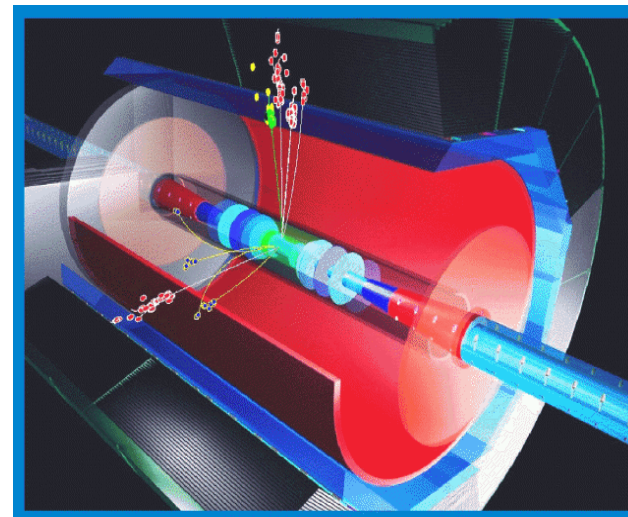


Simulation and Reconstruction Software for the ILC

Frank Gaede, DESY
ACAT05, DESY Zeuthen
May 25, 2005

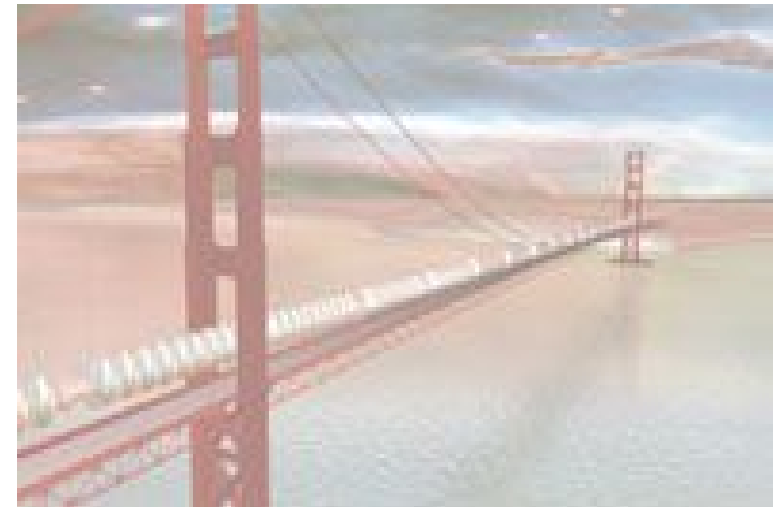
Outline

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



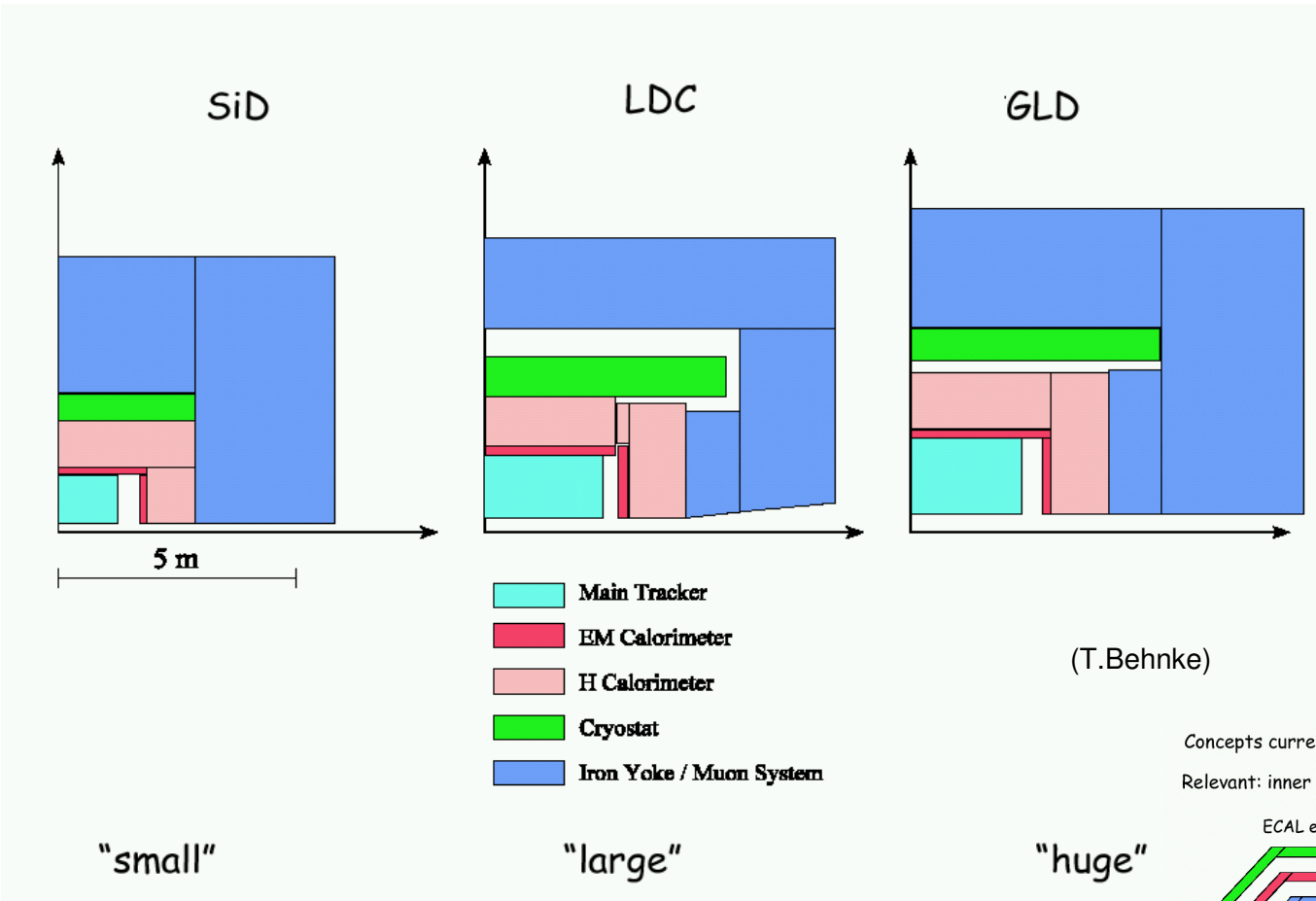
ILC Overview

- future linear e^+/e^- collider ($l \sim 30\text{km}$)
 - **super-conducting RF** technology
 - stage 1: $E=200 \rightarrow 500 \text{ GeV}$
 - stage 2: upgrade $E \sim 1\text{TeV}$
- options: gamma/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



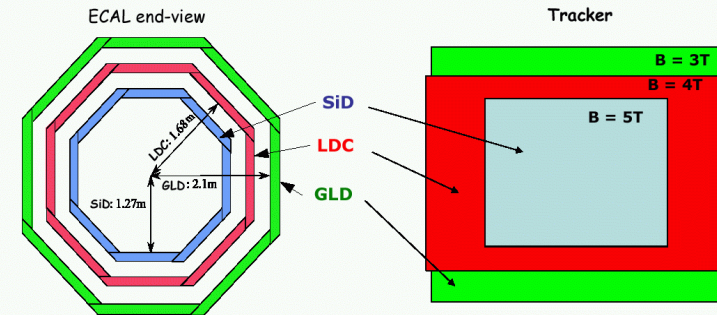
Detector Concept Study

Frank Gaede, DESY, ACAT05, DESY Zeuthen, May 25, 2005



three **interregional** detector concept studies ongoing

Concepts currently studies differ mainly in **SIZE** and **aspect ratio**
 Relevant: inner radius of ECAL: defines the overall scale



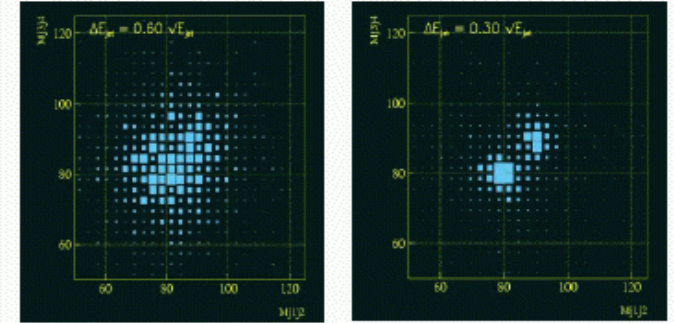
SiD: Silicon based concept GLD: even larger detector concept
 LDC: large detector concept

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)

WW-ZZ separation



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):

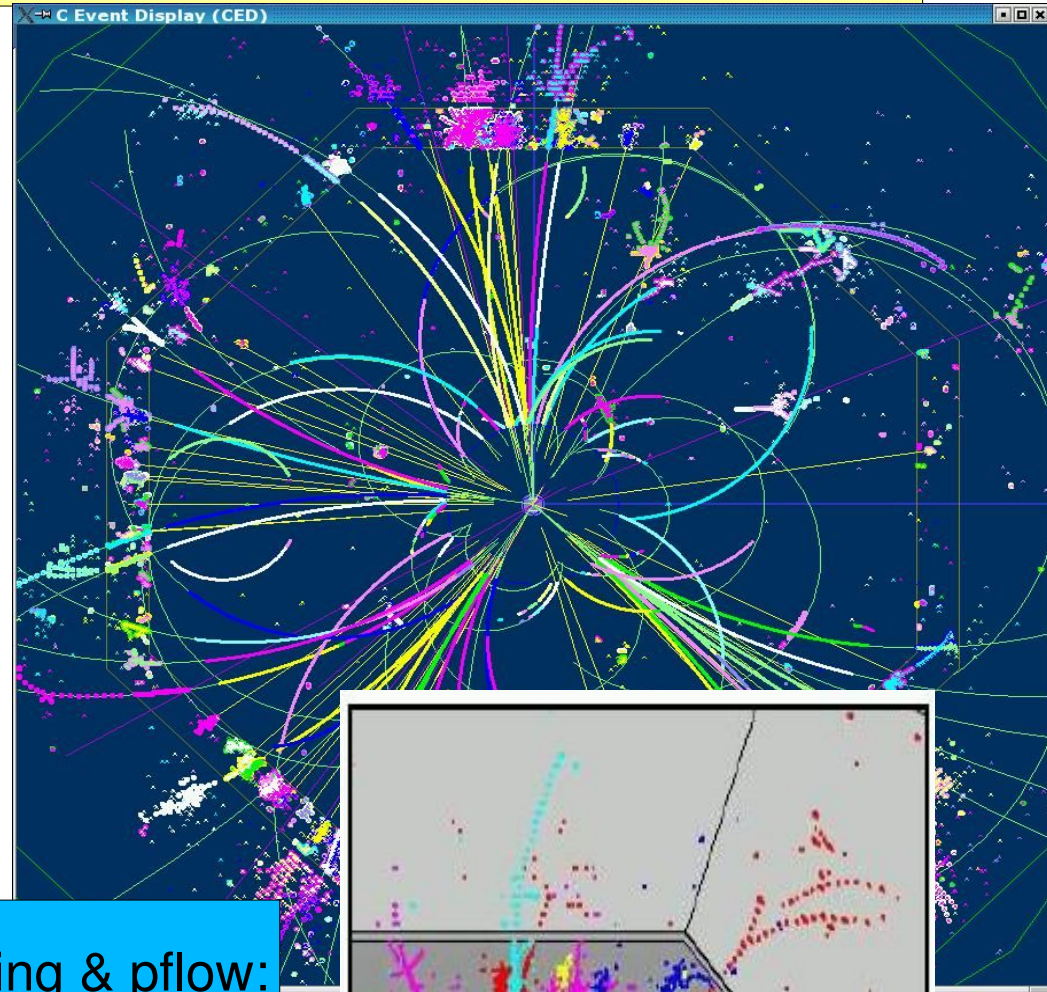
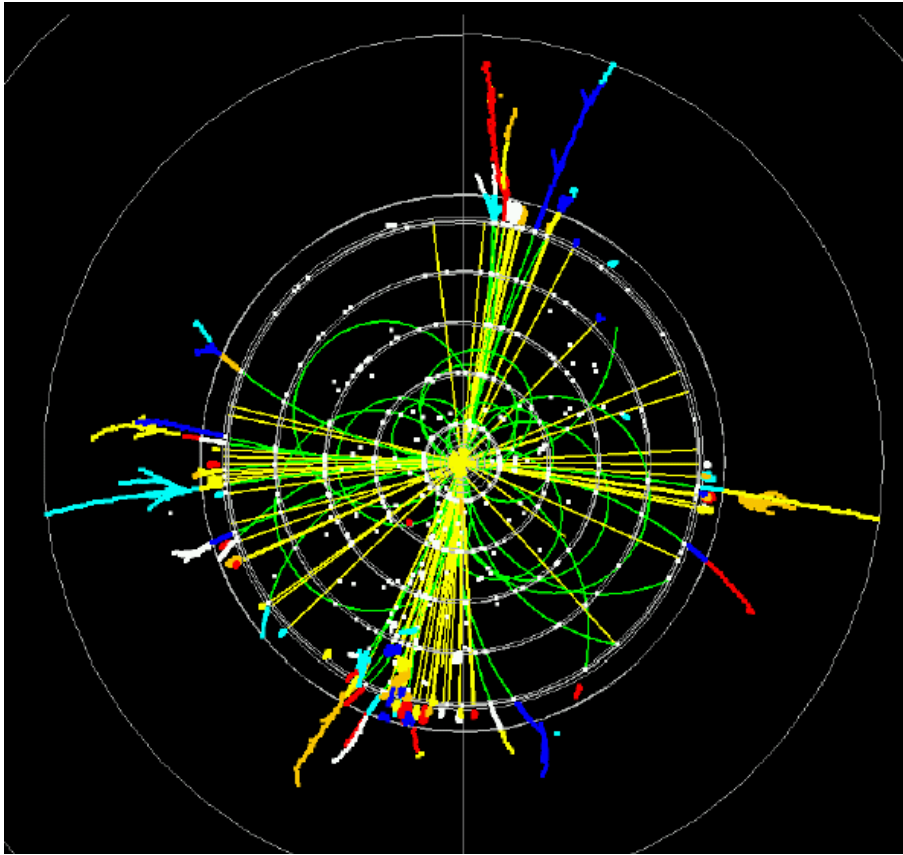
- Hcal resolution
- confusion term

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

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

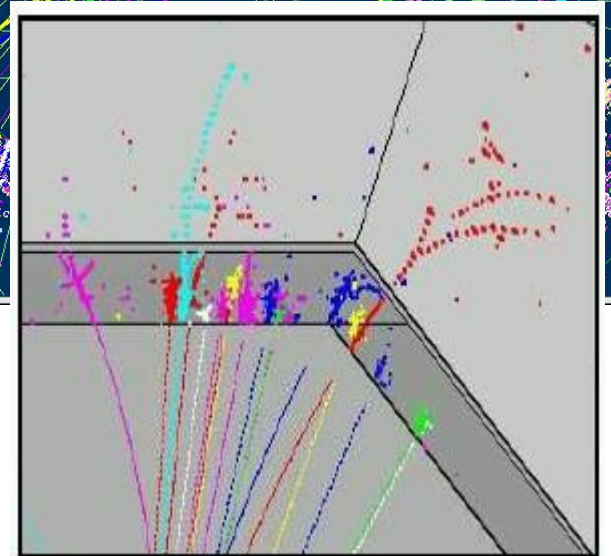
Imaging calorimeter & ParticleFlow

Frank Gaede, DESY, ACAT05, DESY Zeuthen, May 25, 2005



Need software tools to improve clustering & pflow:

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



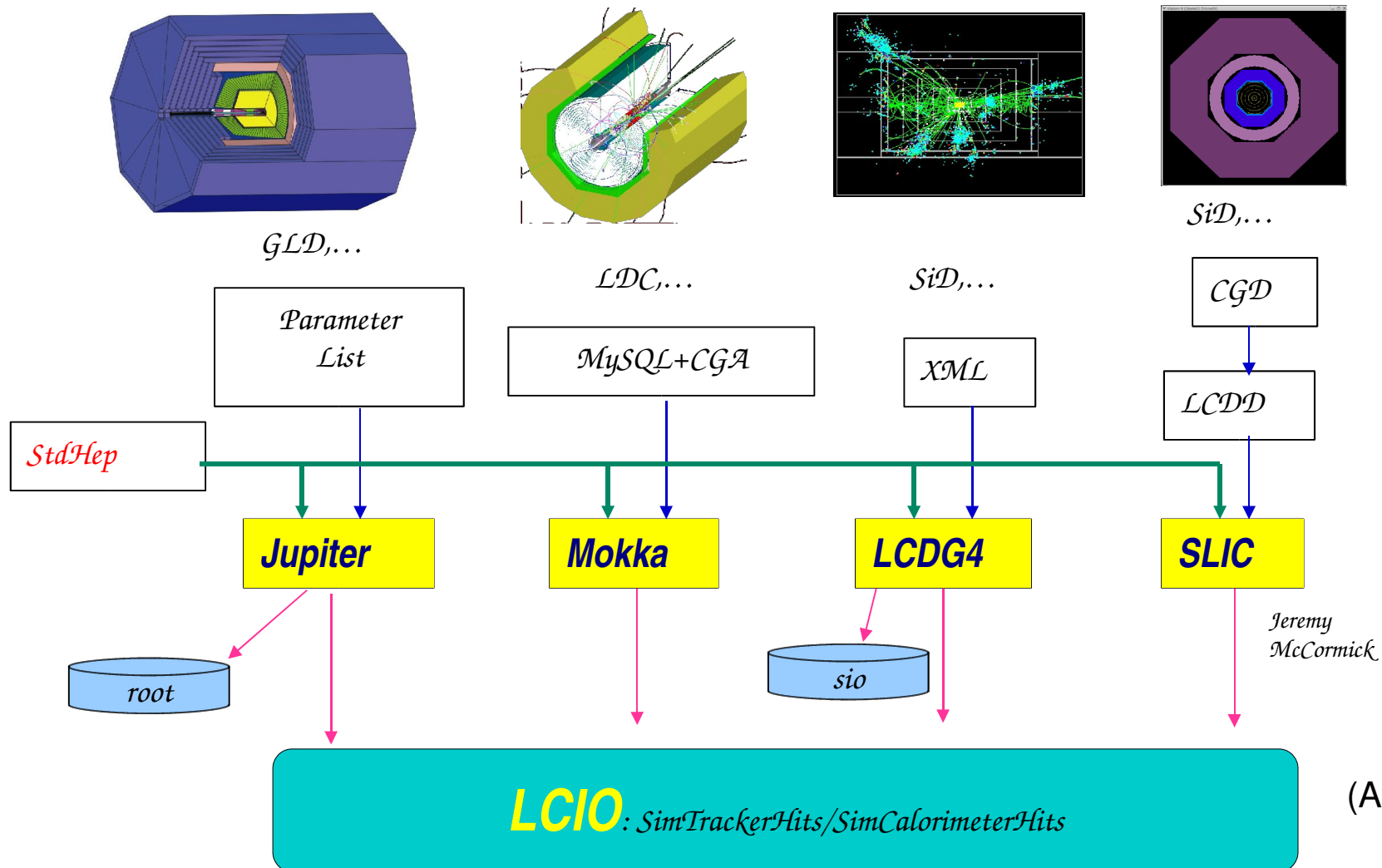
Note: precision tracking and vertexing also very challenging and important !

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*



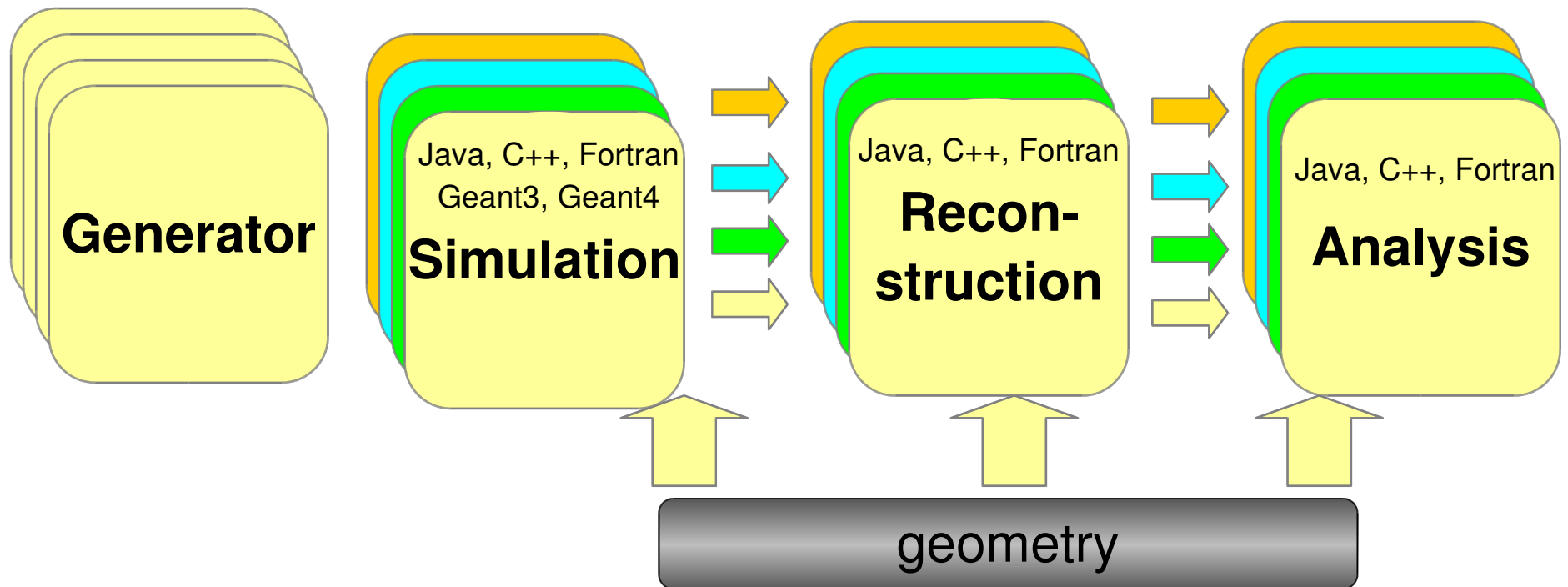
(A.Miyamoto)

A Common Software Framework for the ILC ?

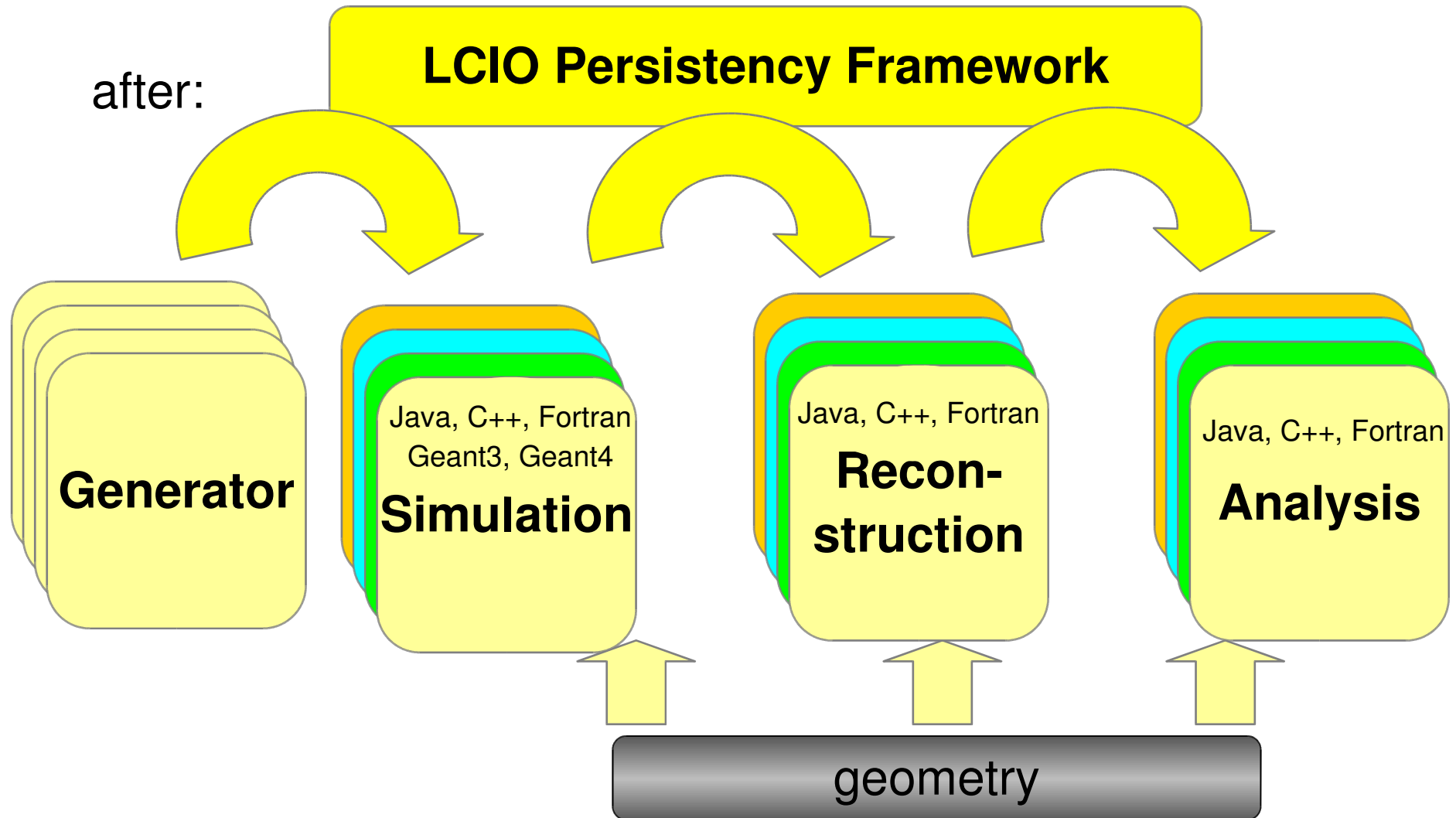
- a common software framework for the ILC that is flexible and easy to use would be highly desirable !
- **but:**
 - ILC emerged out of three regional studies
 - all groups have developed their own software as needed for the R&D
 - different languages used: C++, Java, f77
 - ...
- **aim:**
 - develop 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

data model

contents

data handling

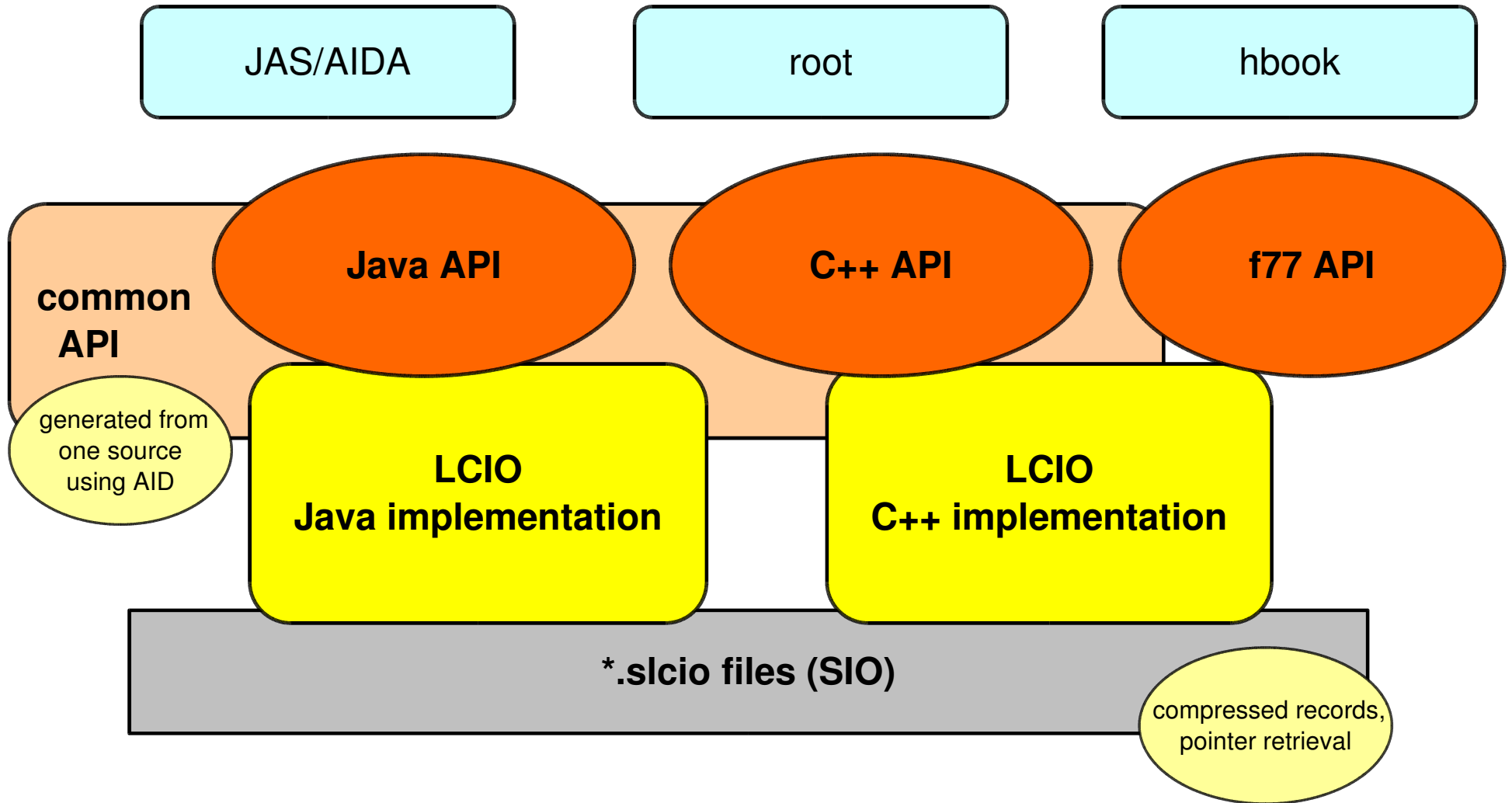
API

implementation

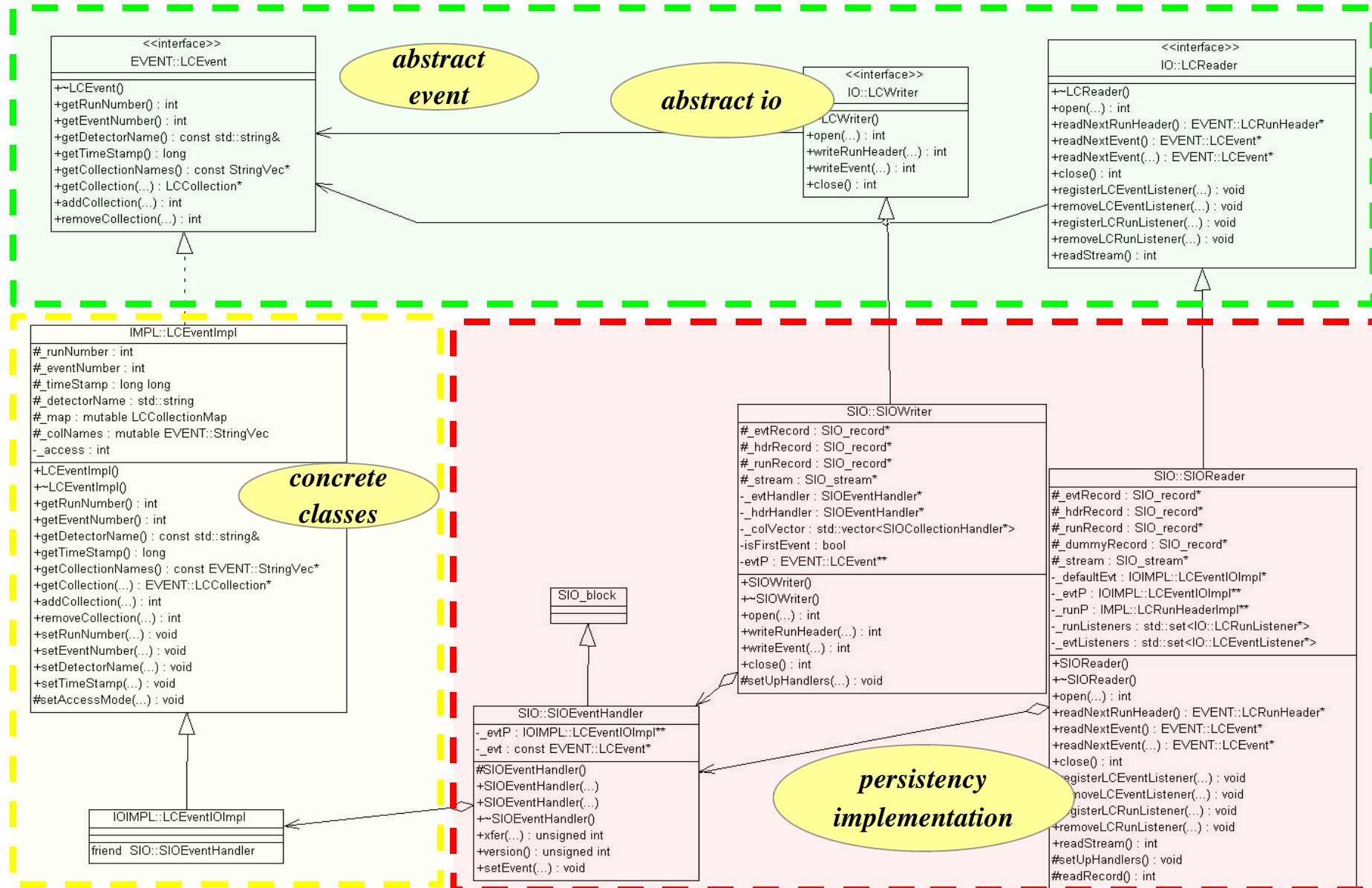
data format

persistence

LCIO SW-Architecture

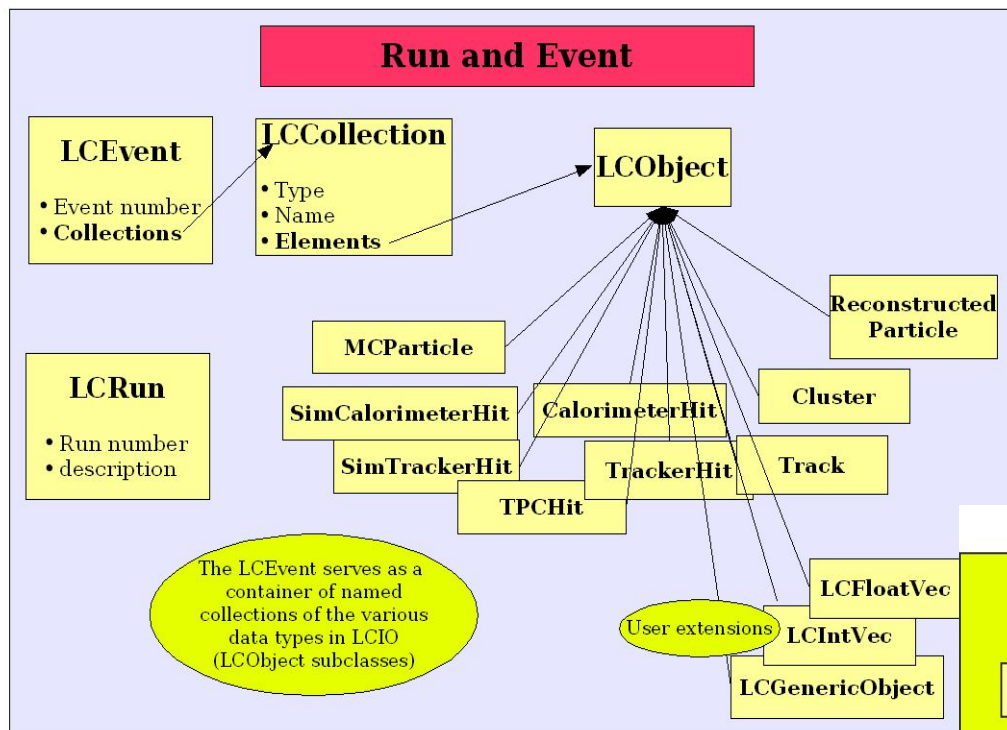


LCIO class design



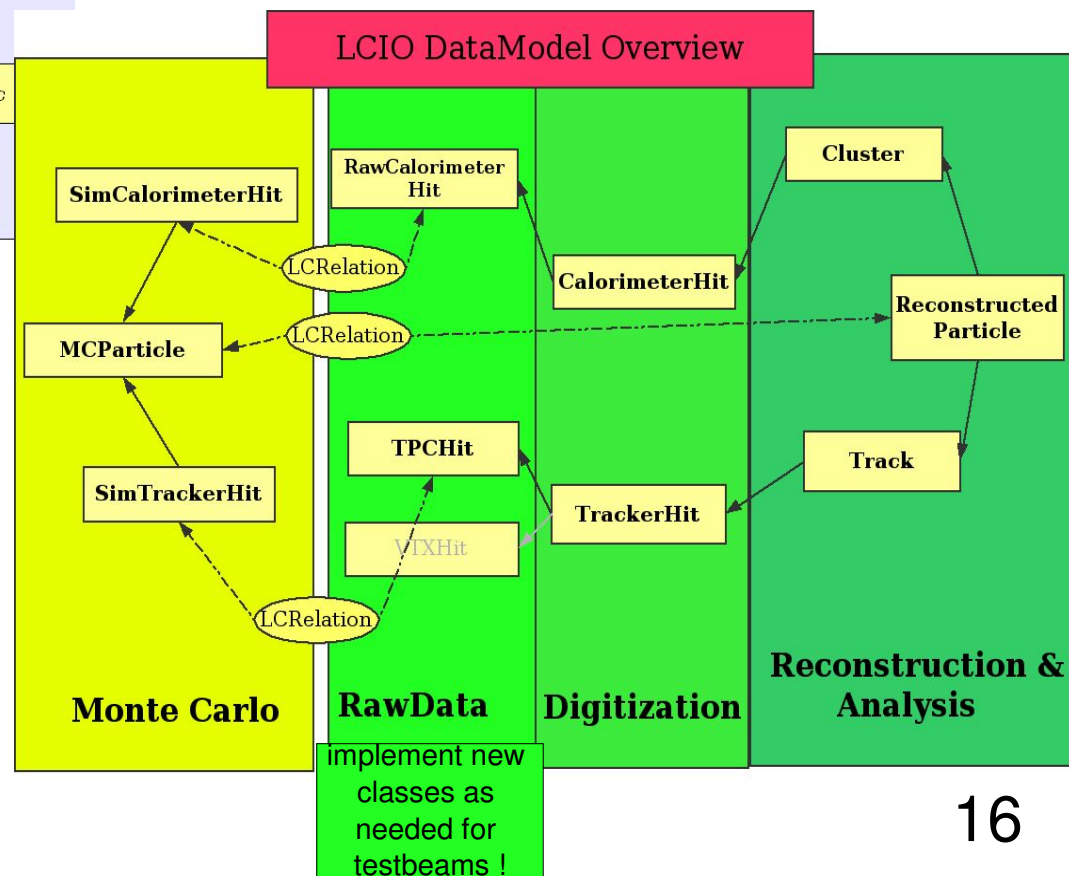
LCIO data model

Frank Gaede, DESY, ACAT05, DESY Zeuthen, May 25, 2005



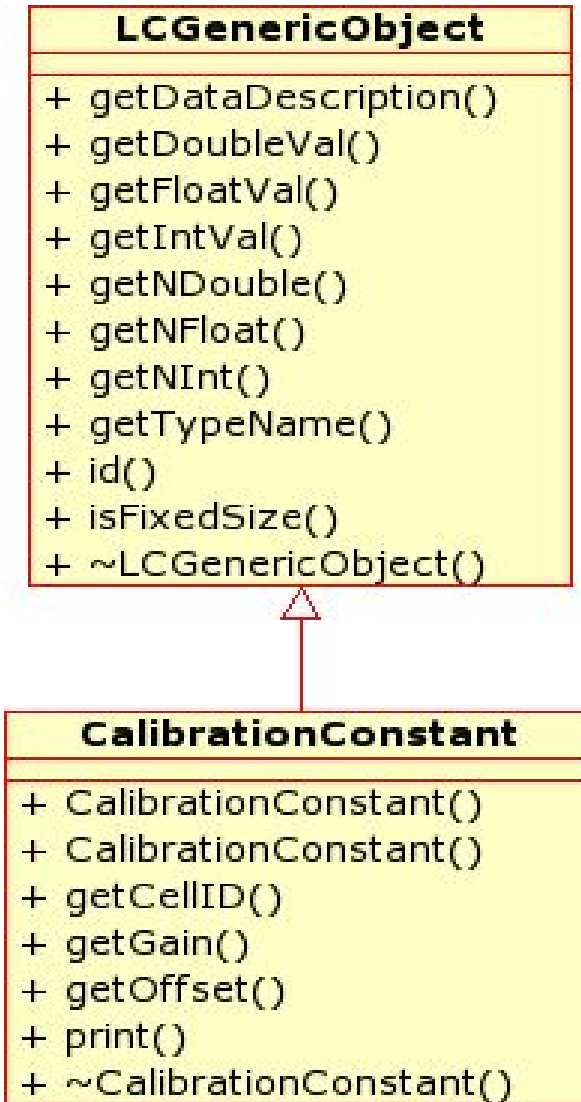
event serves as container of untyped collections

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

Frank Gaede, DESY, ACAT05, DESY Zeuthen, May 25, 2005

Overview (LCIO API Documentation, Version v01-03) - Mozilla Firefox

LCIO API
Version v01-03

All Classes

Packages
[hep.lcio.data](#)
[hep.lcio.event](#)

All Classes
[AnalysisJob](#)
[CalorimeterHit](#)
[Cluster](#)
[DataNotAvailableE](#)
[EventException](#)
[ICalorimeterHit](#)
[ICluster](#)
[ILCCollection](#)
[ILCEvent](#)
[ILCFactory](#)
[ILCFloatVec](#)
[ILCIntVec](#)
[ILCParameters](#)
[ILCRelation](#)
[ILCRunHeader](#)
[ILCStrVec](#)
[IMCParticle](#)
[IParticleID](#)

LCIO - a persistency framework for linear collider simulation studies.

See:

Description

Packages

hep.lcio.data	The package hep.lcio.data has been removed - interfaces are now all defined in hep.lcio.event.
hep.lcio.event	Primary user interface for LCIO.
hep.lcio.example	Simple usage examples.
hep.lcio.exceptions	Exceptions thrown by LCIO.
hep.lcio.implementation.event	
hep.lcio.implementation.io	Default IO implementation.
hep.lcio.implementation.sio	SIO specific LCIO implementation.
hep.lcio.io	Interfaces for IO library.
hep.lcio.test	
hep.lcio.util	Utilities for use with LCIO.

Documentation for LCIO v01-03

- [Users manual](#) (also available as [pdf](#) and [ps](#)) **Read before you get st**
- [Java API documentation](#)
- [C++ API documentation](#)
- (printable version of the C++ API reference: [lciorefman.ps](#))
- XML data format description: [lcio.xml](#)

Last modified: Thu Sep 23 14:51:51 2004

LCIO - Users manual v01-03 - Mozilla Firefox

Building the library

A few variables have to be set depending on your development environment, e.g.

- Linux (and bash):

```
export LCIO=/lcio/v01-03          <-- modify as appropriate
export PATH=$LCIO/tools:$PATH

export JDK_HOME=/usr/lib/j2sdk   <-- modify as appropriate
```
- Windows/Cygwin - DOS shell:

```
set PATH=c:/cygwin/bin;%PATH%   <-- modify as appropriate

set LCIO=c:/lcio/develop/v01-03 <-- modify as appropriate
set JDK_HOME=c:/j2sdk1.4.1_01   <-- modify as appropriate

set PATH=%LCIO%/tools;%LCIO%/bin;%PATH%
```

IMPL::ReconstructedParticleImpl
Class Reference

Implementation of ReconstructedParticle. [More...](#)

#include [<ReconstructedParticleImpl.h>](#)

Inheritance diagram for IMPL::ReconstructedParticleImpl:

```
graph TD
    EVENT_LCObject[EVENT::LCObject] --> IMPL_ReconstructedParticleImpl[IMPL::ReconstructedParticleImpl]
    EVENT_ReconstructedParticle[EVENT::ReconstructedParticle] --> IMPL_ReconstructedParticleImpl
    IMPL_AccessChecked[IMPL::AccessChecked] --> IMPL_ReconstructedParticleImpl
```

ReconstructedParticleImpl ()
Default constructor, initializes values to 0.

virtual ~ReconstructedParticleImpl ()

LCIO XML Data Format Description

Find: desy

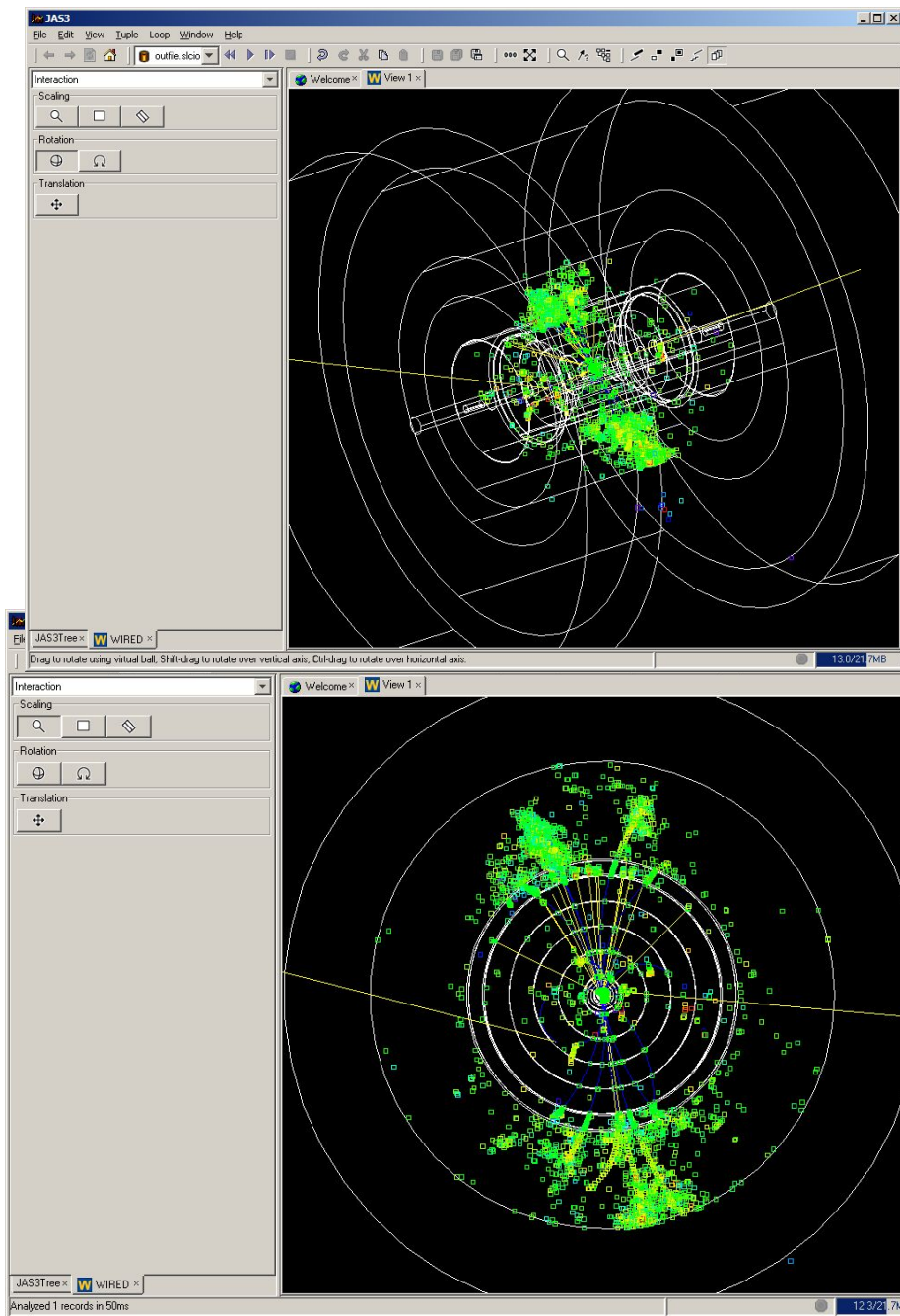
```
<!-- end of named parameters -->
<data type="int" name="nTrack"/>
<!-- end of named parameters -->
<repeat count="nTrack">
  <data type="int" name="type">Type of Track, e.g. TPC, VTX, SIT</data>
  <data type="float" name="d0">Impact Parameter in r-phi</data>
  <data type="float" name="phi">Phi of track in r-phi</data>
  <data type="float" name="omega">curvature signed with charge</data>
  <data type="float" name="z0">Impact Parameter in r-z</data>
  <data type="float" name="tanLambd">tangent of dip angle in r-z</data>
  <data type="float[15]" name="covMatrix"> Covariance matrix</data>
  <data type="float[3]" name="referencePoint">Reference point (x,y,z)</data>
  <data type="float" name="chi2">chi**2 of fit</data>
</repeat>
</file>
```

home: <http://lcio.desy.de>
forum: <http://forum.linearcollider.org>
bugs: <http://bugs.freehep.org>

JAS3 and LCIO

Frank Gaede, DESY, ACAT05, DESY Zeuthen, May 25, 2005

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



Collection: MCParticle type:MCParticle size:473 flags:0

N	Type	Status	Parent	PX	PY	PZ	Mass
0	2212	Document...		0	0	7000.0	0.93827
1	2212	Document...		0	0	-7000.0	0.93827
2	21	Document...	0	0.25815	-0.27900	6.5793	0
3	-3	Document...	1	-0.45454	-0.36117	-1802.7	0
4	4	Document...	2	-0.40964	-1.0530	2.2164	0
5	-3	Document...	3	-13.179	1.9646	-717.51	0
6	22	Document...	4,5	0.78672	0.69178	-4.4768	0
7	24	Document...	4,5	-14.375	0.21979	-710.81	80.667
8	22	Final State	6	0.78672	0.69178	-4.4768	0
9	24	Intermediate	7	-14.375	0.21979	-710.81	80.667
10	3224	Intermediate	1	0.16978	0.20640	-1483.5	1.3846
11	-4	Intermediate	2	1.0287	0.84333	2.4188	1.3500
12	2	Intermediate	0	0.080131	0.087964	0.31987	5.6000E-3
13	-3	Intermediate	9	-11.920	16.413	-260.20	0.19900
14	21	Intermediate	9	-9.7052	16.270	-246.29	0
15	21	Intermediate	9	-0.18941	-0.12814	-6.3494	0
16	21	Intermediate	9	-0.47022	-0.21941	-2.9564	0
17	21	Intermediate	9	0.41252	0.36534	-2.3612	0
18	21	Intermediate	9	-0.11239	-0.075933	0.055171	0
19	21	Intermediate	9	1.3372	-4.4404	-32.038	0
20	4	Intermediate	9	6.2717	-27.965	-160.67	1.3500
21	2	Intermediate		-3.5848	-3.3256	730.00	0
22	-2	Intermediate		3.5848	3.3256	-35.384	0
23	1	Intermediate		-2.7119	2.7973	2.4939	0

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, flexi	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

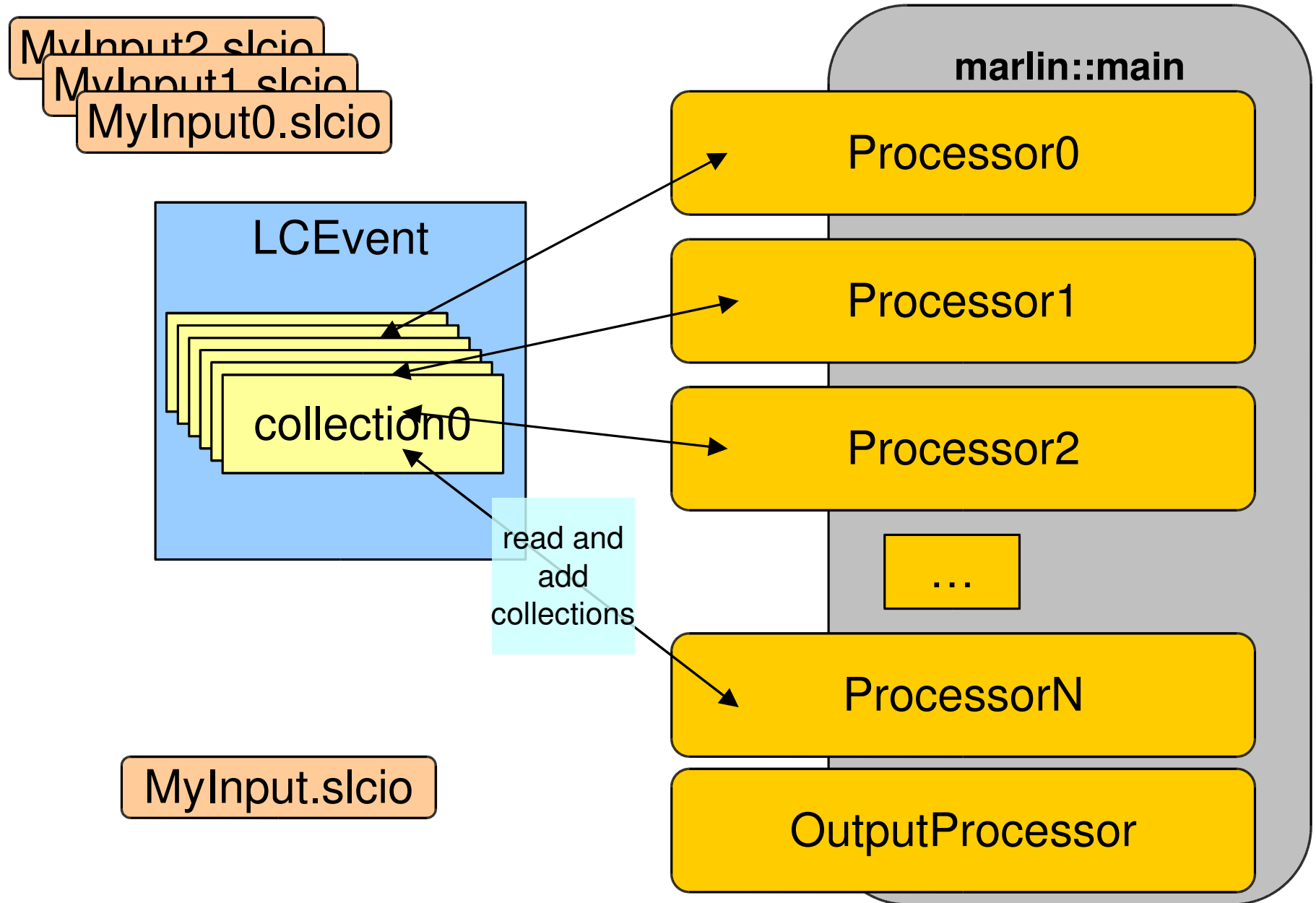
Marlin - Introduction

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

Frank Gaede, DESY, ACAT05, DESY Zeuthen, May 25, 2005

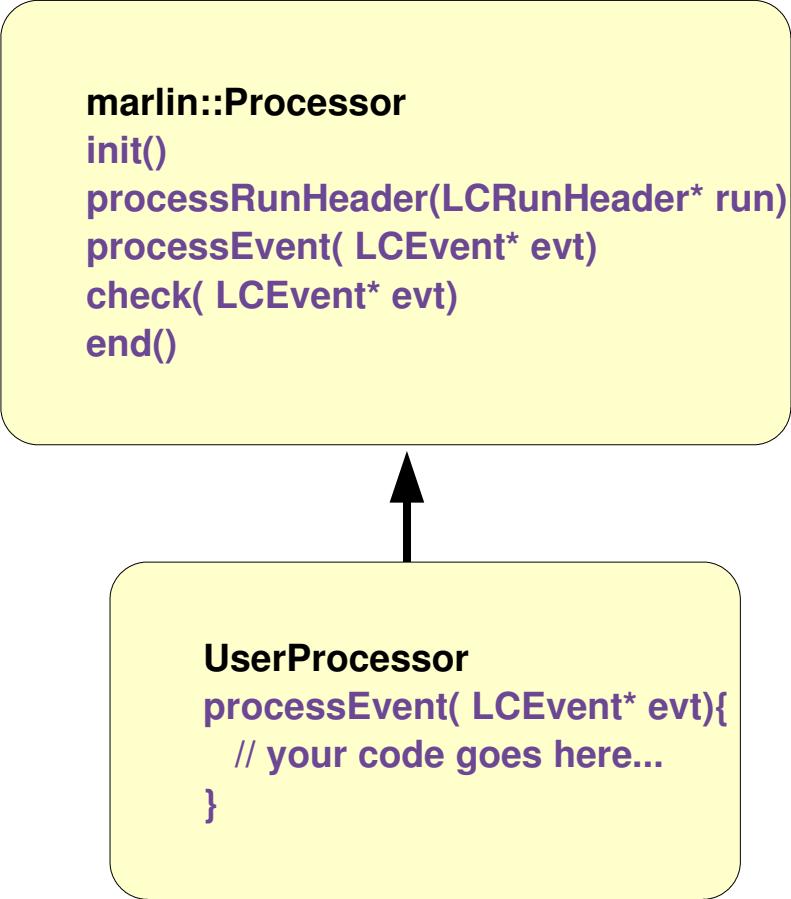


Marlin Processor

- 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::Processor
init()
processRunHeader(LCRunHeader* run)
processEvent( LCEvent* evt)
check( LCEvent* evt)
end()
```

```
UserProcessor
processEvent( LCEvent* evt){
    // your code goes here...
}
```

A diagram showing two yellow rounded rectangular boxes. The top box contains the code for marlin::Processor. The bottom box contains the code for UserProcessor. A black arrow points from the top of the UserProcessor box to the bottom of the marlin::Processor box, indicating that UserProcessor inherits from marlin::Processor.

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

Frank Gaede, DESY, ACAT05, DESY Zeuthen, May 25, 2005



Releases

v00-08 has been released and is available for [download](#).
Marlin can now optionally be linked against [LCCD](#) to provide easy access to conditions data.
[documentation](#) has been improved.

Download

All tagged versions and the current HEAD of the repository can be downloaded from the [CVS](#).

Documentation

[Current API documentation](#).

Talks

LCIO & Marlin ([pdf](#)) - talk given at the DESY Simulation WS 2004.

Last modified: Fri Mar 11 16:01:59 2005

by Frank.Gaede@desy.de

Done

<http://ilcsoft.desy.de/marlin>

Marlin (v00-08)

Marlin [Modular Analysis and Reconstruction for the LINear collider] is a simple modular application framework for analysis and reconstruction code based on LCIO.

Overview

The main purpose of Marlin is to facilitate the modular development of reconstruction and analysis code based on LCIO. As a lot of different groups are involved it should be simple and straight forward to have distributed development of modules and combine existing modules as needed in a larger application.

The base class for a Marlin module is called `marlin::Processor`. It defines a set of callbacks that the user can implement in their subclasses. A steering file mechanism allows to activate the needed processors. These are then called for every event using the LCEvent as container for input and output data in terms of LCCollections:

Installation

The installation of Marlin is described in the [README](#).

Running Marlin

After having installed Marlin you have to write your own `marlin::Processor(s)` subclass that performs the computation. This is fairly straight forward and Marlin provides an example in [/examples/mymarlin](#) that can serve as a template for your own projects.

Note: there is no need to write a main program as this is provided by Marlin. Existing Processors are automatically registered with Marlin provided one instance exists in the library as described in the [README](#).

Steering files

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

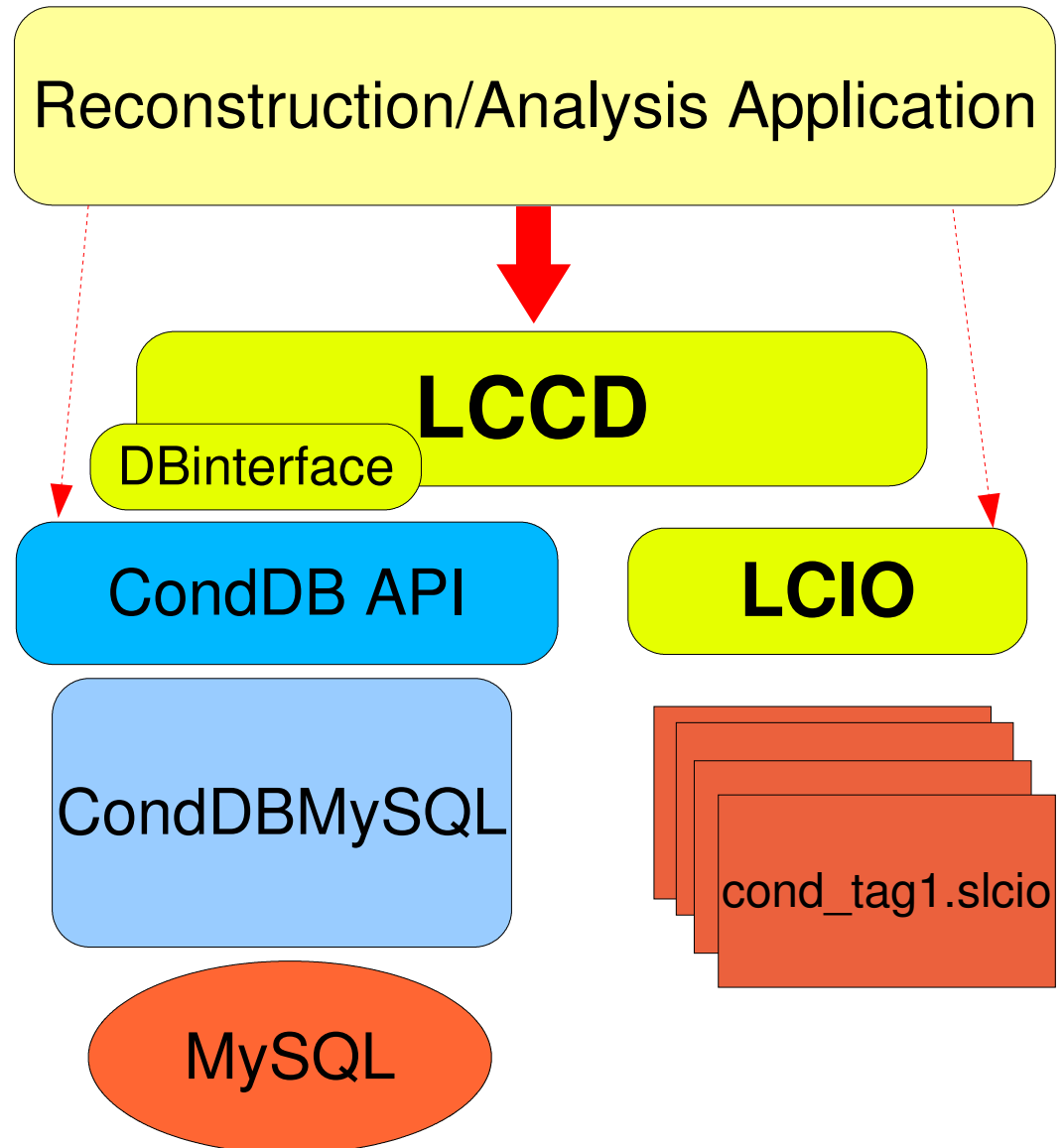
LCCD

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

Frank Gaede, DESY, SLAC-Simulation-Meeting March 16/17 2005

LCCD homepage - Mozilla Firefox

File Edit View Go Bookmarks Tools Help

http://ilcsoft.desy.de/lccd/

simulation/geant4 LCIO Linux DESY IT Group LEO English/Ger... Google MyHome

Linear Collider Conditions Data Toolkit

LCCD is a toolkit that enables users to transparently read conditions data from LCIO files or a conditions database. See the API documentation for more.
LCCD is still under development - so please test before you use it for production.

Releases

v00-02 has been released and is available for [download](#) (requires [LCIO](#) v01-04).

Download

All tagged versions and the current HEAD of the repository can be downloaded from [here](#).

Documentation

[Current \(v00-02\) API documentation.](#)

Talks

[LCCD Proposal \(pdf\)](#) - talk given in the software meeting @ Desy.

Last modified: Fri Mar 11 15:58:08 2005
by Frank.Gaede@desy.de

Done

<http://ilcsoft.desy.de/lccd>

LCCD: IConditionsHandler class Reference - Mozilla Firefox

File Edit View Go Bookmarks Tools Help

http://ilcsoft.desy.de/lccd/v00-02/doc/html/classlccd_1_IConditionsHandler.html

simulation/geant4 LCIO Linux DESY IT Group LEO English/Ger... Google MyHome

Iccd::IConditionsHandler Class Reference

Abstract handler for conditions data. [More...](#)

```
#include <IConditionsHandler.hh>
```

Inheritance diagram for Iccd::IConditionsHandler:

```
graph TD
    IConditionsHandler[Iccd::IConditionsHandler]
    ConditionsHandlerBase[Iccd::ConditionsHandlerBase]
    DataFileHandler[Iccd::DataFileHandler]
    DBCondHandler[Iccd::DBCondHandler]
    DBFileHandler[Iccd::DBFileHandler]
    SimpleFileHandler[Iccd::SimpleFileHandler]

    ConditionsHandlerBase --> IConditionsHandler
    DataFileHandler --> ConditionsHandlerBase
    DBCondHandler --> ConditionsHandlerBase
    DBFileHandler --> ConditionsHandlerBase
    SimpleFileHandler --> ConditionsHandlerBase
```

[List of all members.](#)

Public Member Functions

virtual void **updateEvent** (IcIo::LCEvent *evt)=0
Retrieves the new conditions data if required by `evt->getTimestamp()` and adds a collection to event with its name.

virtual void **update** (LCCDTimeStamp timestamp)=0
Retrieves the new conditions data if required by timestamp.

virtual IcIo::LCCollection * **currentCollection** ()=0
Returns the current collection of conditions data.

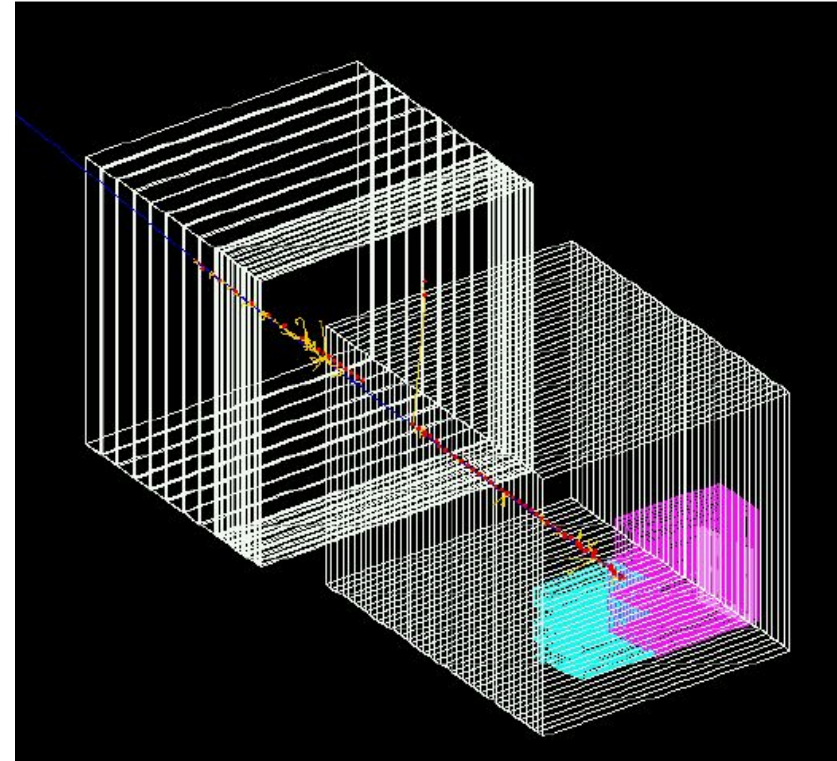
virtual void **registerChangeListener** (IConditionsChangeListener *cl)=0
Every **IConditionsChangeListener** will be notified if the conditions data of this instance has changed.

virtual void **removeChangeListener** (IConditionsChangeListener *cl)=0
Remove the specified listener from list of registered listeners ;.

Done

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
- $n \cdot O(10^8)$ Monte Carlo events
- $\sim 20\,000$ channels

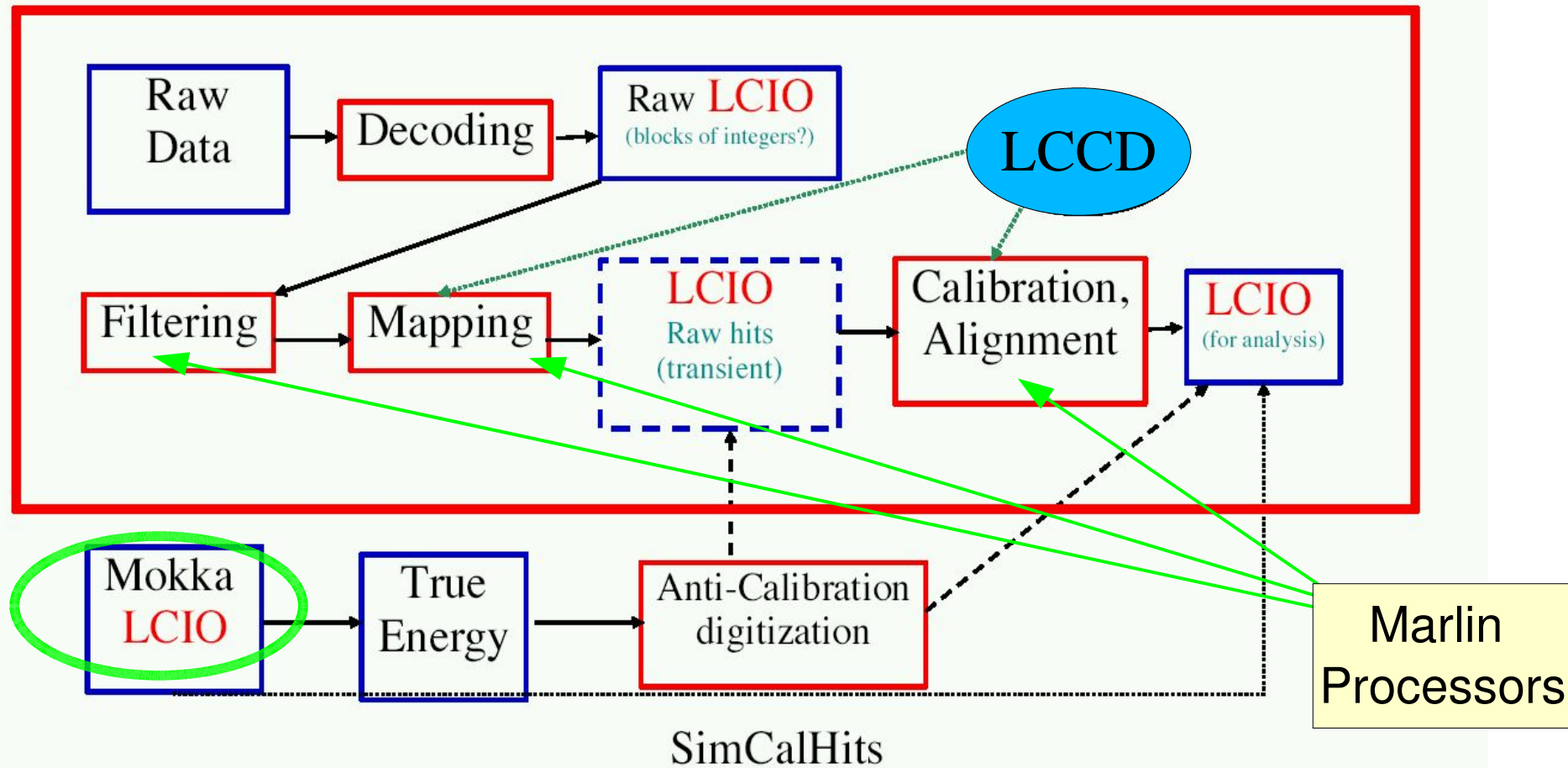


- detector R&D
 - test DAQ
 - test detector concepts
- software
 - hadronic physics in Geant4
 - **test complete software chain !**

Calice testbeam software

Data Processing Scheme

Calibration/Analysis Steps use LCIO as backbone



Realization of Scheme ?

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 ...