- not yet:
  - logging manager
  - exception handling
  - program flow control
  - ...

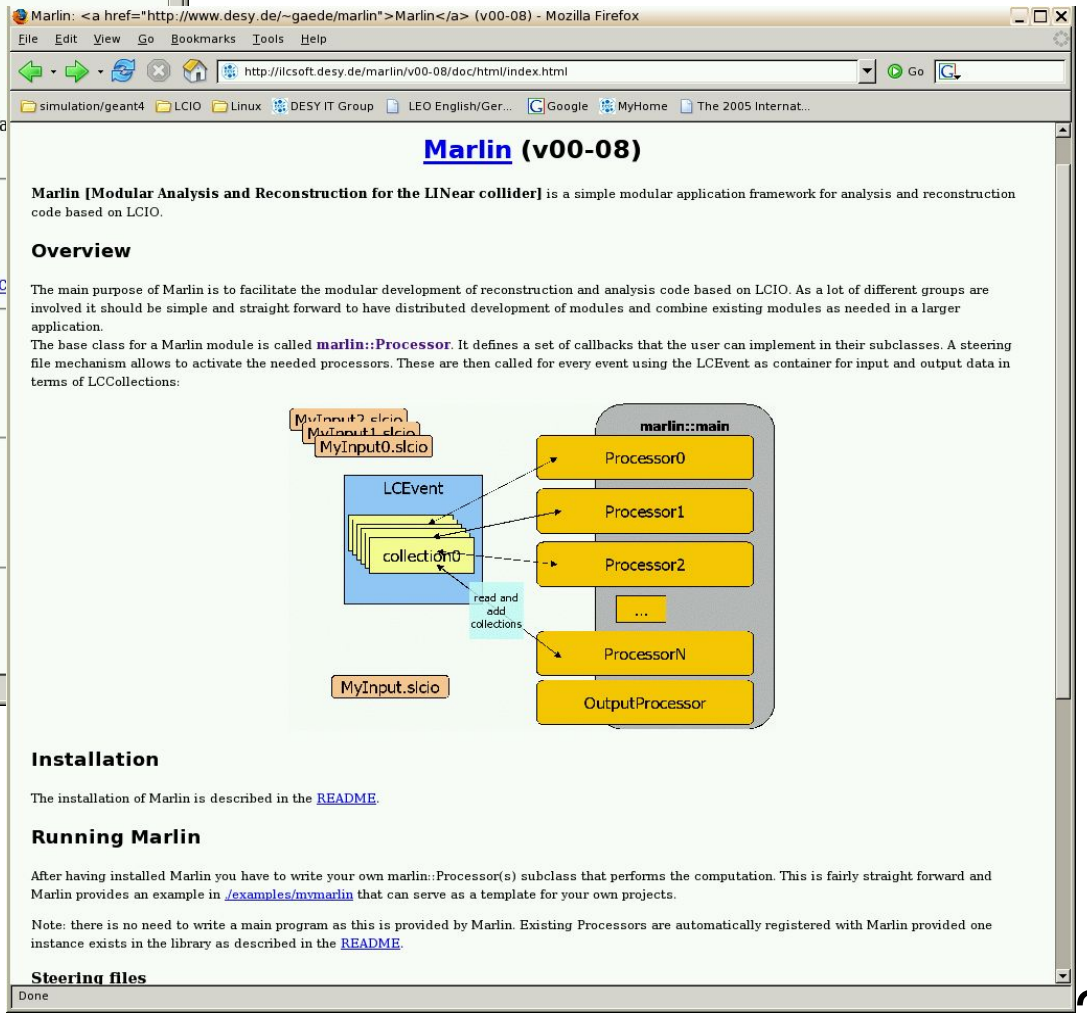
Marlin users

- **Marlin serves as a framework for the distributed development of a full suite of reconstruction algorithms !**
- **aim: have (at least one) complete set for standard reconstruction in C++ soon !**
- ongoing activities:
  - Reconstruction software
    - wrapper for Brahms-Tracking code (S.Aplin)
    - clustering and pflow (A.Raspereza,V.Morgunov)
    - clustering algorithms (Ch. Ainsley, G. Mavromanolakis)
  - Analysis software
    - LCLeptonFinder (J.Samson)
    - JetFinder (Th.Kuhl)
    - ThrustFinder (Th. Kraemer)
  - CALICE testbeam software
    - DigiSim (G.Lima)
    - Ganging and Calibration (R.Poeschl)
  - probably others ...

Marlin on the web



http://ilcsoft.desy.de/marlin



Marlin - Summary & Outlook

- modular C++ application framework for the analysis and reconstruction of LCIO data
- ongoing development of reconstruction processors with the aim to have (at least one) complete set for standard reconstruction in C++ soon !
  - until then: reconstruction workhorse still Brahms (f77)

- **To Do:**
  - **geometry description for reconstruction** – ongoing:
    - have abstract geometry API ala LCIO that can be used in C++ and Java
    - -> need agreement in ILC community
    - also a good idea for Marlin alone ...
  - investigate options for interoperability with US framework, i.e.

    Java-C++ interfacing
    - technically feasible, but not so straight forward how to do this properly !

both features needed for a common ILC framework !

LCCD Motivation

- fairly complete software chain for simulation and reconstruction for the ILC exists (or is under development)
- can be also used for the simulation of upcoming subdetector testbeam studies
- one important ingredient is missing:

  conditions database

-> LCCD

LCCD

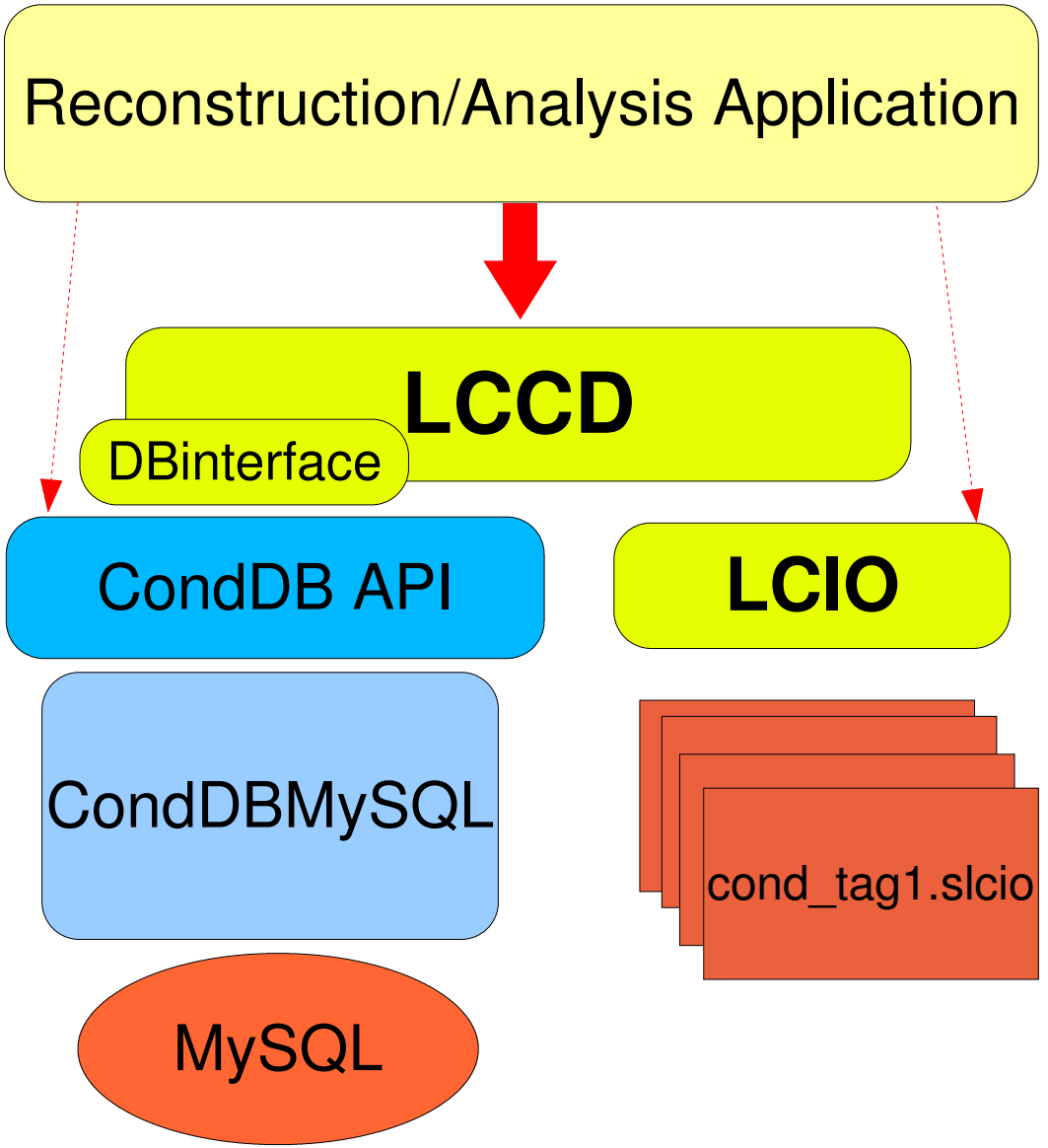
Linear **C**ollider **C**onditions **D**ata Toolkit

- handles access to conditions data transparently from
  - conditions database (CondDBMySQL (by Lisbon Atlas group))
  - LCIO files

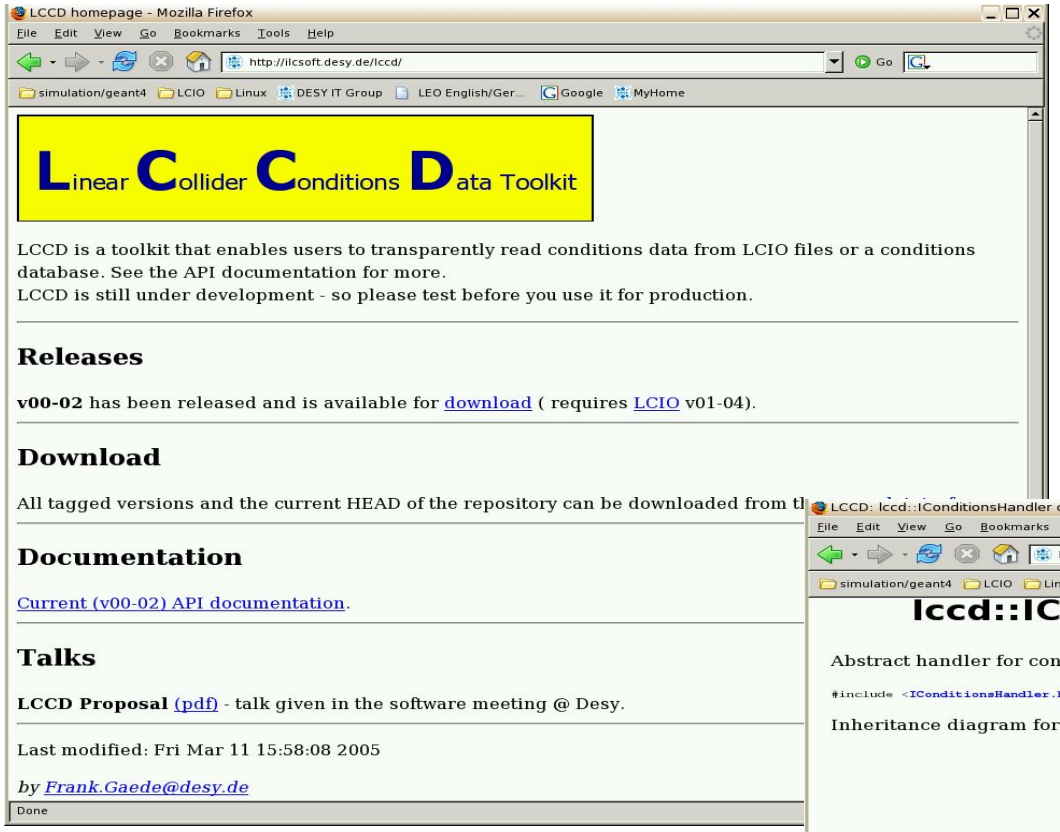
- Conditions Data:
  - all data that is needed for analysis/reconstruction besides the actual event data
    - typically has lifetime/validity range longer than one event
      - can change on various timescales, e.g. seconds to years
    - need for versioning (tagging) (changing calibration constants)
    - also 'static' geometry description (channel mapping, positions,...)

LCCD features

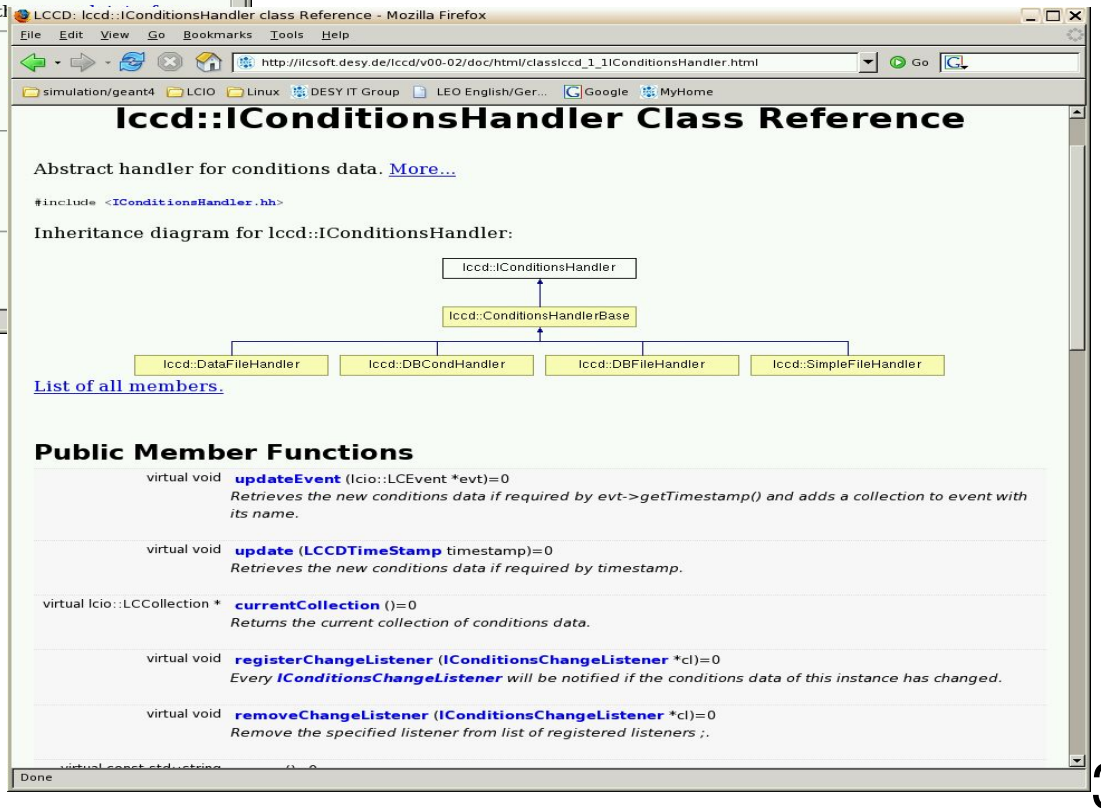
- Reading conditions data
  - from conditions database
    - for given tag
  - from simple LCIO file
    - (one set of constants)
  - from LCIO data stream
    - e.g. slow control data
  - from dedicated LCIO-DB file
    - has all constants for given tag
- Writing conditions data
  - as LCGenericObject collection
  - in folder (directory) structure
  - tagging
- Browsing the conditions database
  - through creation of LCIO files
    - vertically (all versions for timestamp)
    - horizontally (all versions for tag)



LCCD on the web

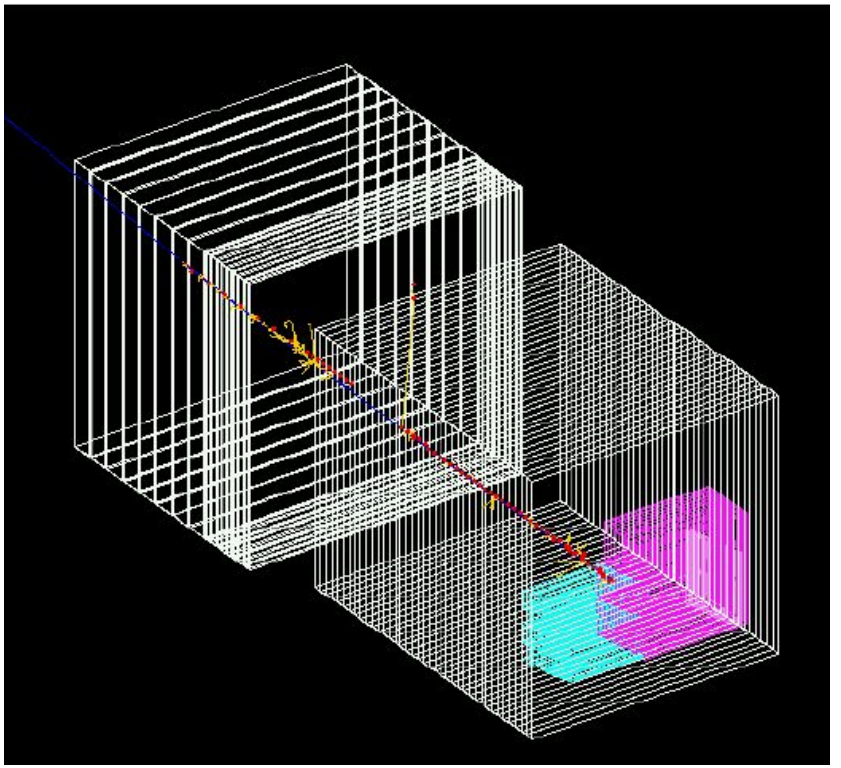


http:/ilcsoft.desy.de/lccd



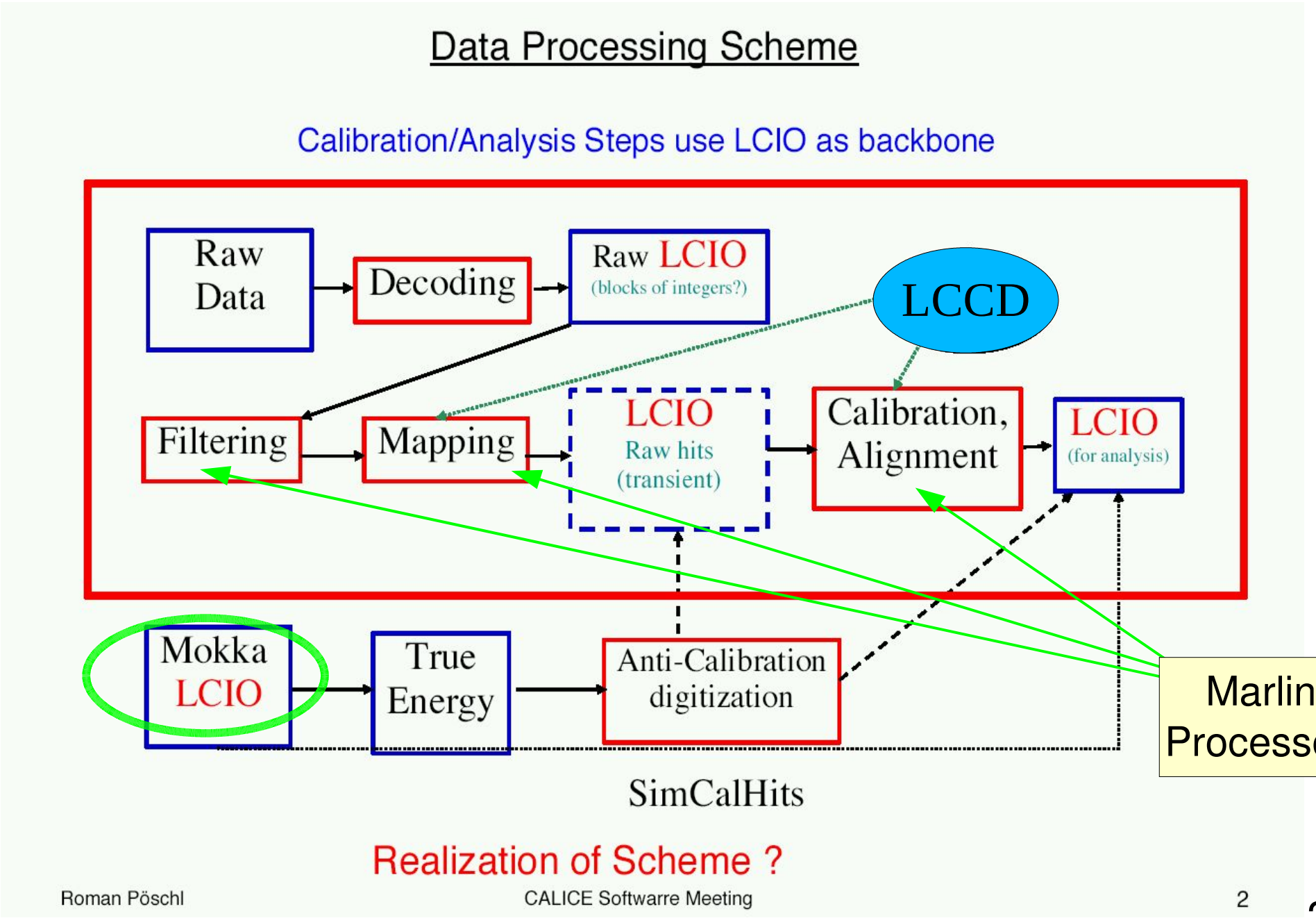
CALICE testbeam

- testbeams for CALICE prototypes (Ecal, Hcal, Tailcatcher)
  - DESY 2005:
    - e- 1-6 GeV
  - FNAL 2006/2007
    - mu,pi,p 1-100GeV
- O(10^8) events
- n*O(10^8) Monte Carlo events
- ~ 20 000 channels



- detector R&D
  - test DAQ
  - test detector concepts
- software
  - hadronic physics in Geant4
  - **test complete software chain !**

Calice testbeam software



Summary & Outlook

- ILC software development is a very active field, e.g.
  - Mokka, Marlin, LCIO, LCCD, ...
  - currently a full C++ reconstruction framework developed within Marlin
  - next topics:
    - **abstract geometry interface**
    - **breaking the Java-C++ - language barrier**

Holy grail: have an internationally used common software framework for the ILC that is used to support R&D and the detector concept studies !

There is work to do – but we are in a reasonably good shape given the ILC timescale ...