LCIO

Persistency and Data Model for LC Simulation and Reconstruction

LCWS 2004, Paris

Simulation, April 20th 2004

Frank Gaede DESY -IT-

Outline

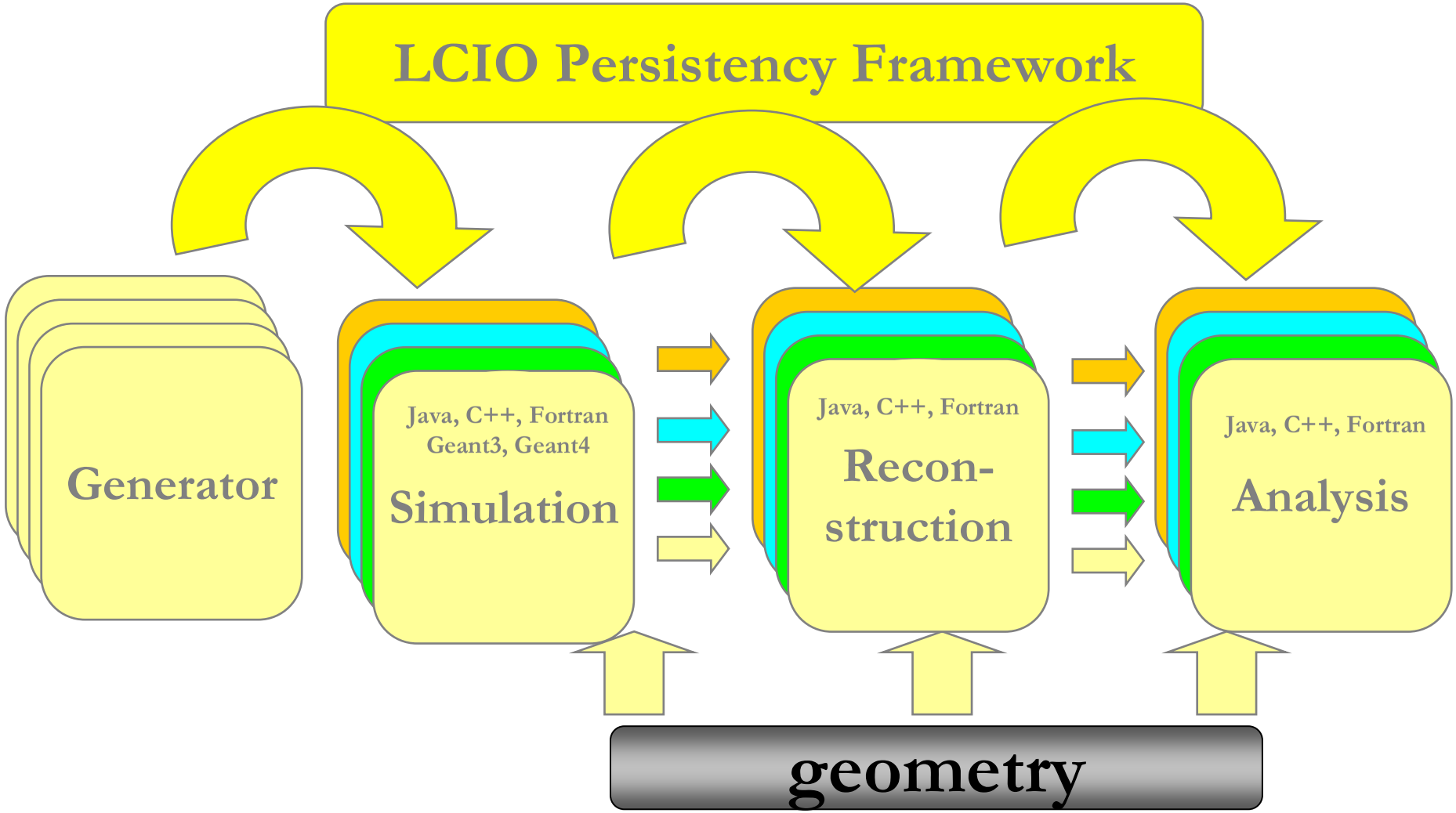
- Introduction/Motivation
- Implementation
- Data model
- Status
- Examples
- Summary

Introduction

- at 3rd ECFA workshop in Prague decided to have
  Data format/persistency task force:
  "Define an abstract object persistency layer and a data model for linear collider simulation studies until the Amsterdam workshop."

- -> **LCIO** – Linear Collider Input/Output
  - DESY/SLAC/LLR joined project
  - design of data model and software first introduced at the 4th ECFA workshop in Amsterdam
  - since Montpellier production version 1.0 (simulation only)
  - now v1.1beta released (incl. reconstruction)

Motivation for LCIO

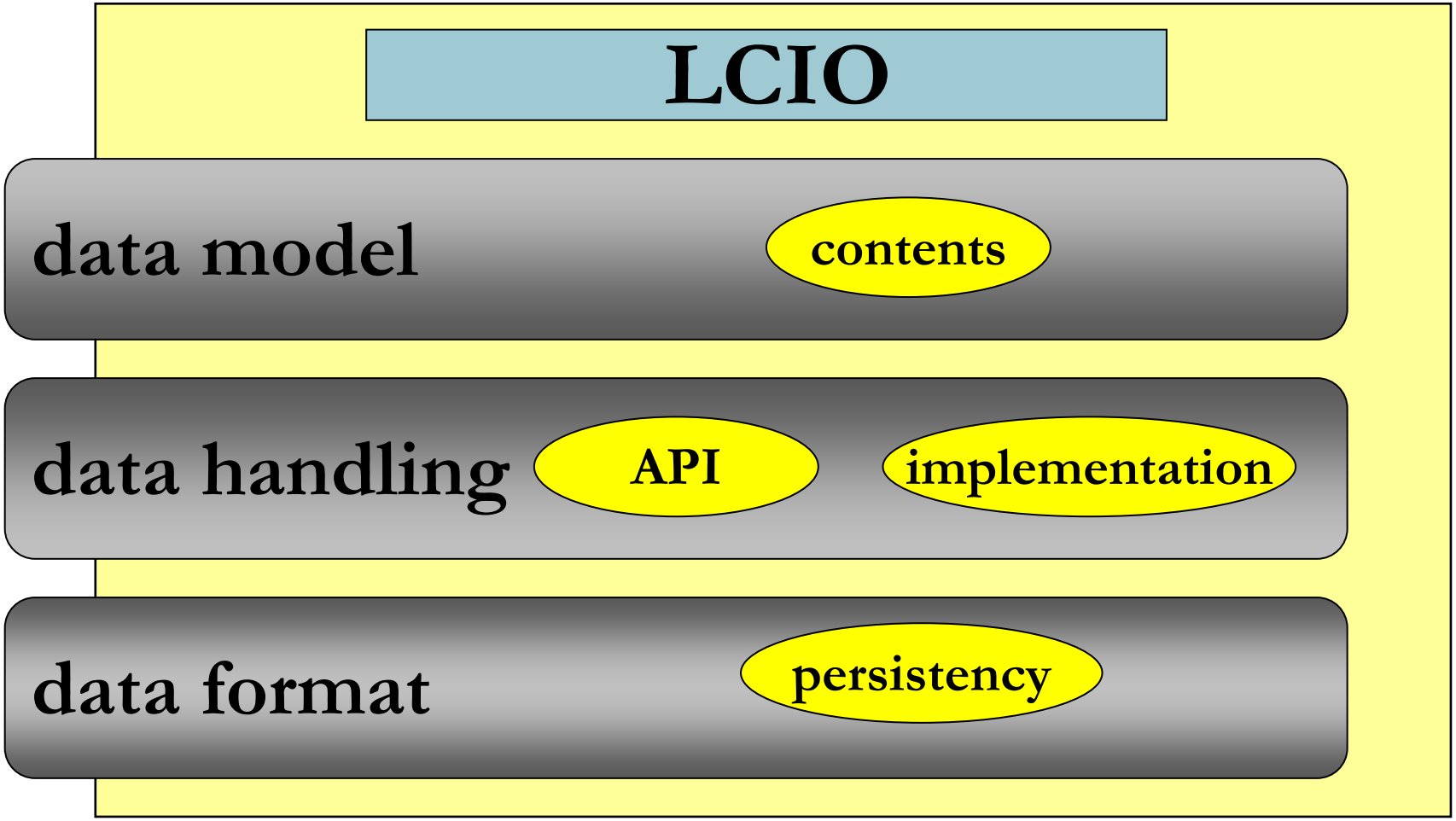


Requirements

- 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

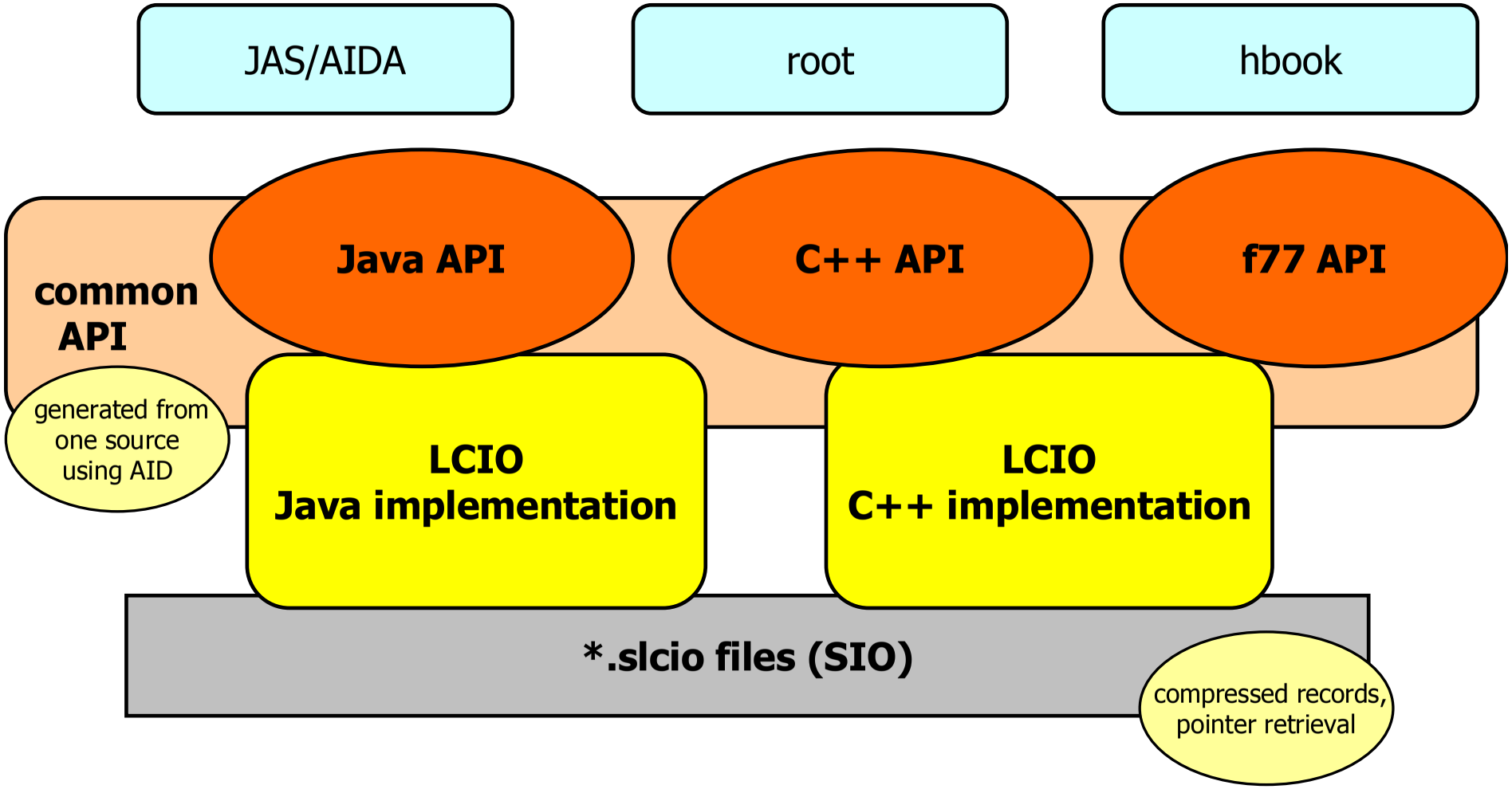


Data Format (persistency): SIO

- 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 😐

LCIO SW-Architecture



C++ and f77 example code

emacs@pcx3340.desy.de <7>

File Edit Options Buffers Tools C++ Help

```
// ---- event loop ------
const LCEvent*  event ;
while( (event = lcRdr->readNextEvent()) != 0 ){


  int runNum = event->getRunNumber()  ;
  int evtNum = event->getEventNumber()  ;
  string detName = event->getDetectorName() ;

std:cout << "  run: " << runNum << std::endl ;
std:cout << "  evt: " << evtNum << std::endl ;
std:cout << "  det: " << detName << std::endl ;

}

//---- end event loop ------
```

--:-- anajob.cc (C++ Abbrev)--L16-- 4%-----

emacs@pcx3340.desy.de <6>

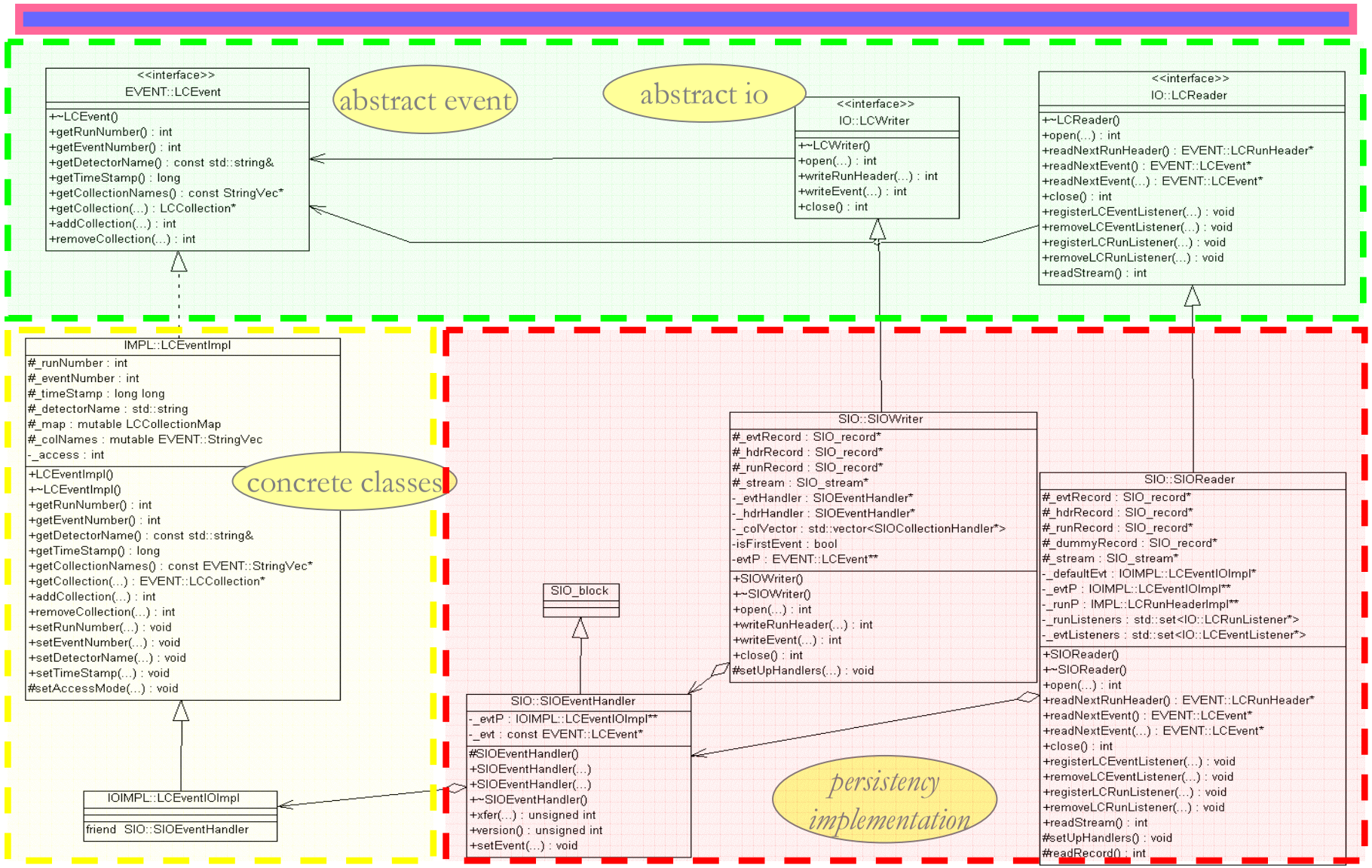
File Edit Options Buffers Tools Fortran Help

```
c     ---- event loop ------
      do 10
         event = lrdrreadnextevent( reader )
         if( event.eq.0 ) goto 11

         runnum = levtgetrunnumber( event )
         evtnum = levtgeteventnumber( event )
         detname = levtgetdetectorname( event )

         write(*,*) "  run: ", runnum
         write(*,*) "  evt: ", evtnum
         write(*,*) "  det: ", detname

 10   continue
 11   continue
c     ---- end even
```

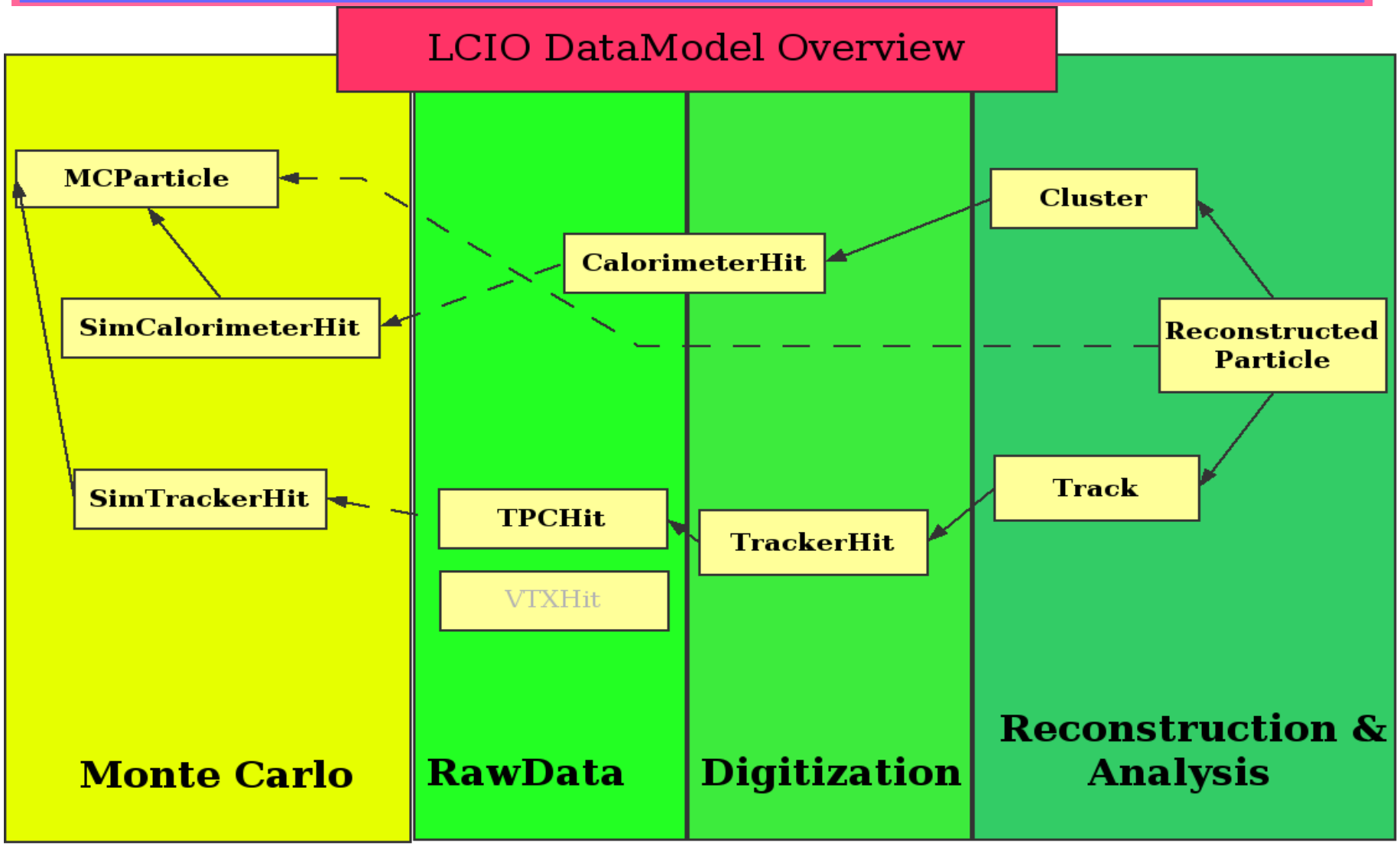
--:-- anajob.f

plus additional methods in f77 for user convenience, e.g. HEPEvt <-> LCIO conversion

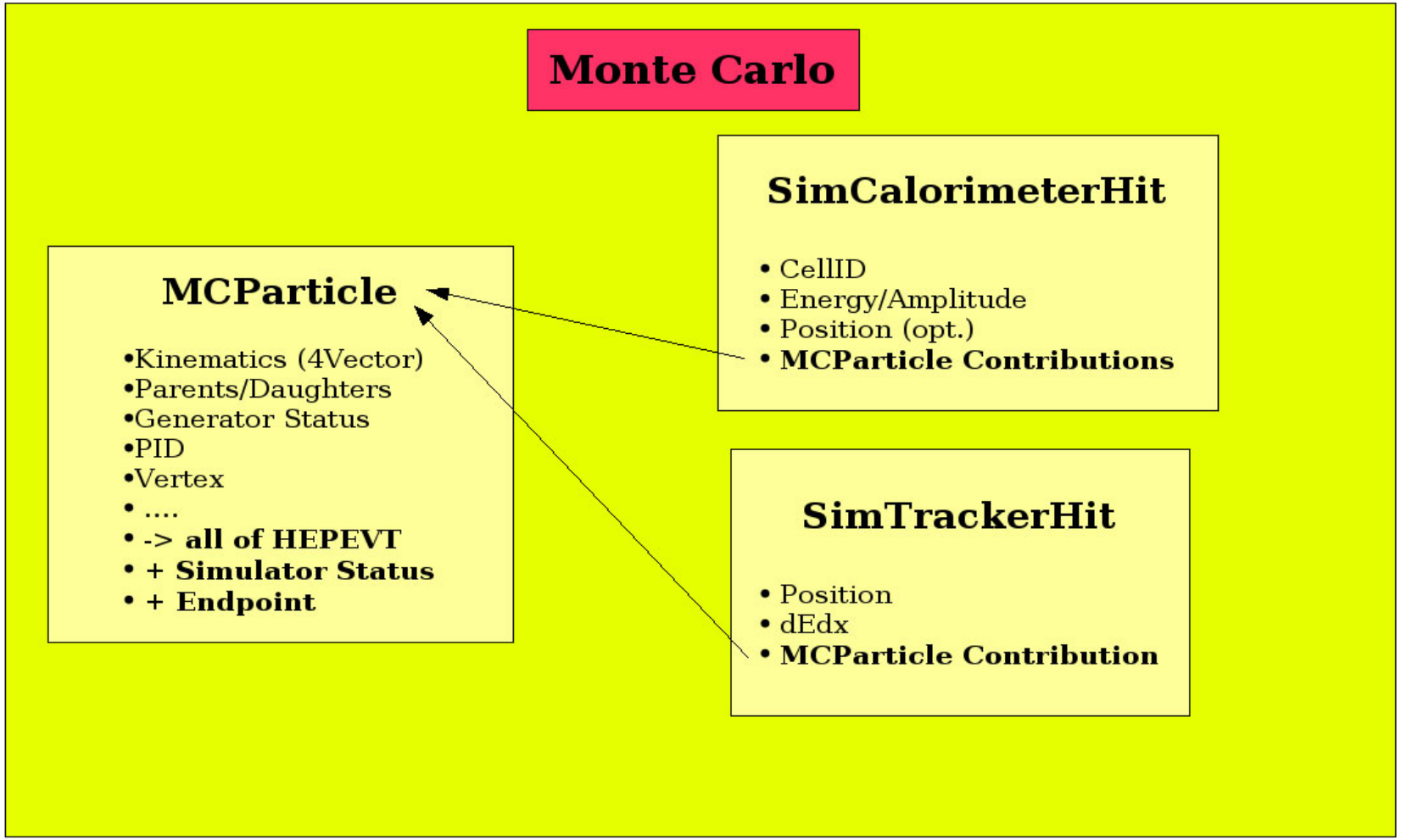
Implementation - Design



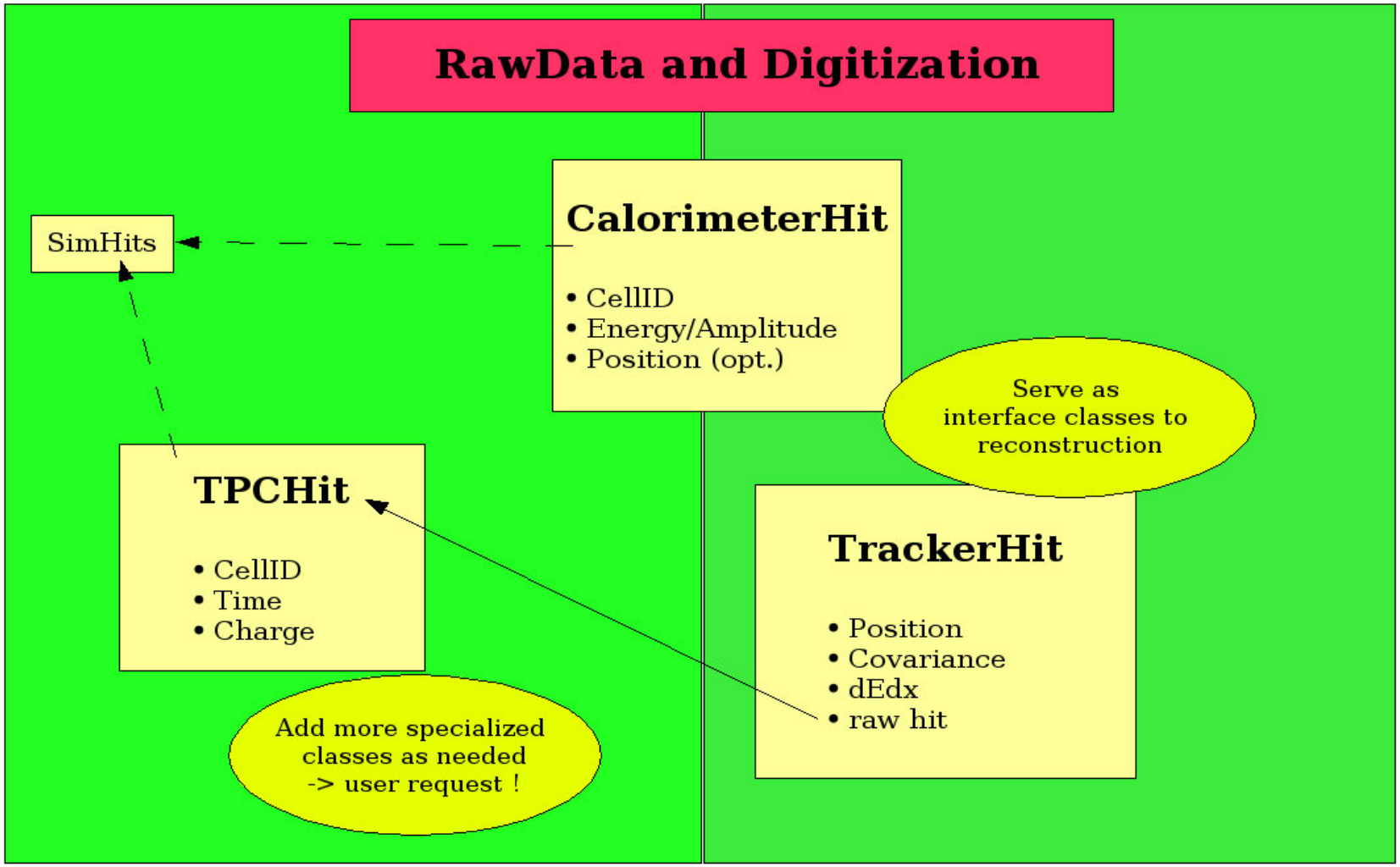
Data Model I


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

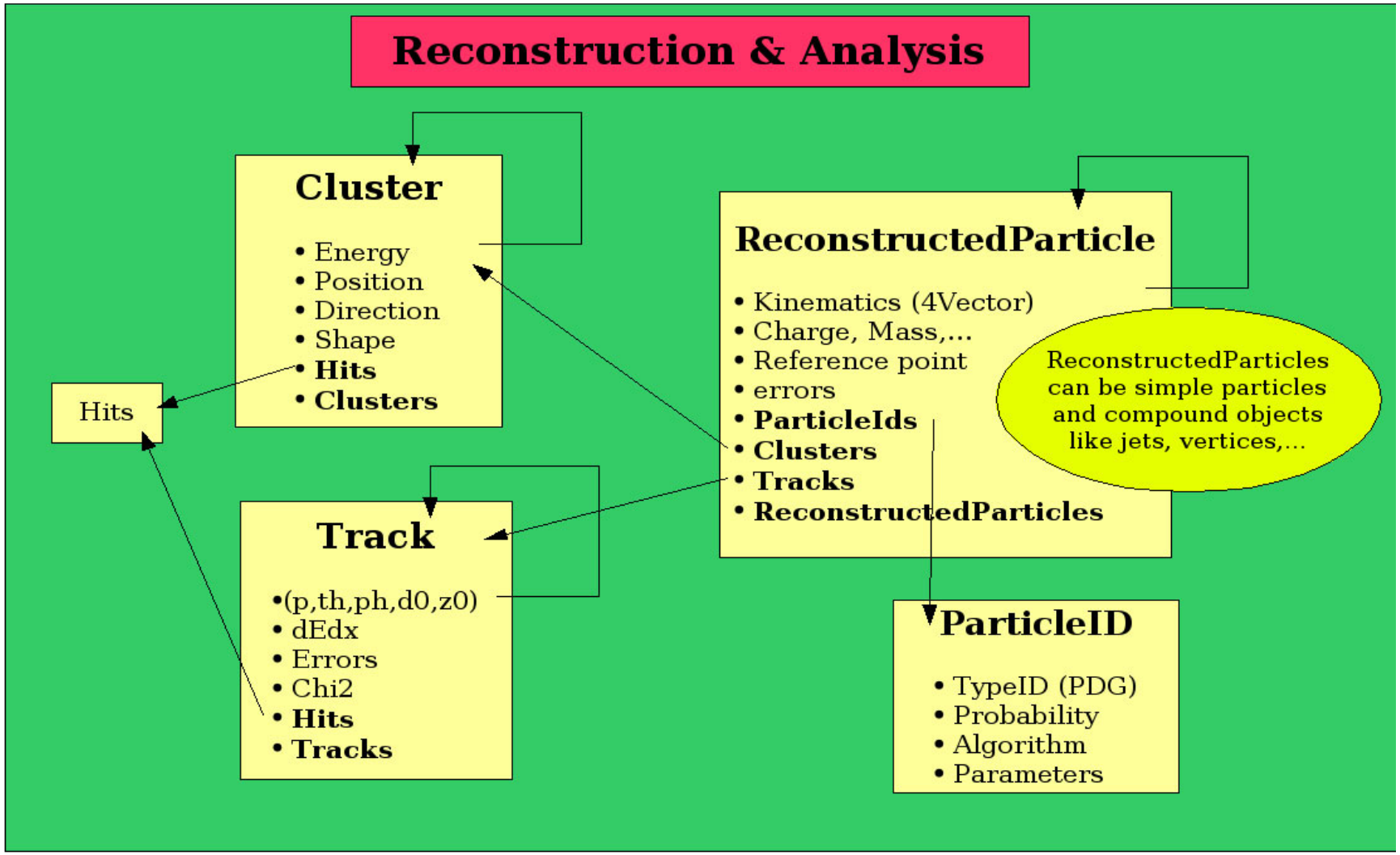
Data Model II



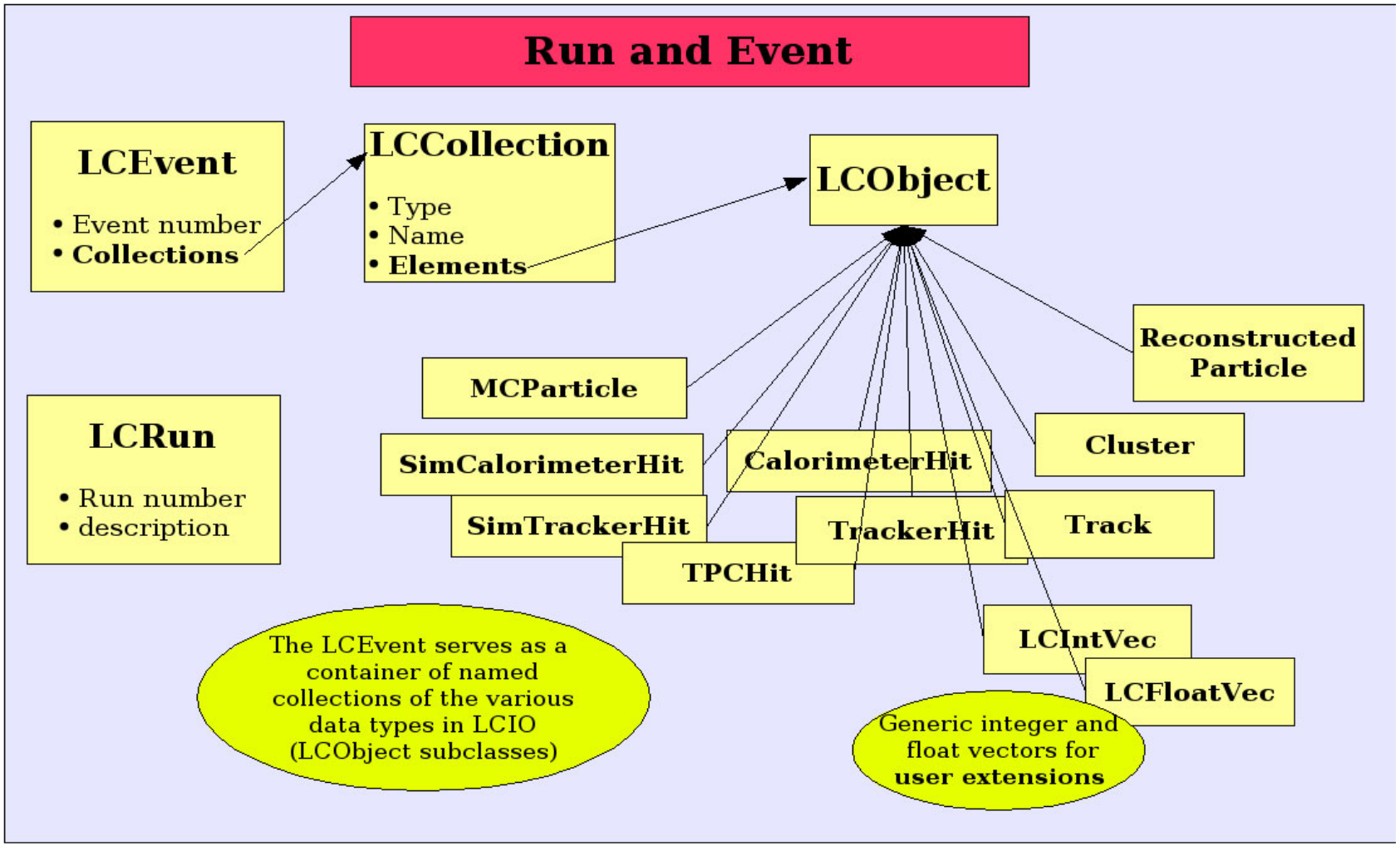
Data Model III



Data Model IV



Data Model V

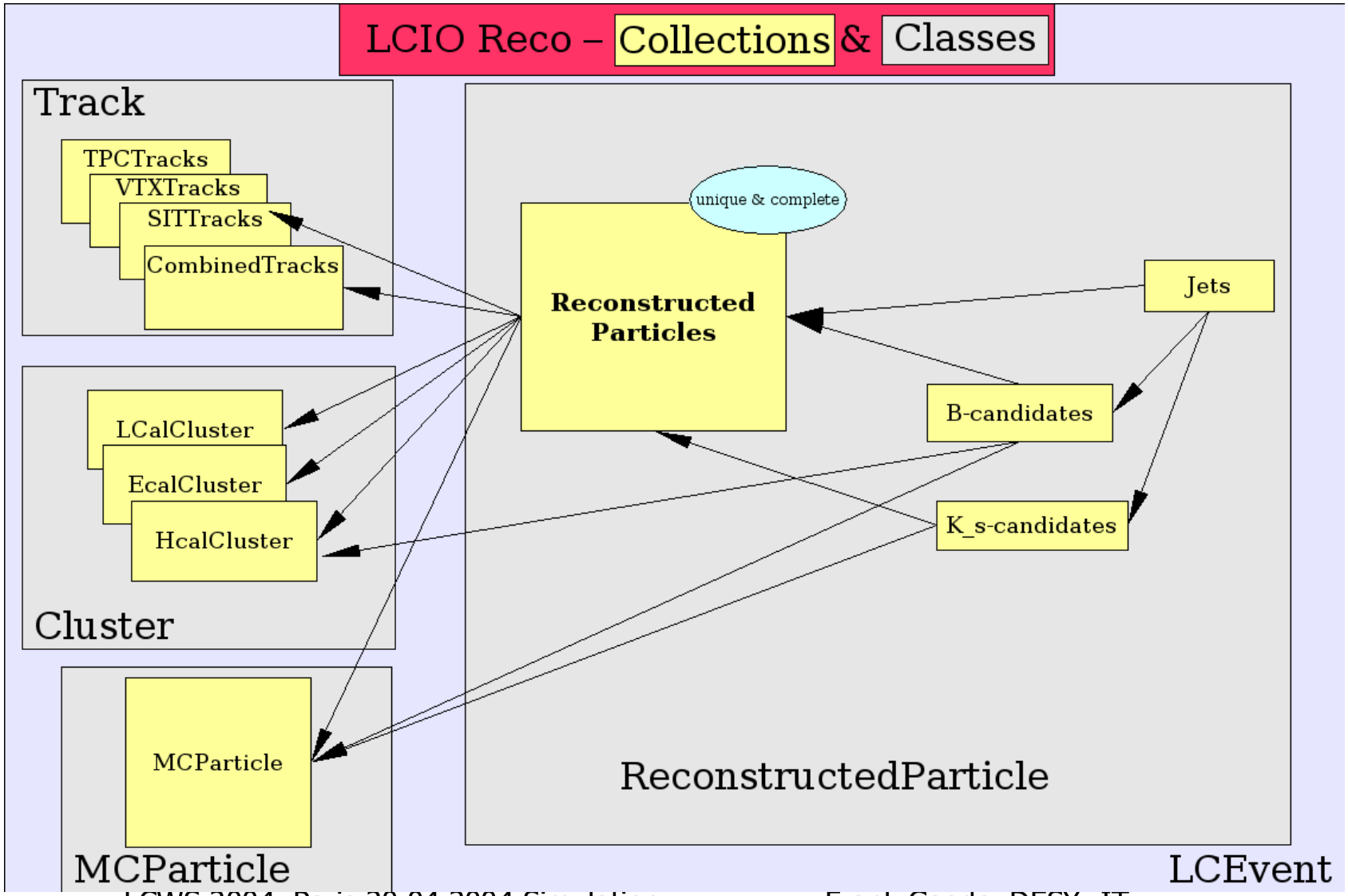

Run and Event
LCEvent
• Event number
• Collections
LCCollection
• Type
• Name
• Elements
LCObject
LCRun
• Run number
• description
MCParticle
SimCalorimeterHit
SimTrackerHit
TPCHit
CalorimeterHit
TrackerHit
Reconstructed Particle
Cluster
Track
LCIntVec
LCFloatVec
The LCEvent serves as a container of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

LCIO for Transient Data

- 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
- LCIO defines a simple application framework

  -> see example *lcioframe* provided in release

Example LCEvent



LCIO Status I

- production version 1.0 (09/2003)
  - C++, Java, f77 complete for simulation data
    - and generator data (HEPEvt<->LCIO)
  - simple example code for all languages
  - 'real world' examples (JAS3, root, hbook)
  - documentation
    - users manual
    - API documentation HTML (javadoc, doxygen)
  - available for download via CVS
  - linux (gcc), windows (cygwin)
- schema evolution from now on (reading old files)

LCIO Status II

- new beta release v1.1beta:
  - includes C++ implementation for reconstruction data (Track, Cluster, ReconstructedParticle)
  - Java and f77 soon to come
  - added optional use of CLHEP four vectors:
    - Handler classes for MCParticle/ReconstructedParticles that can be used as 4vectors and LCIO Objects
  - reading of a list of files (chain)
  - simplified API structure (one level of inheritance less)
  - not for production use yet !

    -> user feedback welcome

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

Javadoc example

LCReader (LCIO API Documentation, Version v00-04) - Microsoft Internet Explorer

File Edit View Favorites Tools Help

Back Search Favorites Media

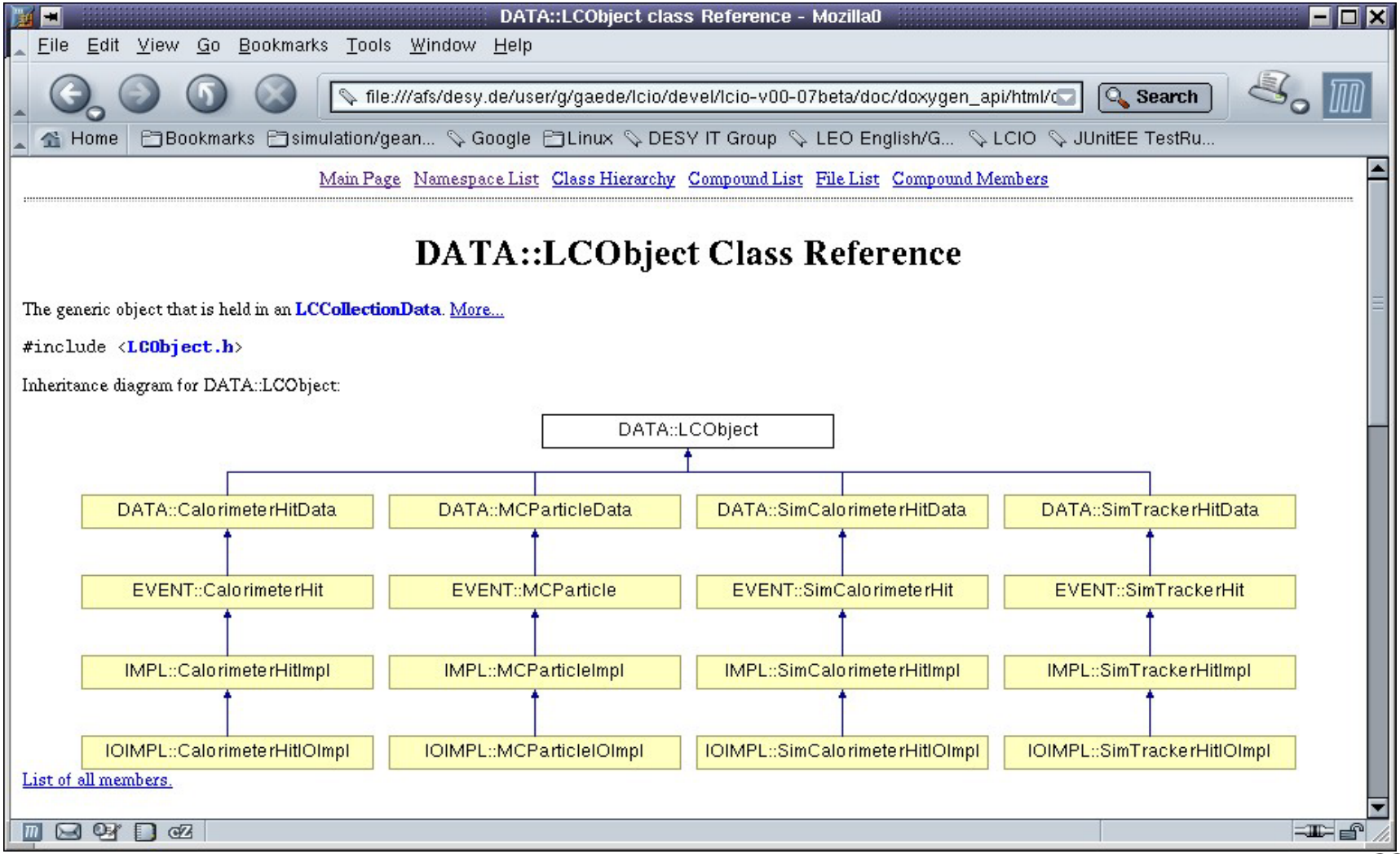
Address http://www-it.desy.de/physics/projects/simsoft/lcio/api_java_v00-04/hep/lcio/io/LCReader.html Go Links

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.

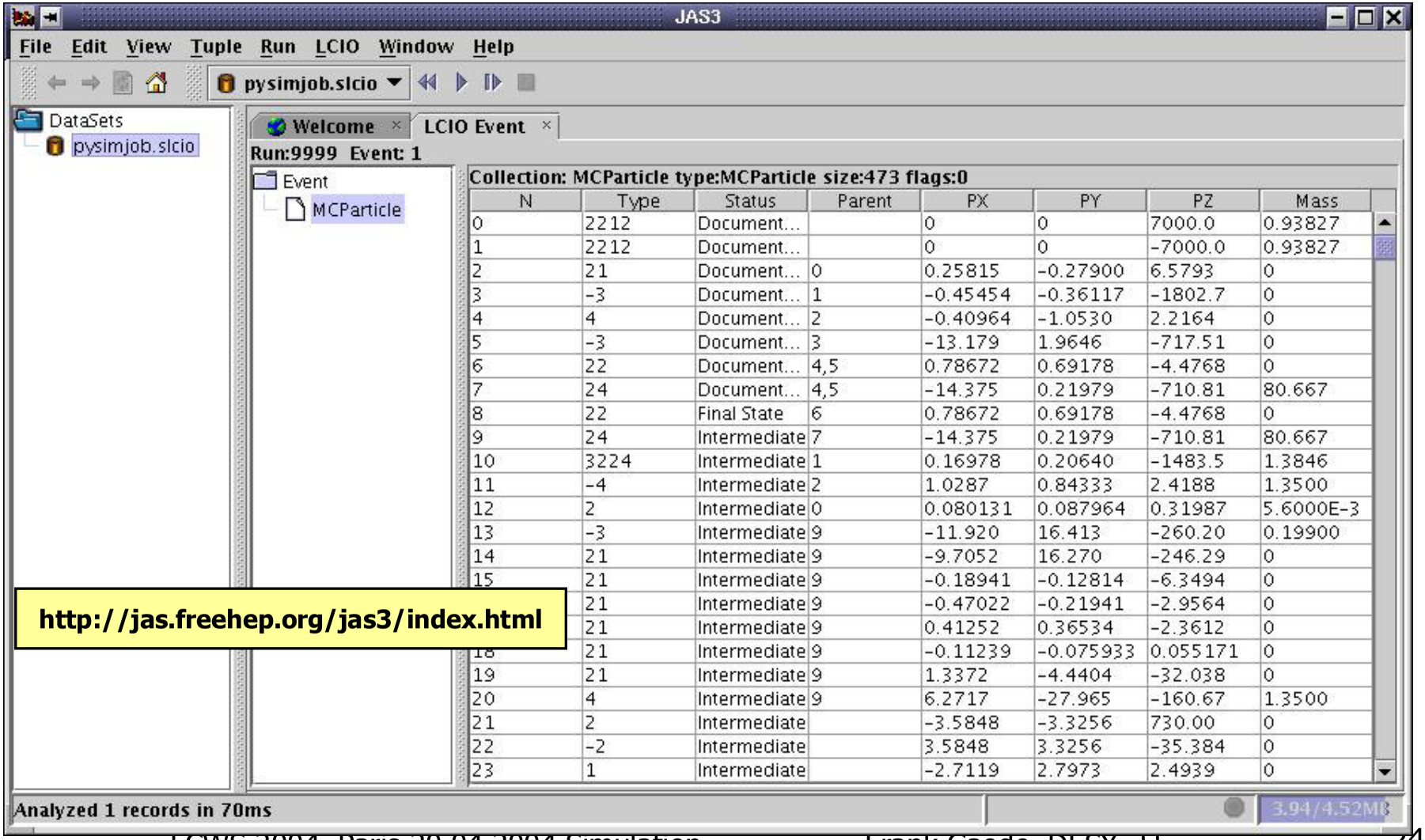
Doxygen example



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 & LBNL?)
  - will use LCIO for prototype
- Lelaps fast Monte Carlo
- hep.lcd reconstruction
- other groups looking into using LCIO

JAS3 – LCIO file browser



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
[illegible]	21	Intermediate	9	-0.47022	-0.21941	-2.9564	0
[illegible]	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

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

LCIO Future developments

- reconstruction data model
  - refine some details
  - Java and f77 implementation soon
- add convenient methods and tools
  - analyzing the MCParticle tree (graph)
    - iterators for MCParticles, e.g. all stable/final particles
    - analyzing parent/daughter relationships
  - adding convenience iterators for LCIO types
    - simplify complex C++ syntax with dynamic_casts etc.
- add possibility to store generic user data
  - calibration constants etc.
  - discussion ongoing about implementation (not so easy)
- respond to user requests

Summary

- LCIO, persistency and datamodel for the LC:
  - Java, C++ and f77 user interface
  - Java and C++ implementation
  - data model for simulation, (prototype) data and reconstruction

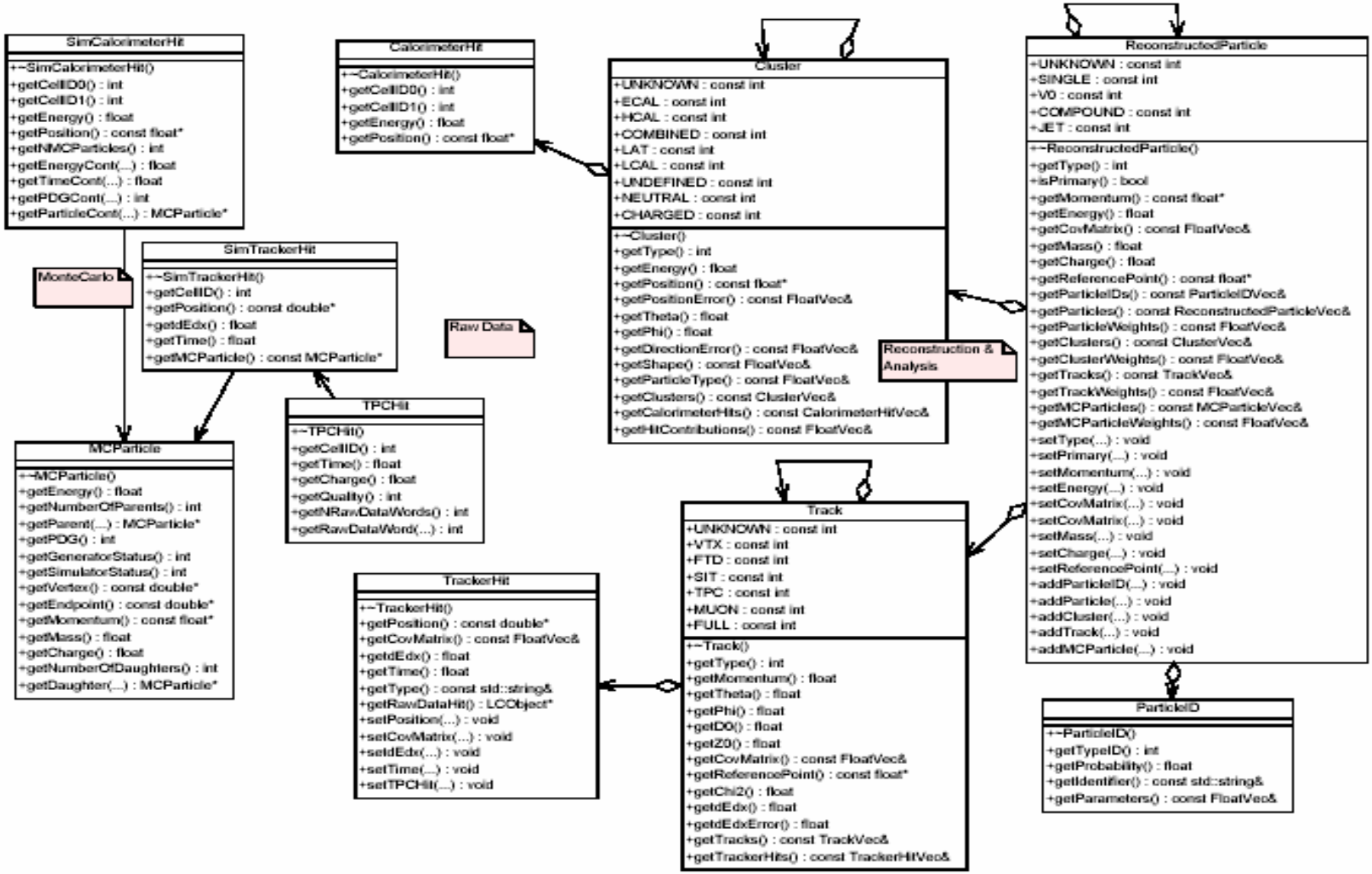
  -> persistent and transient
- production version v1.0
- new beta version released (v1.1beta)
- used by several groups and tools
  - others invited to join !
- see LCIO homepage for more:

http://lcio.desy.de

Appendix

- Extension slides, details, examples

Data entities UML



Example code: reading events

```
LCReader* lcReader =
        LCFactory::getInstance()->createLCReader() ;
  try{
    lcReader->open( "myFile" ) ;
    LCEvent *myEvt ;
    while( 1){
      try{ myEvt =cReader->readNextEvent() ) ;
      }catch(EndOfDataException) { break ;}
      cout << "  Evt :  "  << myEvt->getEventNumber()
                << " -        "  << myEvt->getRunNumber()
            << ":          "  << myEvt->getDetectorName()  << endl ;
    }
    lcReader->close() ;
  }catch(IOException& e){ cout << e.what() << endl ; }
```

Example code: reading collections

```
…LCEvent *evt ;
  while( 1){
    try{ evt =lcReader->readNextEvent() ) ;
    }catch(EndOfDataException) { break ;}
    const LCCollection*col ;
    try{ col = evt->getCollection( "EcalHits");
      int nHits =  col->getNumberOfElements() ;
      for( int i=0 ; i< nHits ; i++ ){
      const CalorimeterHit* hit =
          dynamic_cast<const CalorimeterHit*>
                                  ( col->getElementAt( i ) ) ;
      const float* x =  hit->getPosition() ;
      cout << "x: " << x[0] << ", " << x[1] << ", " << x[2]
            << "  energy:    " << hit->getEnergy() << endl ;
    }catch(DataNotAvailableException){
      cout << "collection not found: EcalHits " << endl ; }
   }…
```

Example code : writing data (events)

```
LCWriter* lcWrt = LCFactory::getInstance()->createLCWriter() ;
try{
   lcWriter->open( "myFile" ) ;
   for( int i=0; i<NEVENT; i++ ){
      LCEventImpl*  evt = new LCEventImpl() ;
      evt->setRunNumber(  rn   ) ;
      evt->setEventNumber( i ) ;
      // add collections ...
      lcWrt->writeEvent( evt ) ;
      delete evt ; // C++ only :)
    }
    lcWriter->close() ;
}catch(IOException& e) { cout << e.what() << endl ; }
```

Data model - Design