LCIO
Overview and Status

ECFA Workshop 2004, Durham

Simulation, Sep. 3rd, 2004

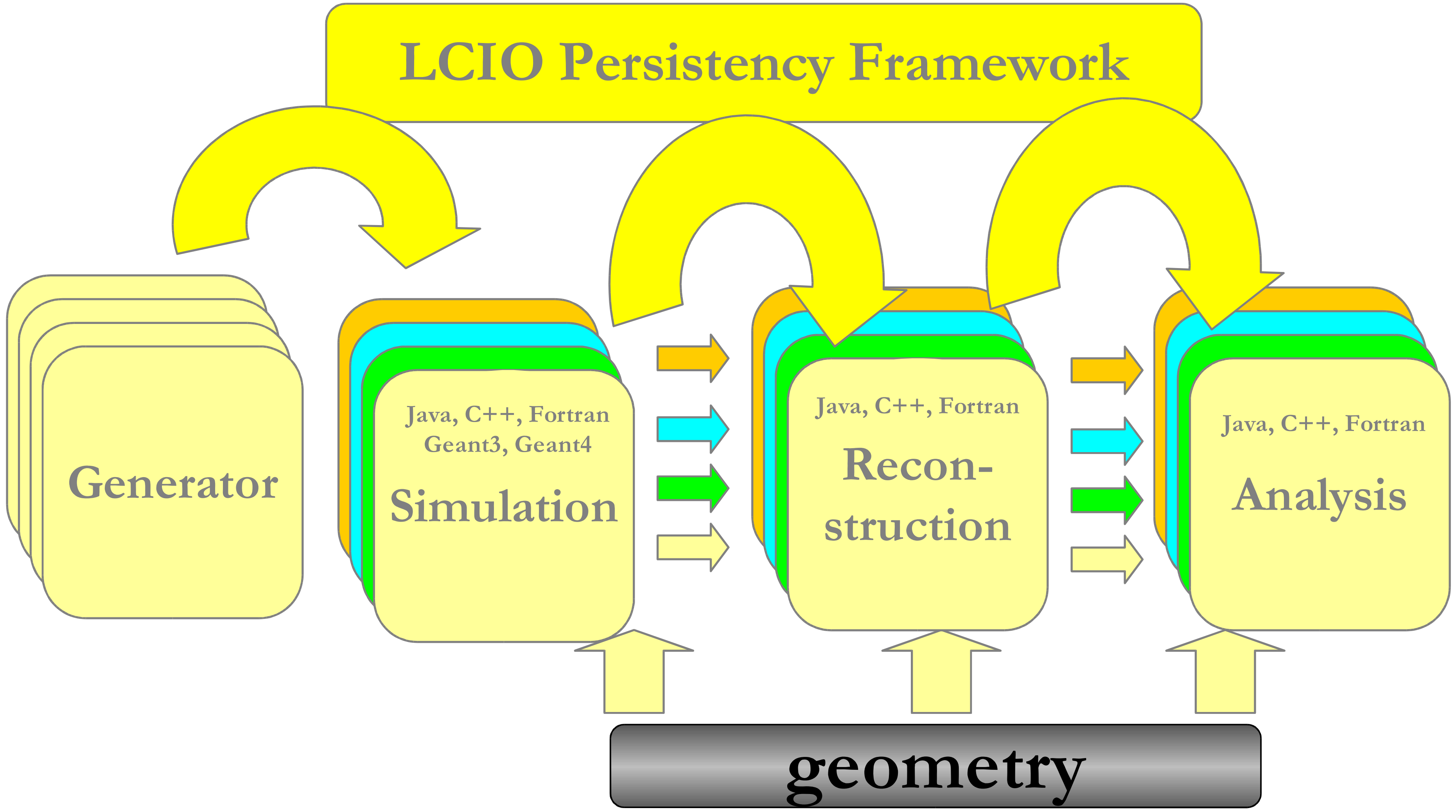
Frank Gaede DESY -IT-

reported by Ties Behnke, DESY

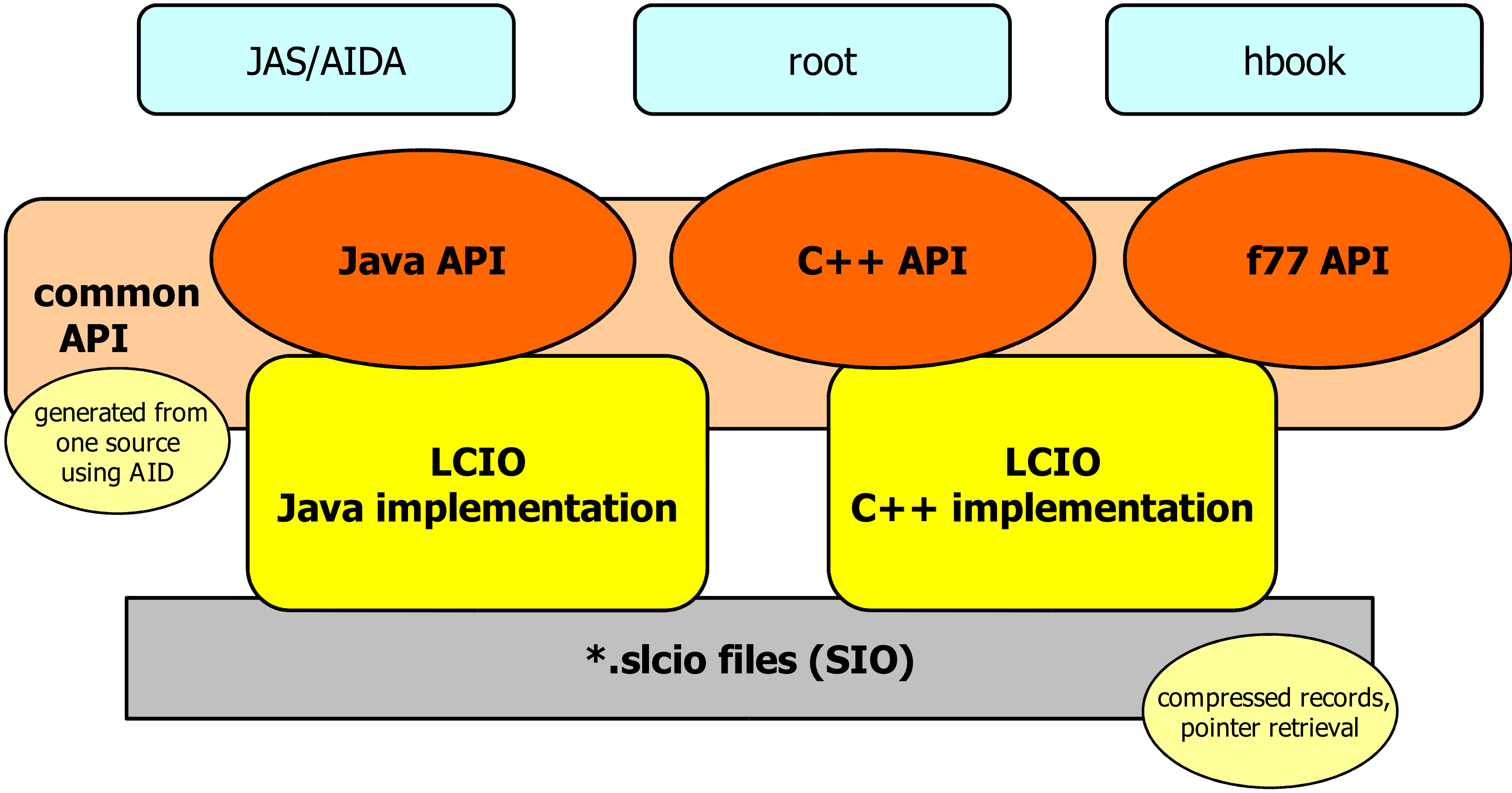
Outline

- Introduction
- Overview
- Changes since Paris Workshop
- Status
- Reconstruction/Analysis Framework
- Summary

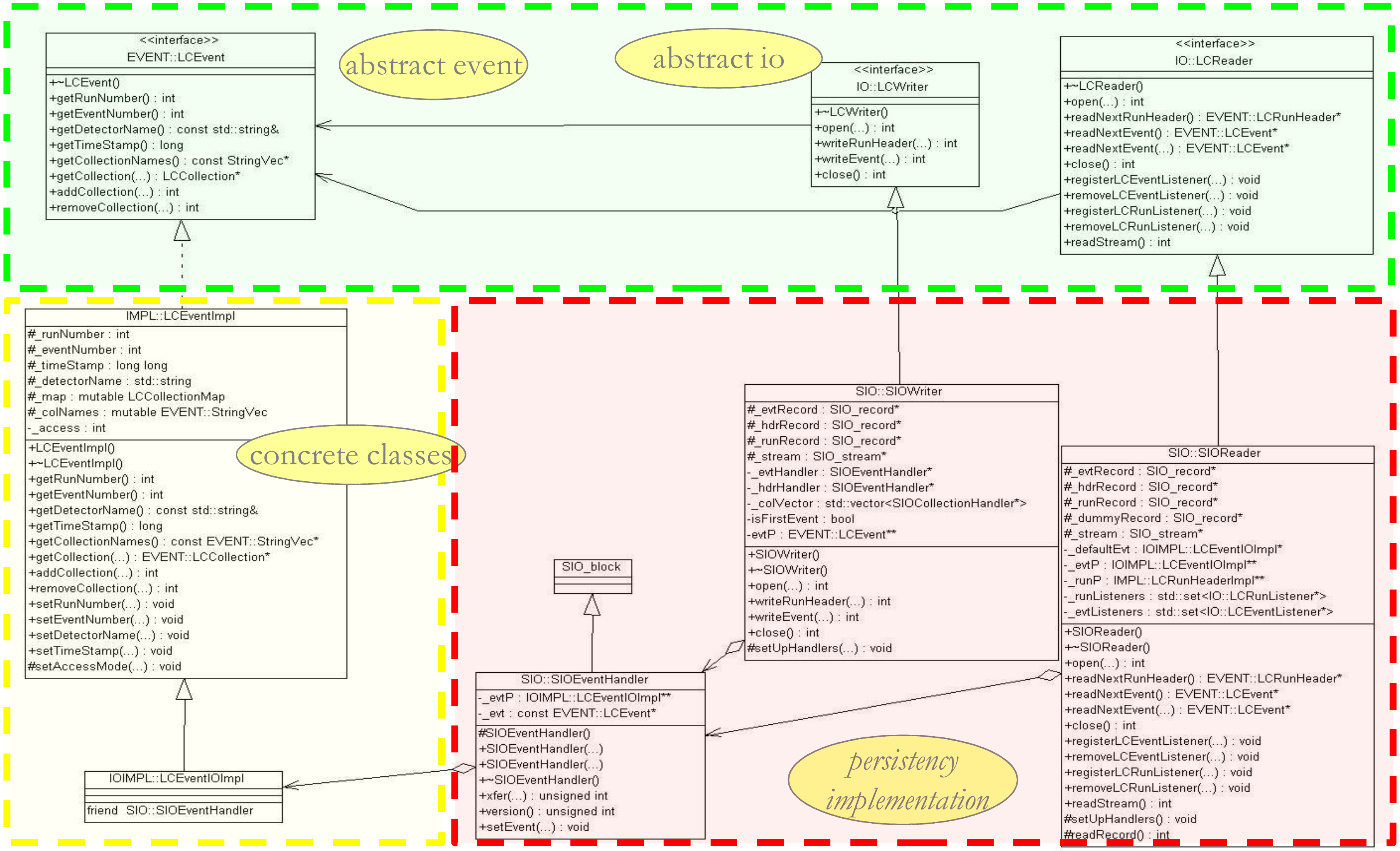
Motivation for LCIO



LCIO SW-Architecture

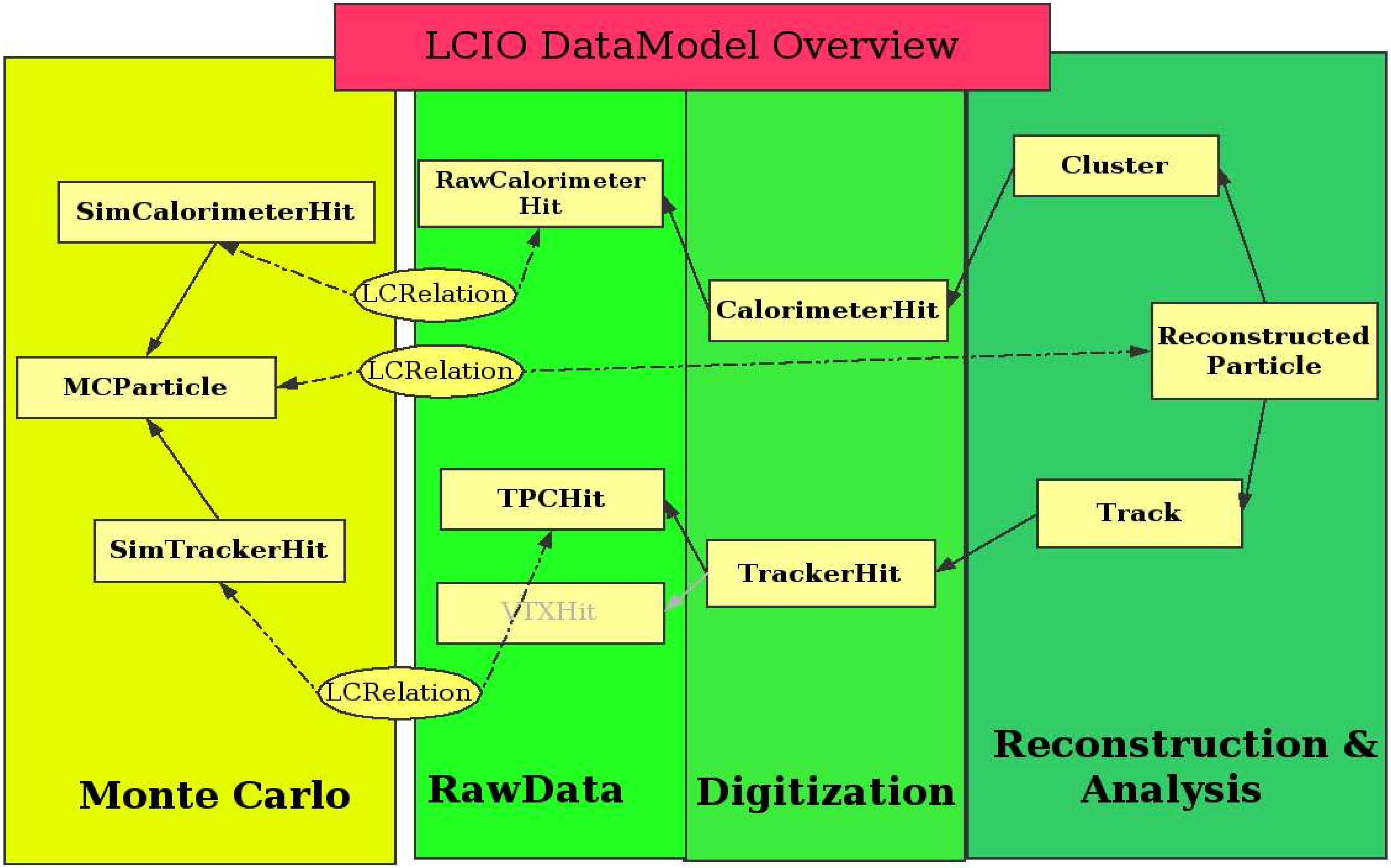


Implementation - Design

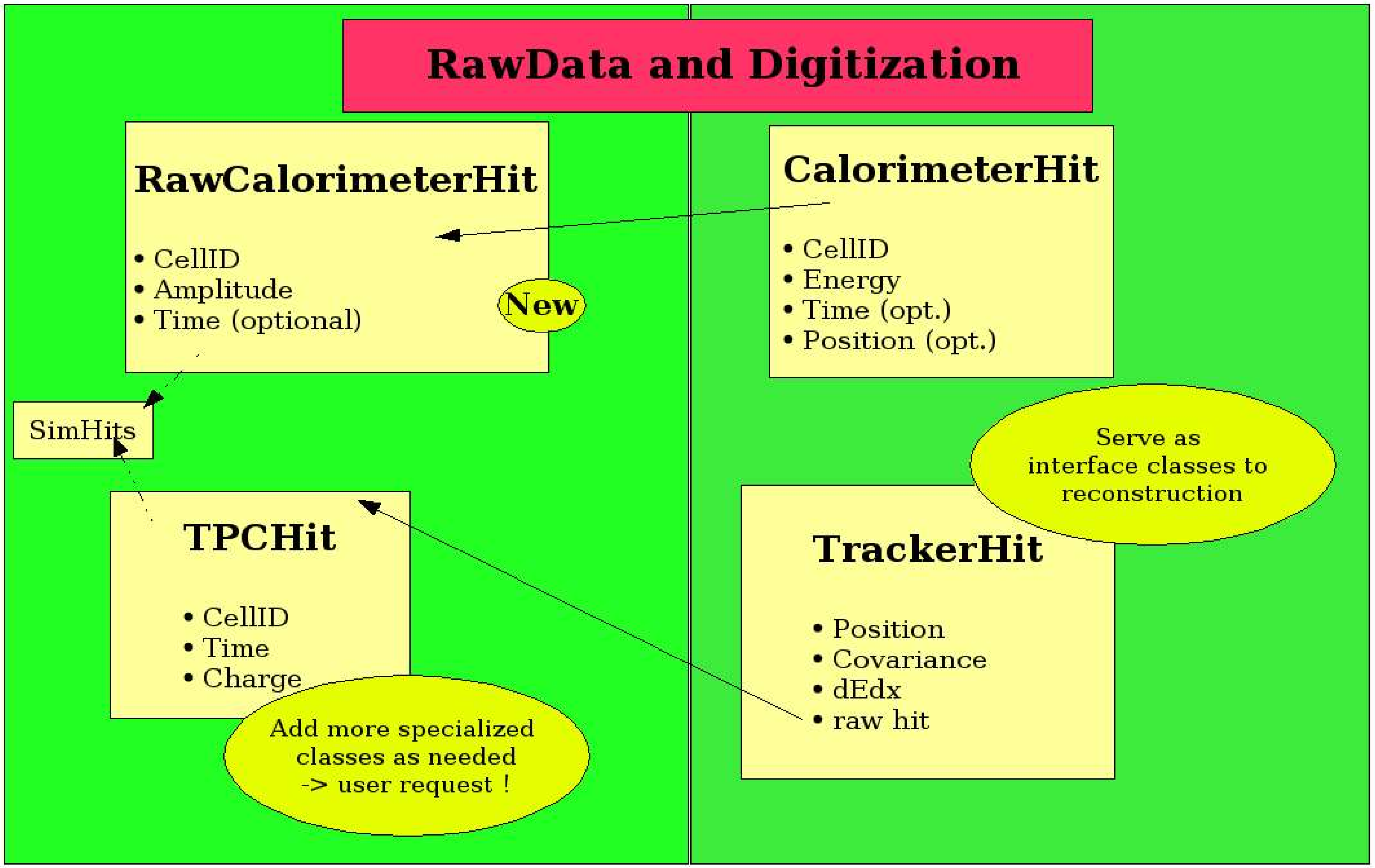


abstract event
abstract io
concrete classes
persistency implementation
<<interface>> EVENT::LCEvent
+~LCEvent()
+getRunNumber() : int
+getEventNumber() : int
+getDetectorName() : const std::string&
+getTimeStamp() : long
+getCollectionNames() : const StringVec*
+getCollection(...) : LCCollection*
+addCollection(...) : int
+removeCollection(...) : int
<<interface>> IO::LCWriter
+~LCWriter()
+open(...) : int
+writeRunHeader(...) : int
+writeEvent(...) : int
+close() : int
<<interface>> IO::LCReader
+~LCReader()
+open(...) : int
+readNextRunHeader() : EVENT::LCRunHeader*
+readNextEvent() : EVENT::LCEvent*
+readNextEvent(...) : EVENT::LCEvent*
+close() : int
+registerLCEventListener(...) : void
+removeLCEventListener(...) : void
+registerLCRunListener(...) : void
+removeLCRunListener(...) : void
+readStream() : int
IMPL::LCEventImpl
#_runNumber : int
#_eventNumber : int
#_timeStamp : long long
#_detectorName : std::string
#_map : mutable LCCollectionMap
#_colNames : mutable EVENT::StringVec
-_access : int
+LCEventImpl()
+~LCEventImpl()
+getRunNumber() : int
+getEventNumber() : int
+getDetectorName() : const std::string&
+getTimeStamp() : long
+getCollectionNames() : const EVENT::StringVec*
+getCollection(...) : EVENT::LCCollection*
+addCollection(...) : int
+removeCollection(...) : int
+setRunNumber(...) : void
+setEventNumber(...) : void
+setDetectorName(...) : void
+setTimeStamp(...) : void
#setAccessMode(...) : void
IOIMPL::LCEventIOImpl
friend SIO::SIOEventHandler
SIO_block
SIO::SIOEventHandler
-_evtP : IOIMPL::LCEventIOImpl**
-_evt : const EVENT::LCEvent*
#SIOEventHandler()
+SIOEventHandler(...)
+SIOEventHandler(...)
+~SIOEventHandler()
+xfer(...) : unsigned int
+version() : unsigned int
+setEvent(...) : void
SIO::SIOWriter
#_evtRecord : SIO_record*
#_hdrRecord : SIO_record*
#_runRecord : SIO_record*
#_stream : SIO_stream*
-_evtHandler : SIOEventHandler*
-_hdrHandler : SIOEventHandler*
-_colVector : std::vector<SIOCollectionHandler*>
-isFirstEvent : bool
-evtP : EVENT::LCEvent**
+SIOWriter()
+~SIOWriter()
+open(...) : int
+writeRunHeader(...) : int
+writeEvent(...) : int
+close() : int
#setUpHandlers(...) : void
SIO::SIOReader
#_evtRecord : SIO_record*
#_hdrRecord : SIO_record*
#_runRecord : SIO_record*
#_dummyRecord : SIO_record*
#_stream : SIO_stream*
-_defaultEvt : IOIMPL::LCEventIOImpl*
-_evtP : IOIMPL::LCEventIOImpl**
-_runP : IMPL::LCRunHeaderImpl**
-_runListeners : std::set<IO::LCRunListener*>
-_evtListeners : std::set<IO::LCEventListener*>
+SIOReader()
+~SIOReader()
+open(...) : int
+readNextRunHeader() : EVENT::LCRunHeader*
+readNextEvent() : EVENT::LCEvent*
+readNextEvent(...) : EVENT::LCEvent*
+close() : int
+registerLCEventListener(...) : void
+removeLCEventListener(...) : void
+registerLCRunListener(...) : void
+removeLCRunListener(...) : void
+readStream() : int
#setUpHandlers() : void
#readRecord() : int

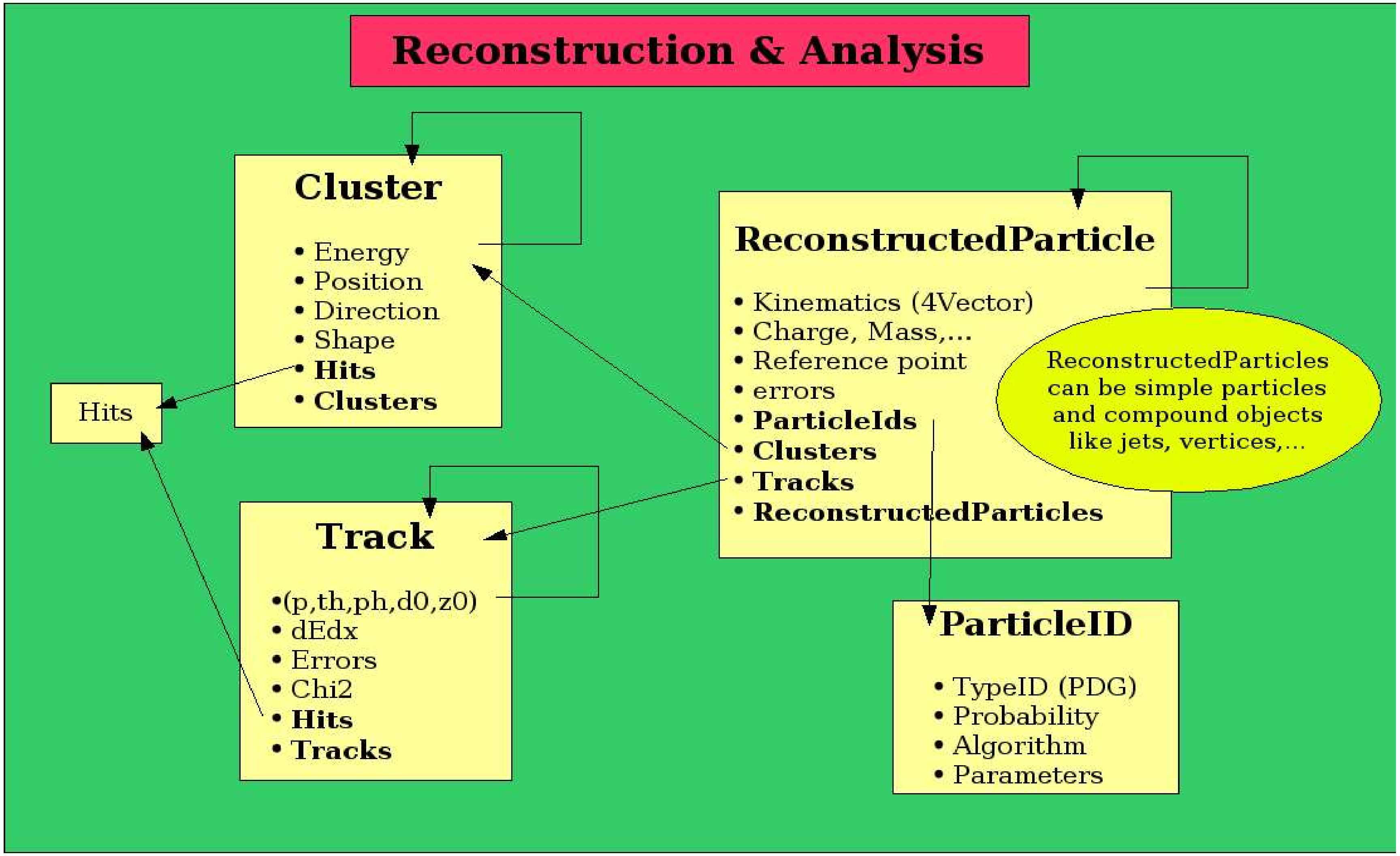
Data Model I


LCIO DataModel Overview
SimCalorimeterHit
RawCalorimeter Hit
LCRelation
CalorimeterHit
Cluster
MCParticle
LCRelation
Reconstructed Particle
TPCHit
Track
SimTrackerHit
TrackerHit
VTXHit
LCRelation
Monte Carlo
RawData
Digitization
Reconstruction & Analysis

Data Model III



Data Model IV



The Data Model: Comments

important ingredients:

- objects (tracks, clusters, ...) are grouped into collections
- there can be several collections of the same type of objects in the event:
  - tracks at IP
  - tracks at Calo face
  - VTX tracks
  - ...

  (if this is done, documentation is essential!)
- self-referencing of the objects allows the buildup of tree structures

Changes since LCWS-2004 -I-

- added LCRelation class to store (weighted) nxm relationships between LCObjects
  -> can be used to point back to MC-truth.
  -> can be used to link collections
- changed track parameters, now:
  d0, phi, omega, z0, tanLambda
- added generic named parameters to LCRunHeader, LCEvent and LCCollection
  -> use to store meta information on data

Changes since LCWS-2004 -II-

- added support for ‘generic’ user objects, that hold floats, ints and doubles:

  -> can be used to store arbitrary additional data
- added RawCalorimeterHit

  -> int Amplitude and int time
- added some convenient methods to the classes
- modified some classes to make the API more consistent

Reconstruction and Analysis

- need to provide a simple, lightweight environment for reconstruction and analysis
  - simple to use
  - low thresholds
  - in Europe: C++ support is essential (most people work on LHC in C++ environments)
  - no dependence on user backends (root, JAS, PAW, ...)

- Simple C++ based framework, in many ways similar to the existing LCD framework
- Developed in close collaboration with people doing actual test data analyses for TPC, Calo and physics studies

MARLIN

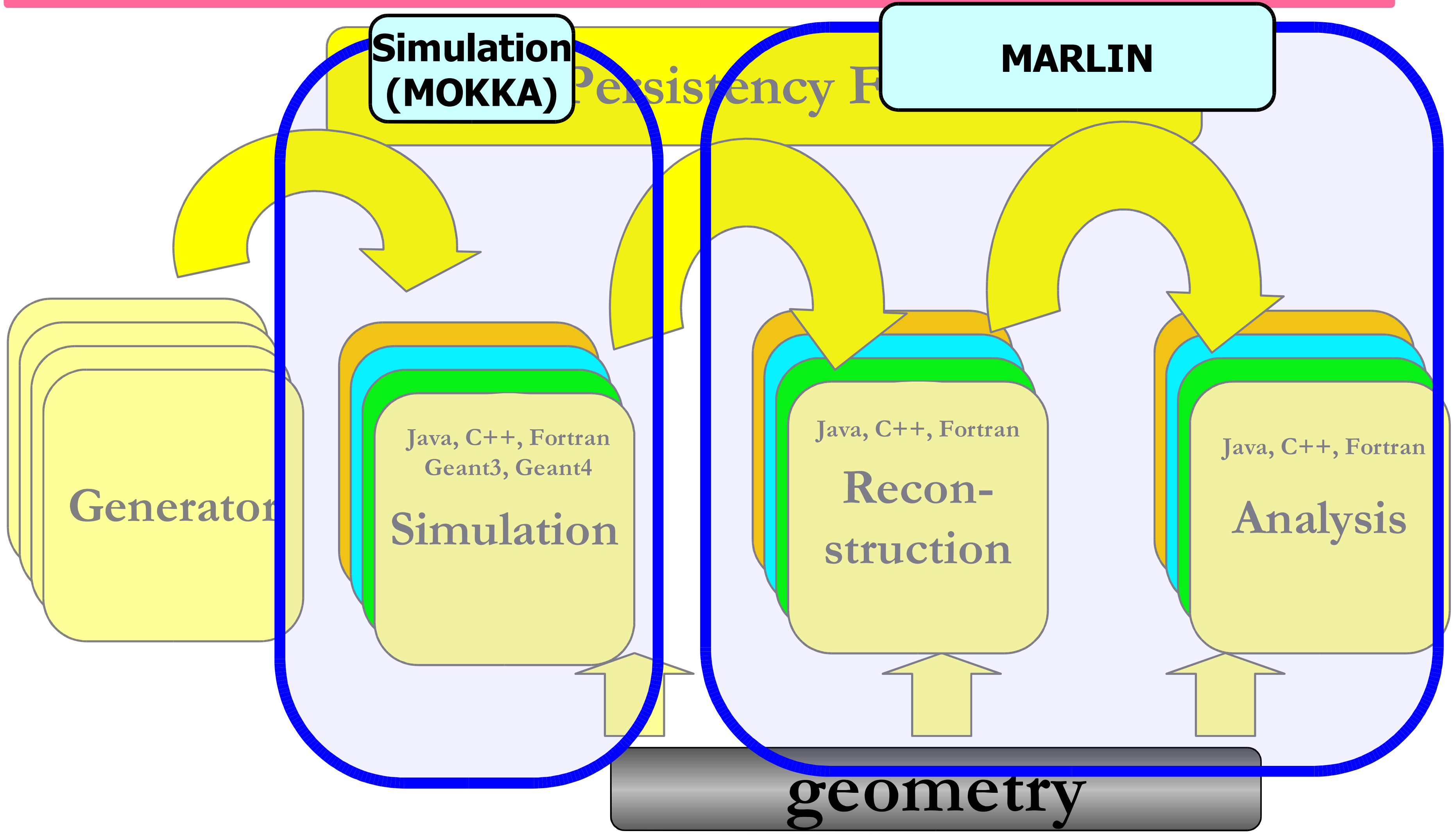
main author Frank Gaede
other contributors are welcome

Reconstruction and Analysis

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

- The LCEvent can be used as container for transient data in an application, e.g. reconstruction
- Application will call list of modules that read existing collections from the LCEvent and add resulting new Collections
- LCIO has (Event/Run)-Listener classes that can serve as base classes for modules
- define an application framework based on LCIO for reconstruction and analysis:

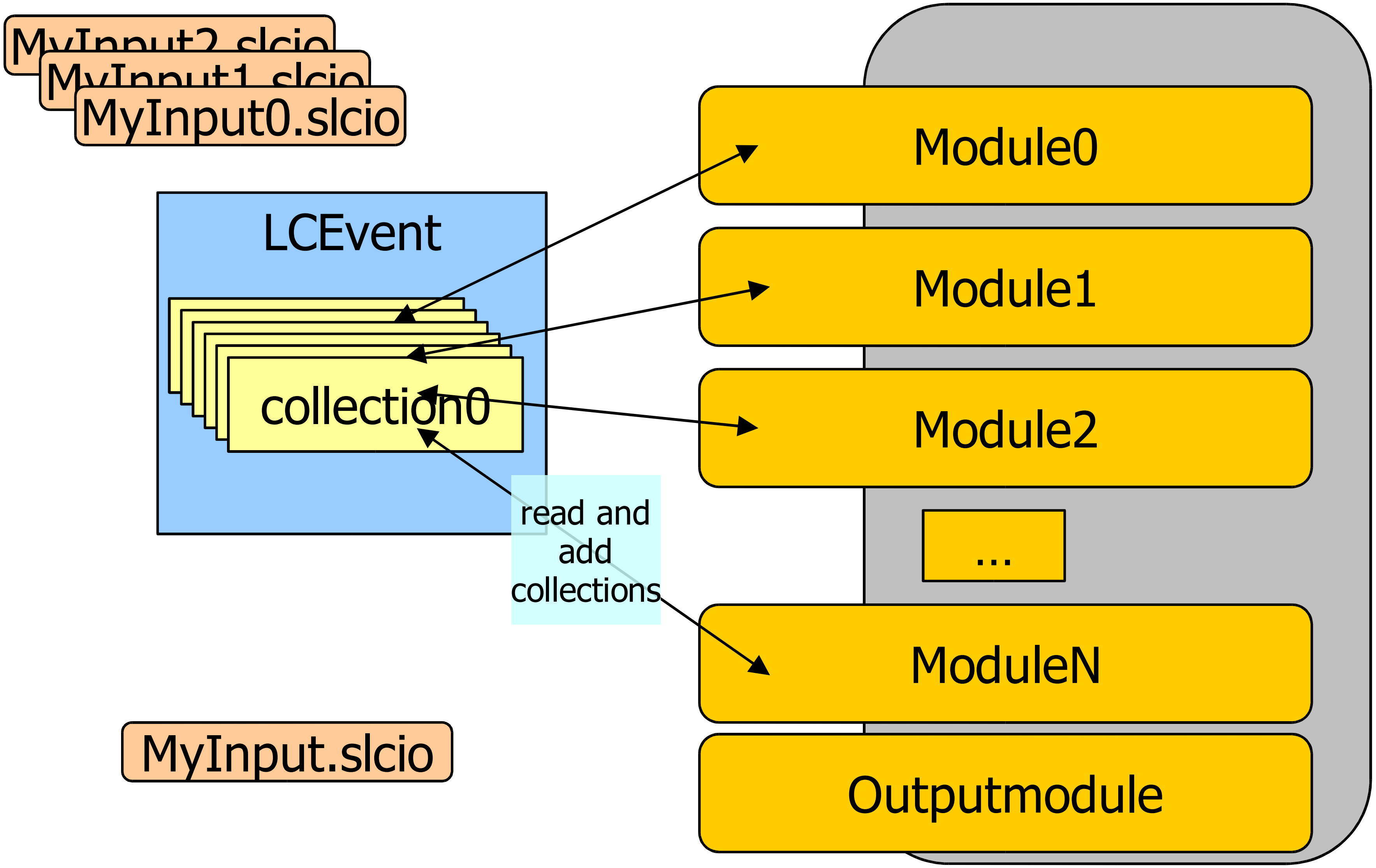
Motivation for MARLIN



Implementation of MARLIN

- use LCIO as transient data model
- use C++ only (so far)
- define base classes for modules that operate on LCIO (event) data
- provide simple user steering:
  - user defined variables for each module
  - input/output files
- provide main program !

Modules and the LCIOEvent



LCIOModule

- LCIOModule: base class for all user modules
- provides hooks (callbacks) for user actions:
  - init()
    - called once at program start
    - use to initialize histograms, counters, etc.
  - processRunHeader(LCRunHeader* run)
    - called for bookkeeping – new run conditions ?
  - processEvent(LCEvent* evt)
    - the working horse – this where the analysis takes place
  - end()
    - called once at end of job
    - write out histos, …

Under development in Marlin

- error handling
  - log files
  - error/warning messages
- naming convention for common parameters, e.g. InputCollectionName, OutputCollectionN.
- convention for passing user data between modules, e.g.:
  - as LCCollections of LCObjects
  - as global objects (singletons)
- some logic to control execution and I/O of events, e.g. a module might want to decide that the event is not worth processing then the rest of the modules should not be called …
- lots of additional functionality? need user feedback

MARLIN developments

Under discussion:

try to make the user hooks as similar as possible to the ones in the JAVA (LCD) framework to facilitate exchange of ideas

A problem:

The true parallel use of JAVA and C++ code to access the same LCIO even in memory is difficult

We are still far from a truly language independent frame

Status of Marlin

- very first implementation released to beta users at DESY (as LCIOFrame)

  -> see talk from J.Samson

- cvs repository with web based public access (will be provided by H.Vogt, Zeuthen)

- hope to have public beta release soon

  -> stay tuned

LCIO on the web

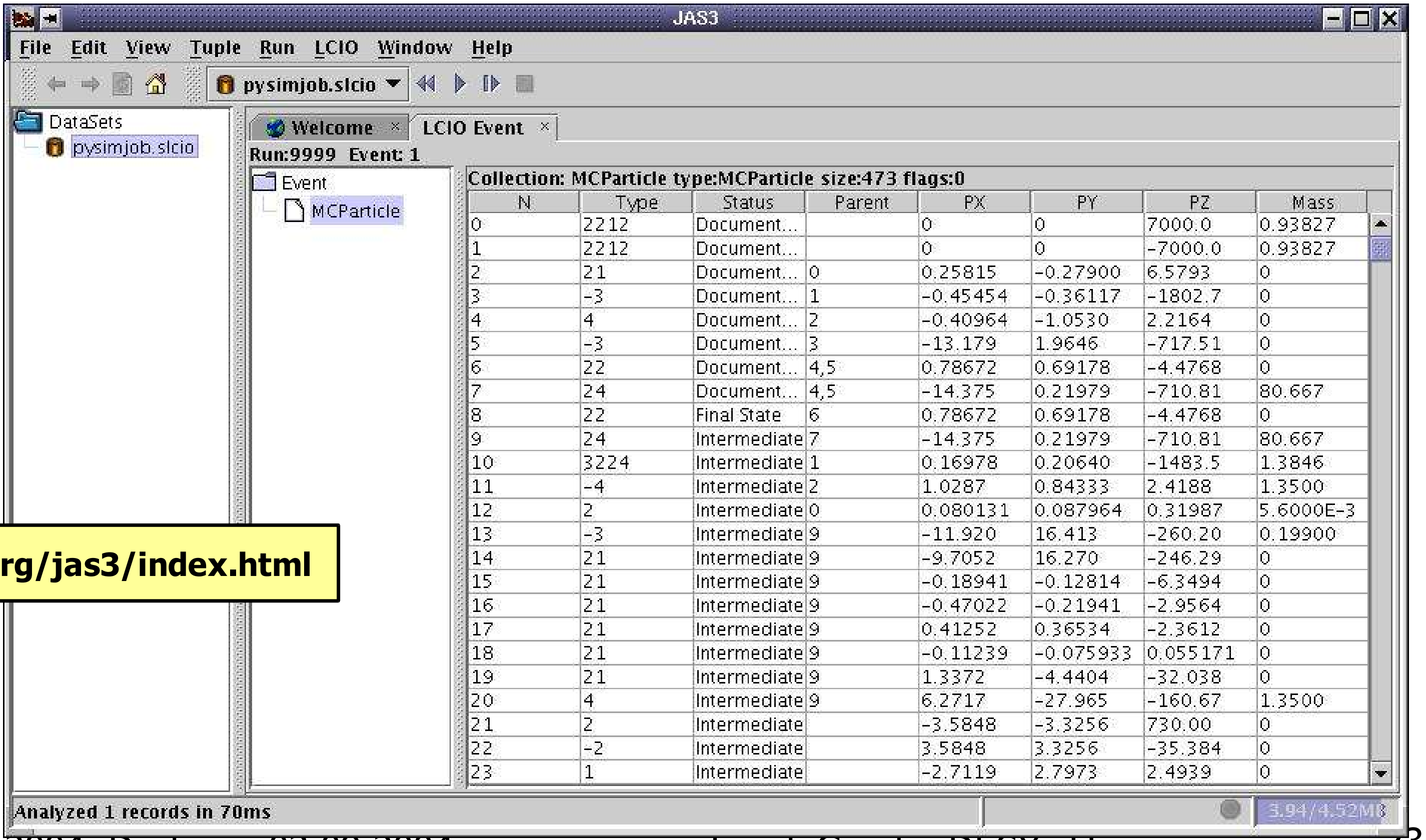
- LCIO homepage: http://lcio.desy.de
  - downloads and documentation
- LCIO forum at: http://forum.linearcollider.org
  - user/developer questions and comments
  - discussions on new developments
- LCIO bug reports at: http://bugs.freehep.org
  - bug report and new feature requests

LCIO Customers/Users

- Mokka simulation (see talk)
- Brahms reconstruction (see talk)
- JAS3
  - provides convenient file browser
  - will have LCIO-WIRED plugin -> generic event display !
- Calorimeter group (DESY)
  - has MiniCal raw data converted to LCIO files
  - to be used also for Hcal physics prototype
- TPC groups (DESY & Aachen & ...)
  - will use LCIO for prototype
- Lelaps fast Monte Carlo
- hep.lcd reconstruction
- other groups looking into using LCIO

JAS3 – LCIO

Note: JAS3 provides very nice native interfaces to LCIO:
browser, code wizard, event display



Run:9999 Event: 1

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

Analyzed 1 records in 70ms

http://jas.freehep.org/jas3/index.html

Summary

LCIO:

- available since some time, stable version 1.0, beta version 1.1
- new major release very soon
- http://lcio.desy.de and http://forum.linearcollider.org

MARLIN:

- first beta release available
- CVS in Zeuthen being set up (same place as other LC software)
- user feedback needed!
- http://www.desy.de/~gaede (real site to come soon)

User feedback is extremely important on all these projects!

Use the forum: http://forum.linearcollider.org
or sent e-mail to one of us

Appendix

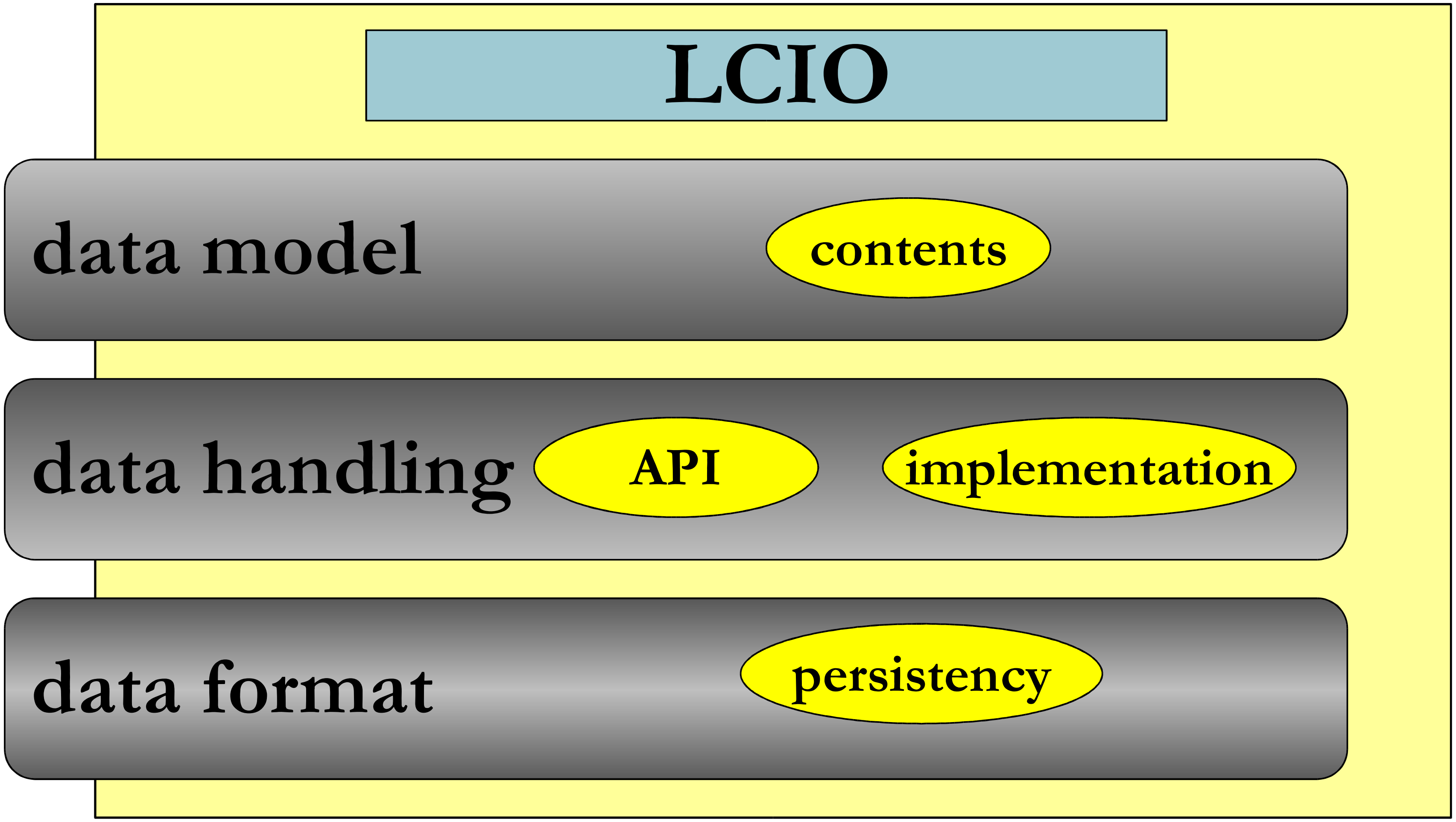
- Extension slides, details, examples

Requirements

- need Java, C++ and f77 (!) implementation
- extendable data model for current and future simulation studies
- user code separated from concrete data format
  - -> want to be flexible for future decisions on persistency
- needed a.s.a.p.

  -> keep it simple (lightweight)
- no dependence on other frameworks

LCIO persistency framework

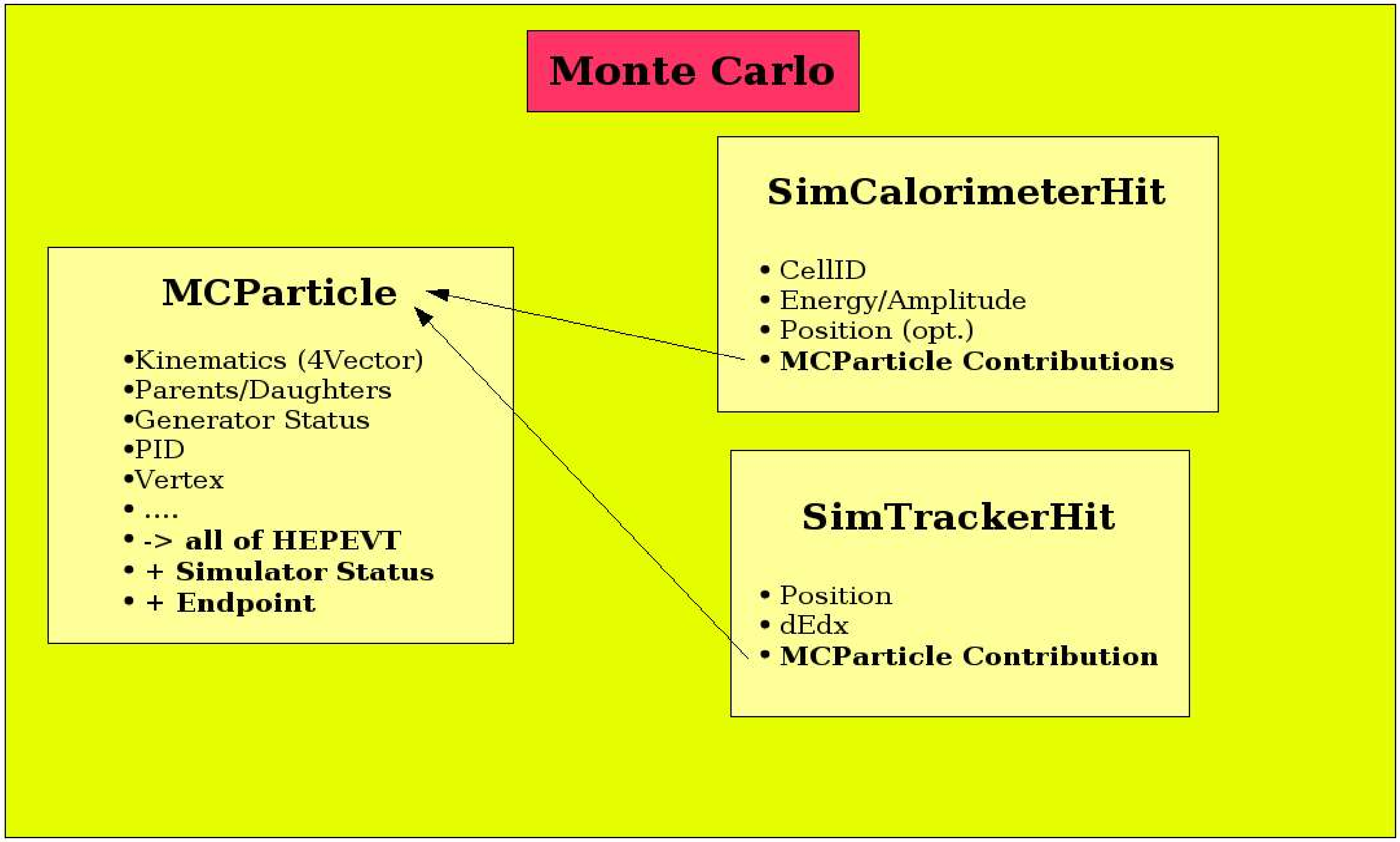


Data Format (persistency): SIO

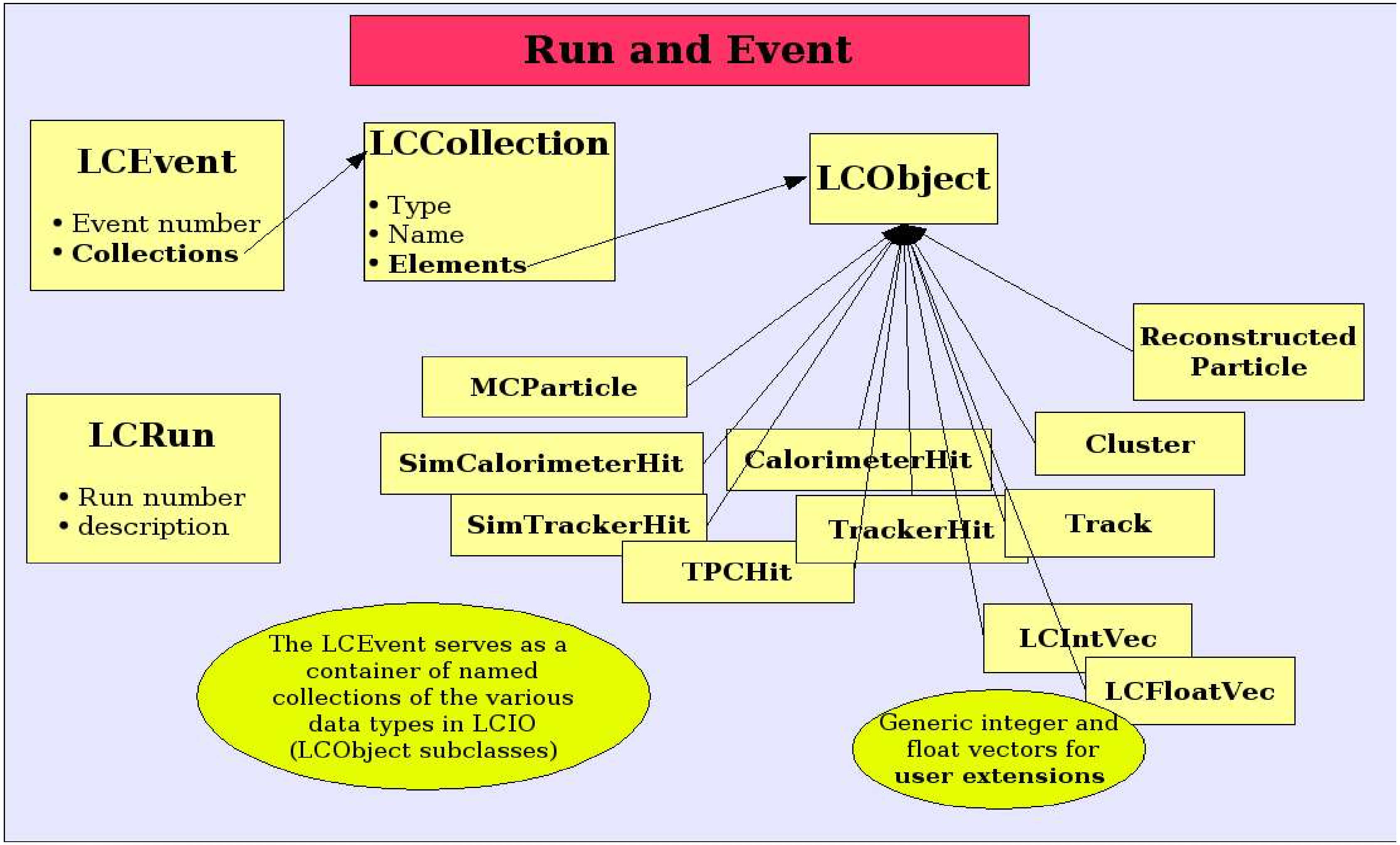
- SIO: Simple Input Output
- developed at SLAC for NLC simulation
- already used in hep.lcd framework
- features:
  - on the fly data compression ☺
  - some OO capabilities, e.g. pointers ☺
  - C++ and Java implementation available ☺
  - no direct access ☹

    -> use fast skip 😐

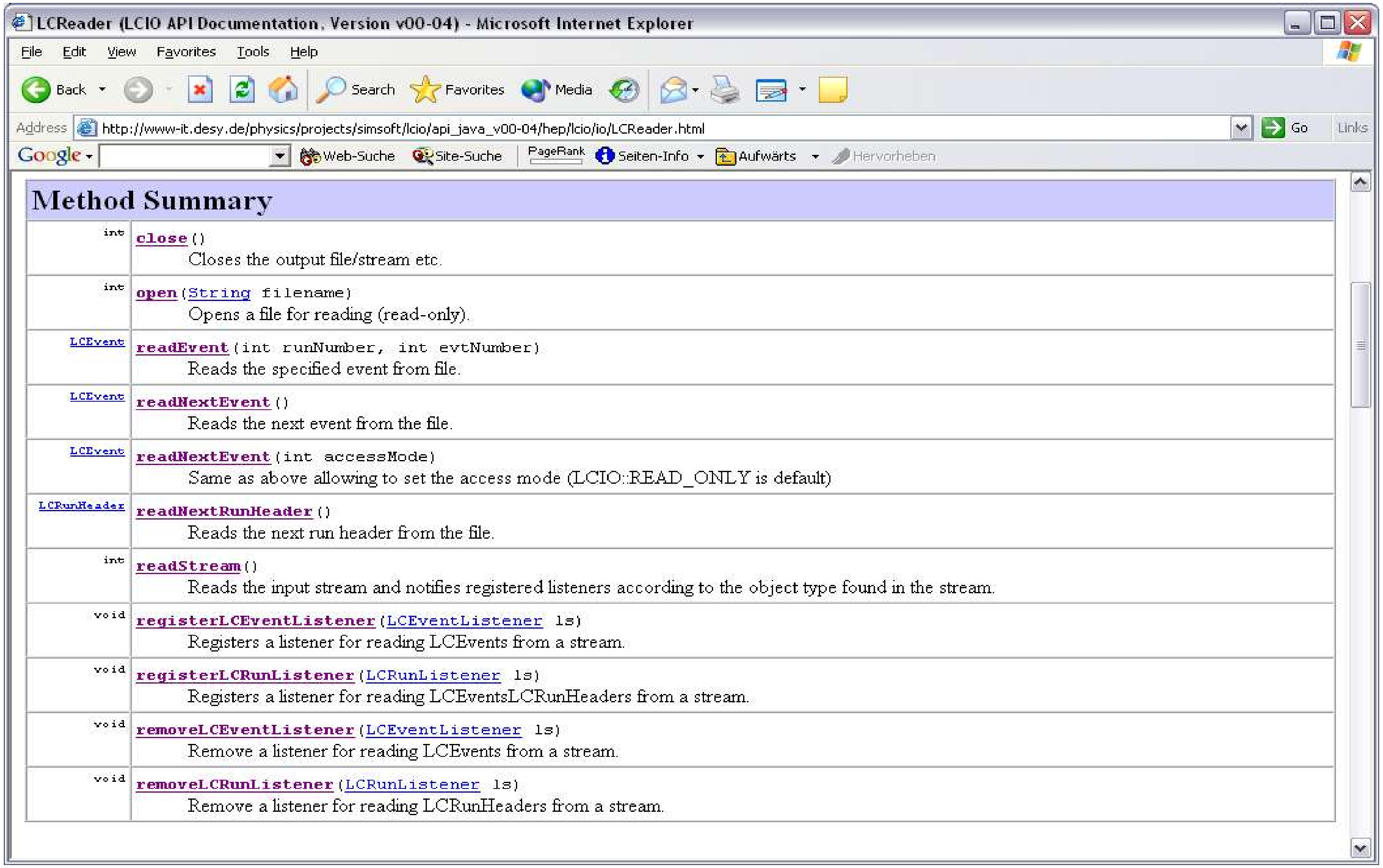
Data Model II



Data Model V


Run and Event
LCEvent
• Event number
• Collections
LCCollection
• Type
• Name
• Elements
LCObject
LCRun
• Run number
• description
MCParticle
SimCalorimeterHit
SimTrackerHit
TPCHit
CalorimeterHit
TrackerHit
Track
Cluster
Reconstructed Particle
LCIntVec
LCFloatVec
The LCEvent serves as a container of named collections of the various data types in LCIO (LCObject subclasses)
Generic integer and float vectors for user extensions

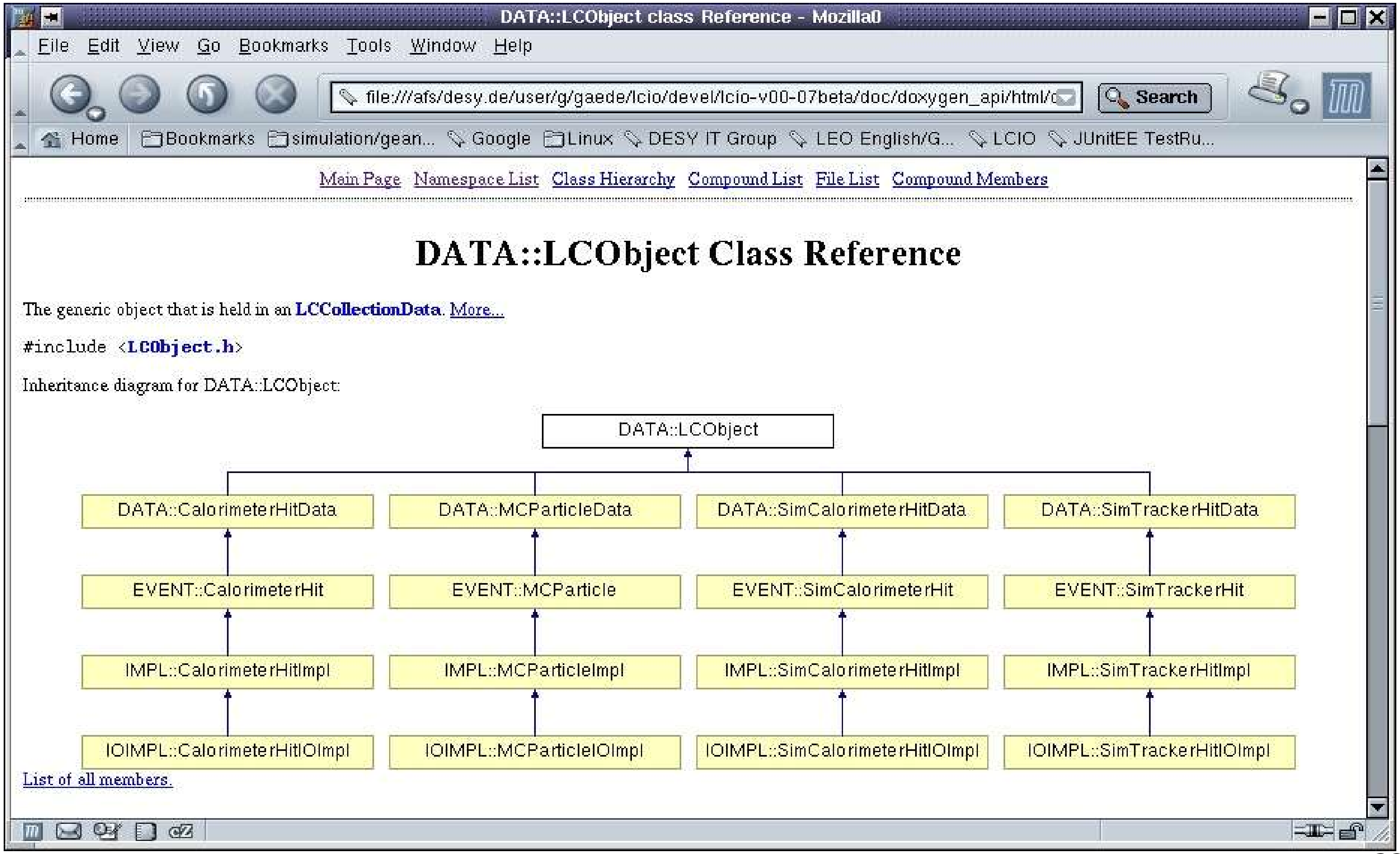
Javadoc example


LCReader (LCIO API Documentation, Version v00-04) - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://www-it.desy.de/physics/projects/simsoft/lcio/api_java_v00-04/hep/lcio/io/LCReader.html
Go Links
Google Web-Suche Site-Suche PageRank Seiten-Info Aufwärts Hervorheben

Method Summary

int	`close()` Closes the output file/stream etc.
int	`open(String filename)` Opens a file for reading (read-only).
LCEvent	`readEvent(int runNumber, int evtNumber)` Reads the specified event from file.
LCEvent	`readNextEvent()` Reads the next event from the file.
LCEvent	`readNextEvent(int accessMode)` Same as above allowing to set the access mode (LCIO::READ_ONLY is default)
LCRunHeader	`readNextRunHeader()` Reads the next run header from the file.
int	`readStream()` Reads the input stream and notifies registered listeners according to the object type found in the stream.
void	`registerLCEventListener(LCEventListener ls)` Registers a listener for reading LCEvents from a stream.
void	`registerLCRunListener(LCRunListener ls)` Registers a listener for reading LCEventsLCRunHeaders from a stream.
void	`removeLCEventListener(LCEventListener ls)` Remove a listener for reading LCEvents from a stream.
void	`removeLCRunListener(LCRunListener ls)` Remove a listener for reading LCRunHeaders from a stream.

Doxygen example


DATA::LCObject class Reference - Mozilla0
File Edit View Go Bookmarks Tools Window Help
file:///afs/desy.de/user/g/gaede/lcio/devel/lcio-v00-07beta/doc/doxygen_api/html/c
Search
Home Bookmarks simulation/gean... Google Linux DESY IT Group LEO English/G... LCIO JUnitEE TestRu...
Main Page Namespace List Class Hierarchy Compound List File List Compound Members

DATA::LCObject Class Reference

The generic object that is held in an **LCCollectionData**. More...

`#include <LCObject.h>`

Inheritance diagram for DATA::LCObject:

DATA::LCObject
DATA::CalorimeterHitData
DATA::MCParticleData
DATA::SimCalorimeterHitData
DATA::SimTrackerHitData
EVENT::CalorimeterHit
EVENT::MCParticle
EVENT::SimCalorimeterHit
EVENT::SimTrackerHit
IMPL::CalorimeterHitImpl
IMPL::MCParticleImpl
IMPL::SimCalorimeterHitImpl
IMPL::SimTrackerHitImpl
IOIMPL::CalorimeterHitIOImpl
IOIMPL::MCParticleIOImpl
IOIMPL::SimCalorimeterHitIOImpl
IOIMPL::SimTrackerHitIOImpl

List of all members.