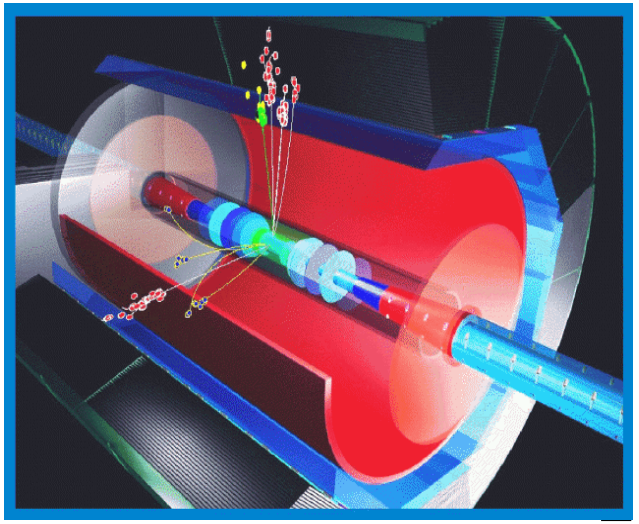
Marlin et al
Introduction to ILC-LDC Simulation and Reconstruction Software

Frank Gaede
DESY

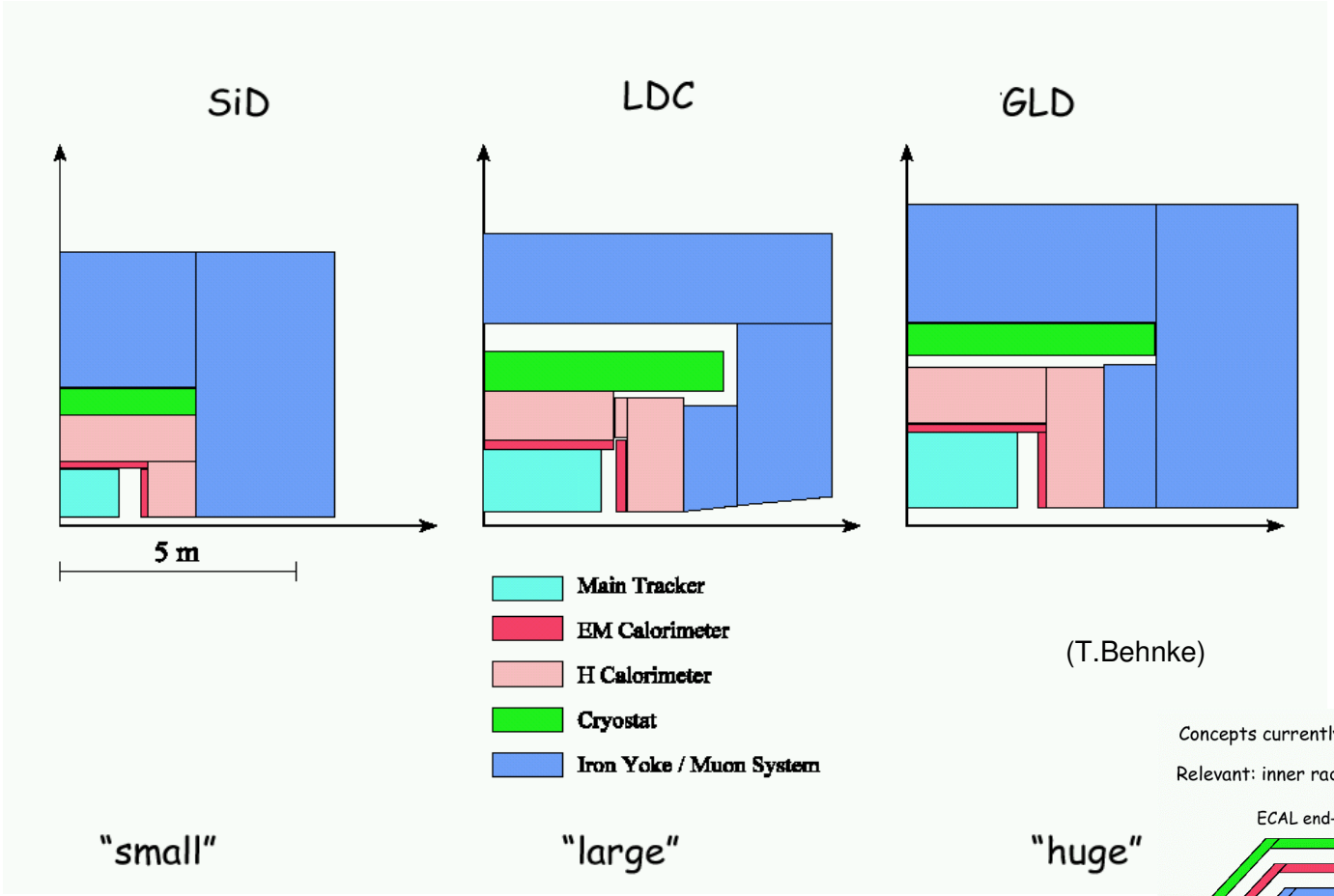
ILC Detector and Physics Workshop, Snowmass
August 14-27, 2005

Outline

- Introduction - overview international software
- Central tools for LDC study
  - LCIO – data model & persistency
  - Simdet – fast simulation
  - Brahms – geant3 full simulation and reconstruction
  - Mokka – geant4 full simulation
  - **Marlin** – C++ reconstruction framework
  - LCCD - conditions data toolkit
  - **GEAR** – geometry description
- Summary & Outlook



Detector Concept Study

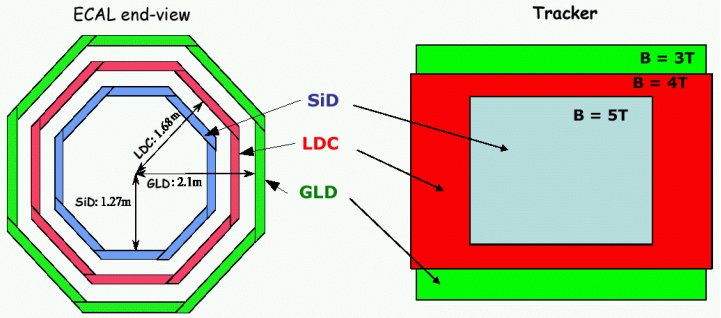


three interregional detector concept studies ongoing

(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 !

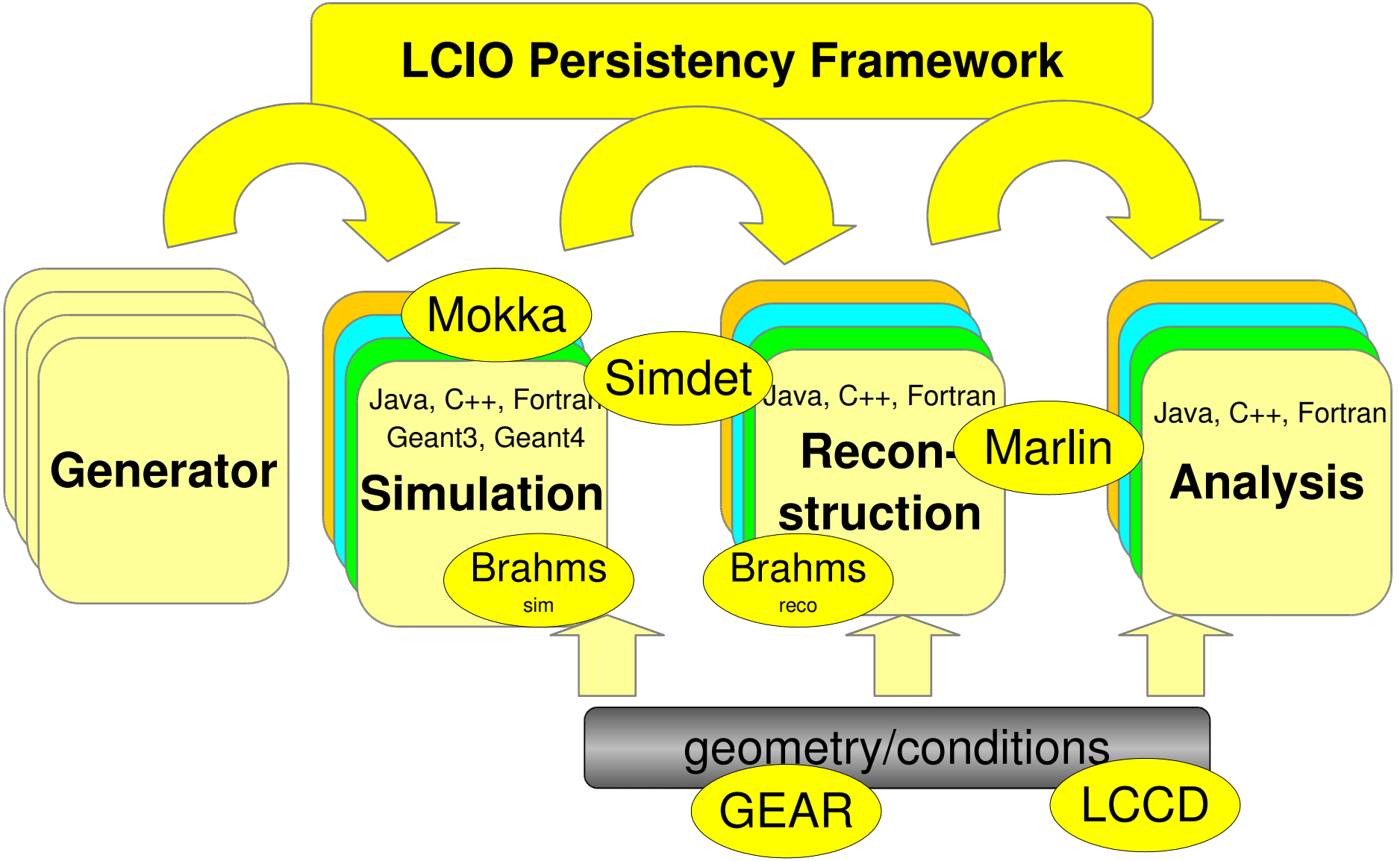
ILC software packages

	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
GEAR	Geometry description	Flexible	C++ (Java?)	XML	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

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
  - (also for detector concept study!)
- 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 !?)**
- common basis: **LCIO**

Overview of LDC software tools



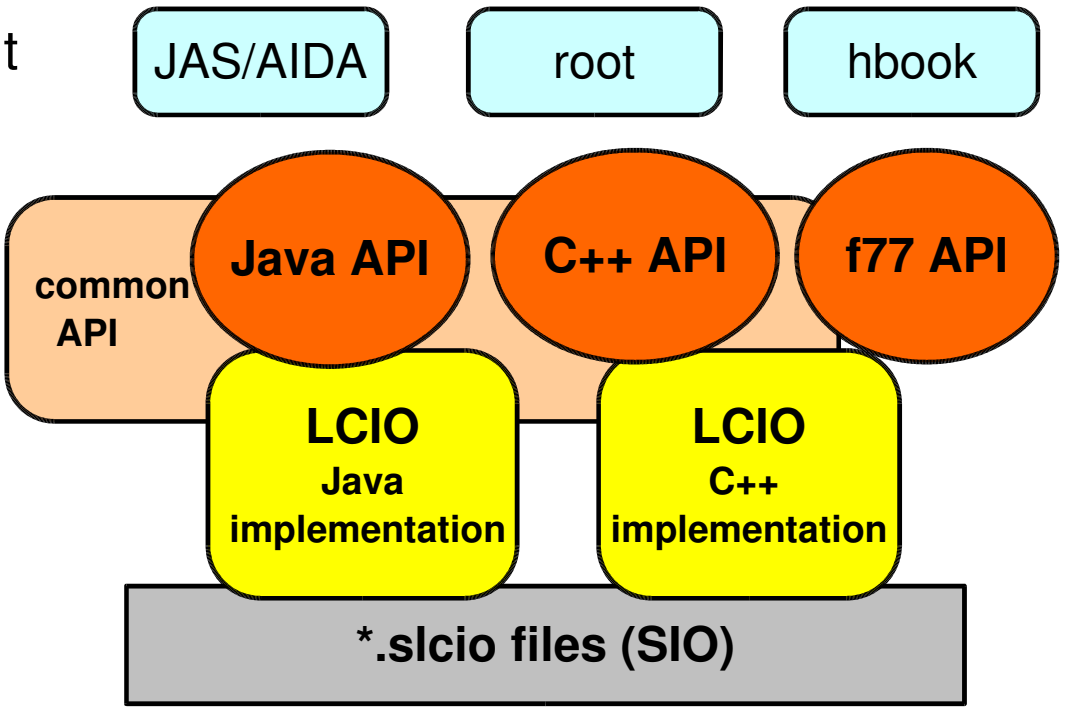
LCIO overview

- DESY and SLAC joined project:
  - provide common basis for ILC software
- Features:
  - Java, C++ and f77 (!) API
  - extensible data model for current and future simulation and testbeam studies
  - user code separated from concrete data format
  - no dependency on other frameworks

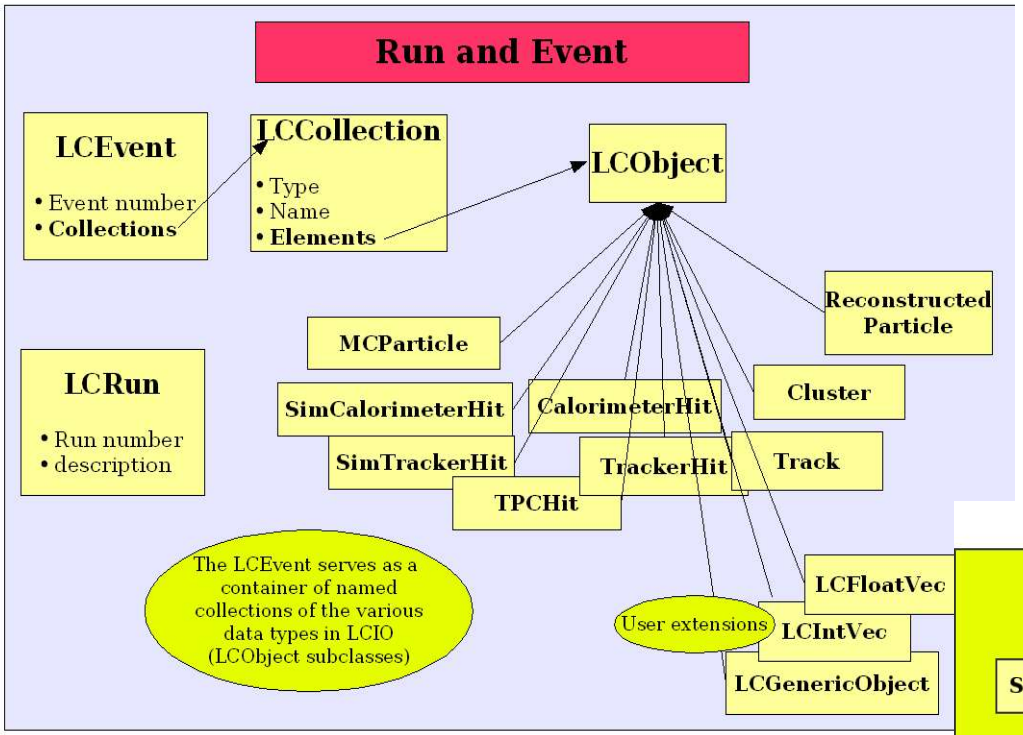
simple & lightweight

now de facto standard for ILC software

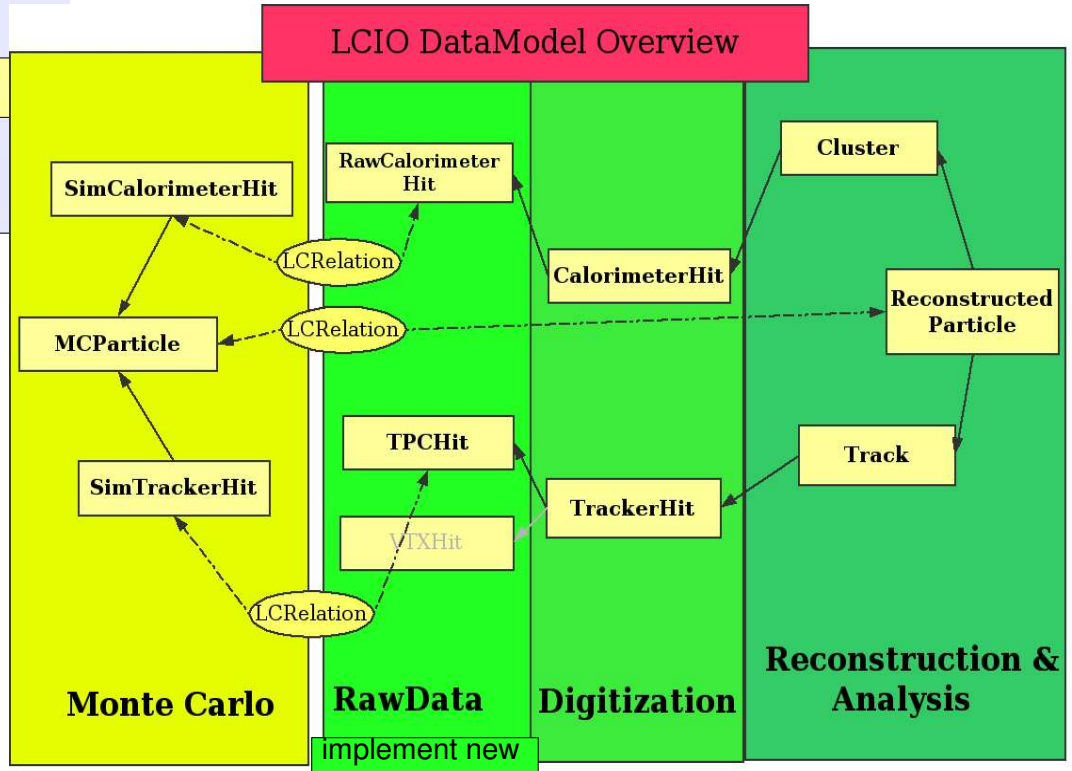
SW-Architecture



LCIO data model



event serves as container of untyped collections



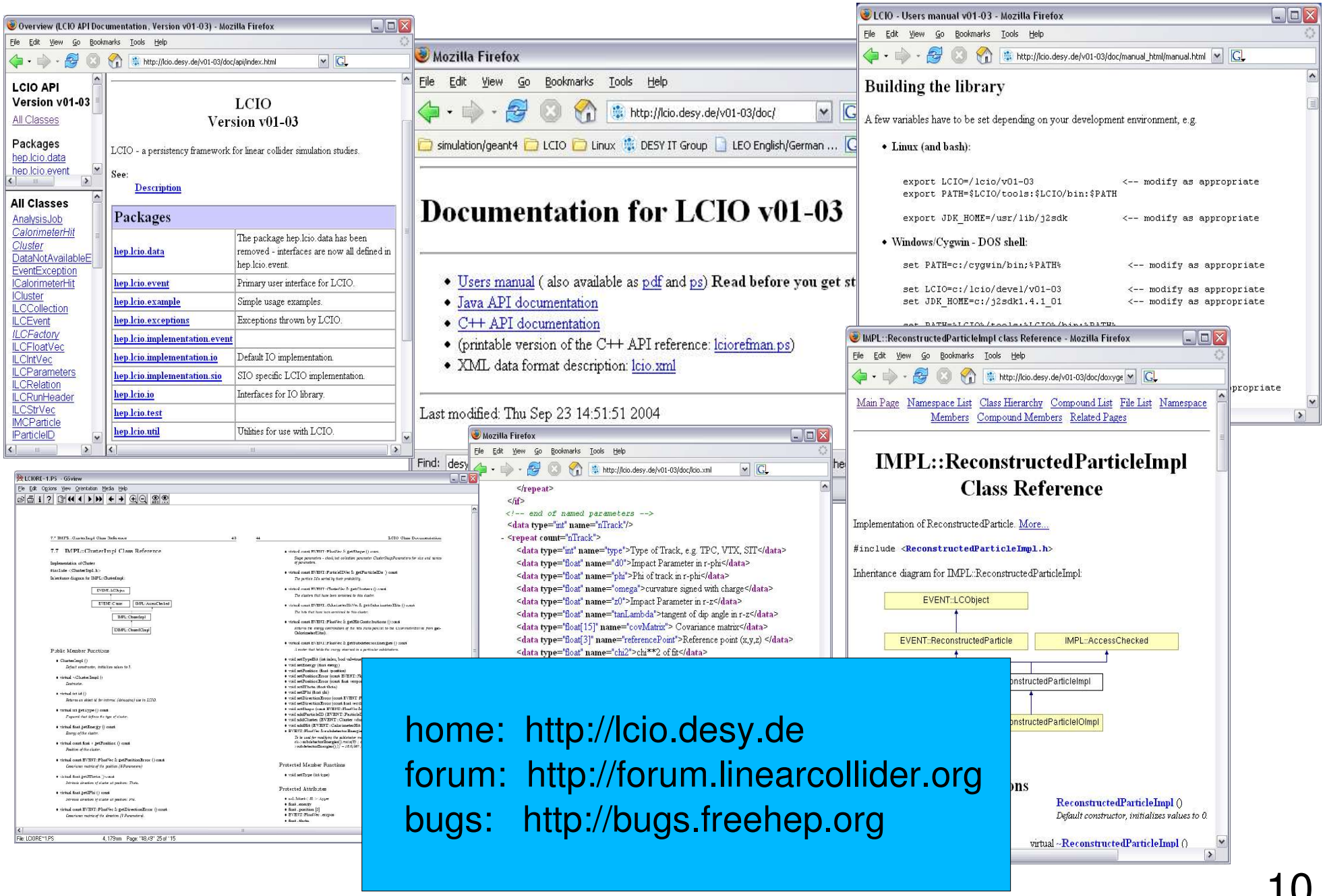
hierarchy of data objects in the event

implement new classes as needed for testbeams !

LCIO status [v01-05]

- changed return values of E, P, m to double for MCParticle and ReconstructedParticle
  - stored are still floats
  - requires some trivial changes (float->double) in code where indicated by the compiler !
- new template UTIL::LCTypedVector<T> for creating std::vector<LCObjectType> from LCCollections
  - allows to use iterators and STL algorithms, e.g. std::for_each()
- added first implementations of generic tracker raw data classes (TPC, VTX, SiliconStrip,...)
  - TrackerRawData, TrackerData, TrackerPulse
- files are downward compatible with LCIO 1.4
- bug fixes
- see $LCIO/doc/versions.readme for more

LCIO on the web



LCIO Future Plans

- current data model fairly complete
  - check usability of data model, e.g. track parameters
    - ongoing through development of reconstruction code !
  - need to define conventions on how to use LCIO
    - collection names, object types, collections to be present in event
- need to make LCIO more convenient (and efficient)
  - decoding of MCParticle information (parent/daughter, ...)
  - inverse relations (get all tracks for one MCParticle)
  - attach user information to LCObjects
- would like to make C++ version more C++ like, e.g. allow to use STL algorithms (templates in general)

- -> had first LCIO meeting here at snowmass ...

Simulation tools: Simdet, Brahms

- **SIMDET**
  - parameterized fast Monte Carlo (f77)
  - tracks + cov. matrix and clusters
  - hard coded geometry: TESLA TDR Detector

writes LCIO

- **Brahms**
  - geant3 full simulation (f77)
  - hard coded geometry: TESLA TDR Detector
  - full standalone reconstruction part (pflow)
    - tracking based on LEP reconstruction code

reads + writes LCIO

for download (cvs web interface)
and more information:
http://www-zeuthen.desy.de/linear_collider

Simulation tools: Mokka

- geant4 based full detector simulation for the ILC
- developed at LLR-Ecole Polytechnique (P. Mora de Freitas, G. Musat)
- http://polywww.in2p3.fr/geant4/tesla/www/\mokka/mokka.html
- some features:
  - steering files for configuration
  - all geant4 physics lists available
  - writes LCIO
  - reads StdHep / ASCII HepEvt
- Geometry
  - MySQL databases
    - geometry parameters
    - one database per subdetector
  - C++ Geometry Drivers
    - one for each subdetector type (e.g. TPC, HCAL)
    - define material and sensitive detector
  - abstract geometry layer: CGA

new features: (v05-01)
- **SID detector model**
- **scalable detector models**

-> detector optimization !!

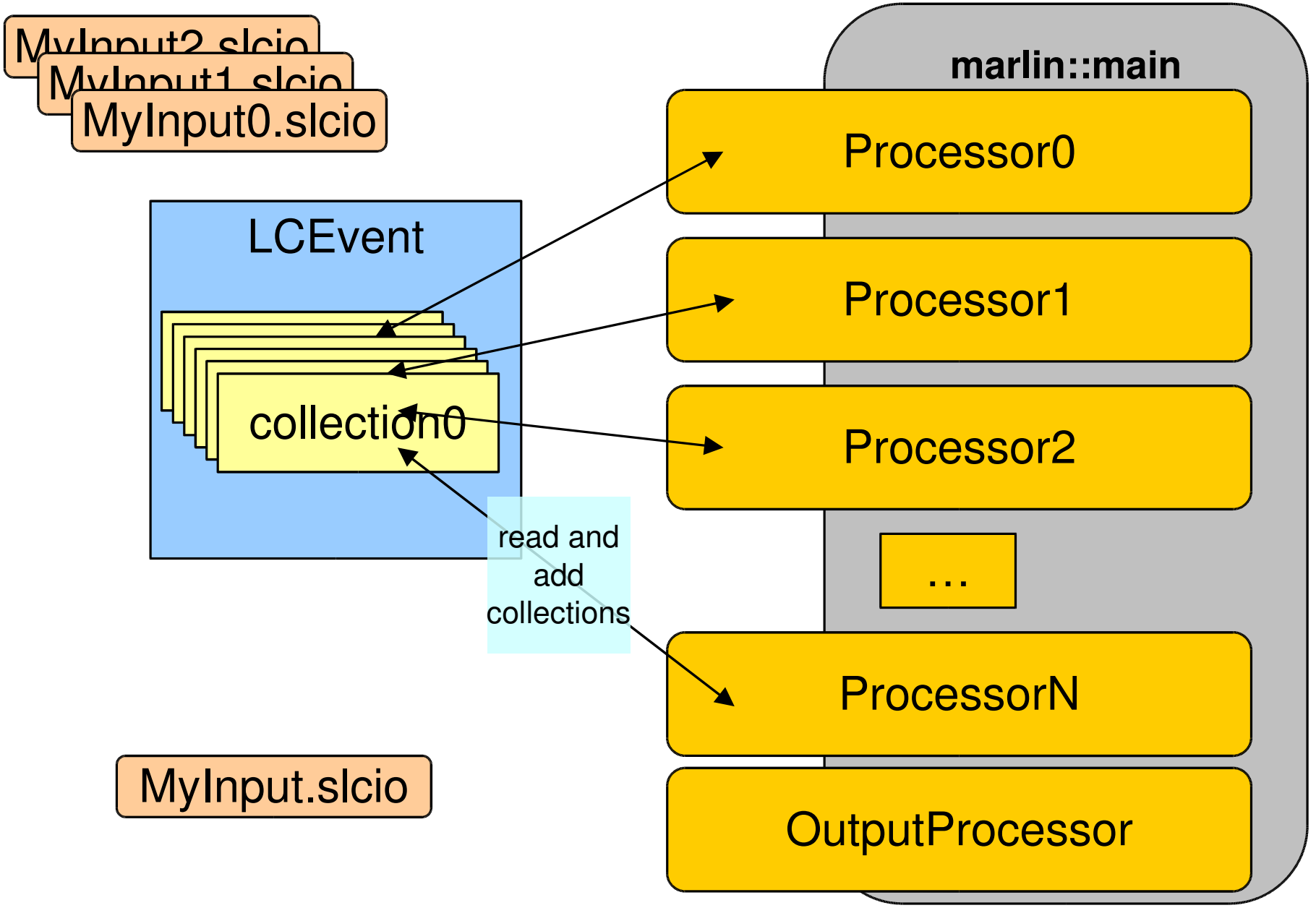
Marlin - Introduction

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

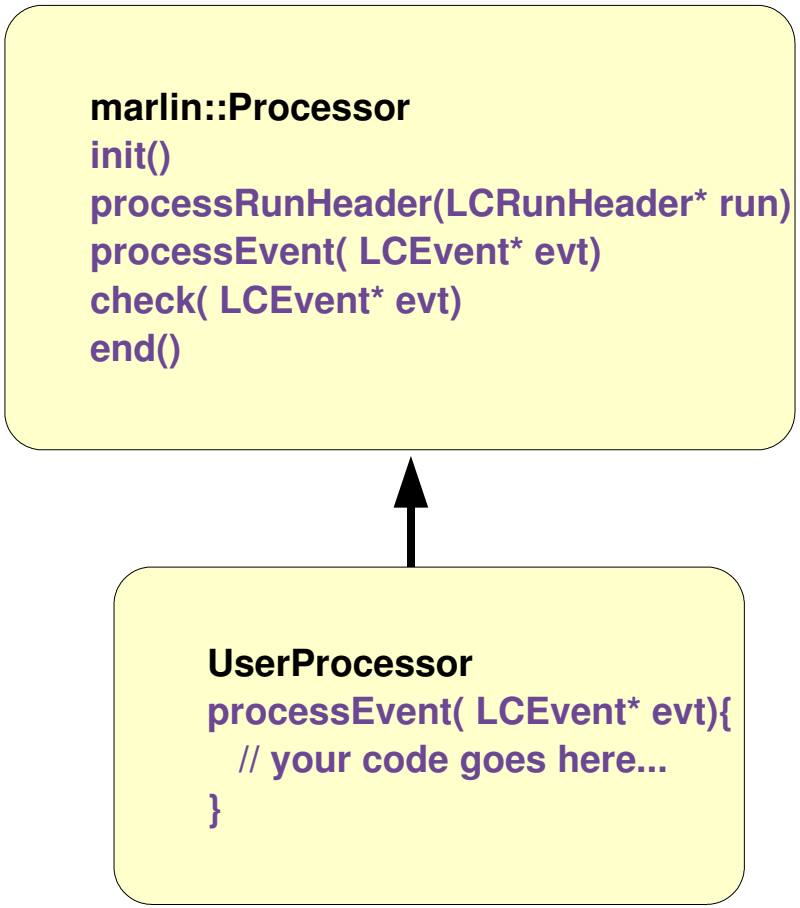
- Application framework:
  - set of classes that provide the core functionality needed in problem domain and provide hooks (callbacks) for specific user code
  - provides main program
  - note: most current experiments that use OO (C++) have application frameworks

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
  - MyProcessor
    - simple example – serves as template for user code
  - ConditionsProcessor New
    - read conditions transparently with LCCD
  - DataSourceProcessor New
    - read non LCIO input, e.g. StdHepReader
  - SimpleFastMCProcessor New
    - fast smearing Monte Carlo
    - needs testing
- fully configurable through steering files:
  - program flow
  - input parameters
    - processor based and global
- self-documenting:
  - MyApplication -l

    will print all available processors with their parameters and example/default values

Marlin - new features I

- v00-09 (details in $MARLIN/doc/release.notes)
- new optional XML steering files
  - based on TinyXml (open source) – included in Marlin
  - same basic structure as old ASCII steering files
  - additional features:
    - allows conditional execution of processEvent() method based on boolean flags set by processors
    - allows grouping of processors with shared parameters
  - 'Marlin -x > steer.xml' creates example steering file with all available processors
- XML can be used in Marlin for other input files

Marlin – example XML steering

```
- <marlin>
  - <execute>
      <processor name="MyAIDAProcessor"/>
      <processor name="MyEventSelection"/>
    - <if condition="MyEventSelection">
        <group name="Tracking"/>
        <processor name="MyClustering"/>
        <processor name="MyPFlow"/>
        <processor name="MyLCIOOutputProcessor"/>
      </if>
    </execute>
  - <global>
      <parameter name="LCIOInputFiles"> simjob.slcio </parameter>
      <parameter name="MaxRecordNumber" value="5001"/>
      <parameter name="SupressCheck" value="false"/>
    </global>
  - <processor name="MyLCIOOutputProcessor" type="LCIOOutputProcessor">
      <parameter name="LCIOOutputFile" type="string">outputfile.slcio </parameter>
      <parameter name="LCIOWriteMode" type="string">WRITE_NEW</parameter>
    </processor>
  - <group name="Tracking">
      <parameter name="NTPCLayers" value="200"/>
      <processor name="MyTrackfinder" type="Trackfinder"/>
    - <processor name="MyTrackfitter" type="Trackfitter">
        <parameter name="Algorithm" value="DAF"/>
      </processor>
    </group>
    <!-- ... -->
  </marlin>
```

- ActiveProcessors replaced by <execute>...</execute> section
- Reconstruct only events that pass the event selection

- Parameters defined as content of <parameter/> tag or as its value attribute

- Processors can be enclosed by <group/> tag
- Parameters in <group/> joined by all processors

Marlin - new features II

- Exceptions (subclasses of lcio::Exception):
  - uncaught Exceptions terminate program with error message (given in constructor)
  - ParseException
    - thrown if steering has wrong syntax
  - SkipEventException
    - can be thrown by any processor
    - skips processEvent() for all subsequent processors
    - printout at end of program (#evts skipped by processor)
  - StopProcessingException
    - can be thrown by any Processor
    - ends program gracefully by calling all end() methods
  - others ?

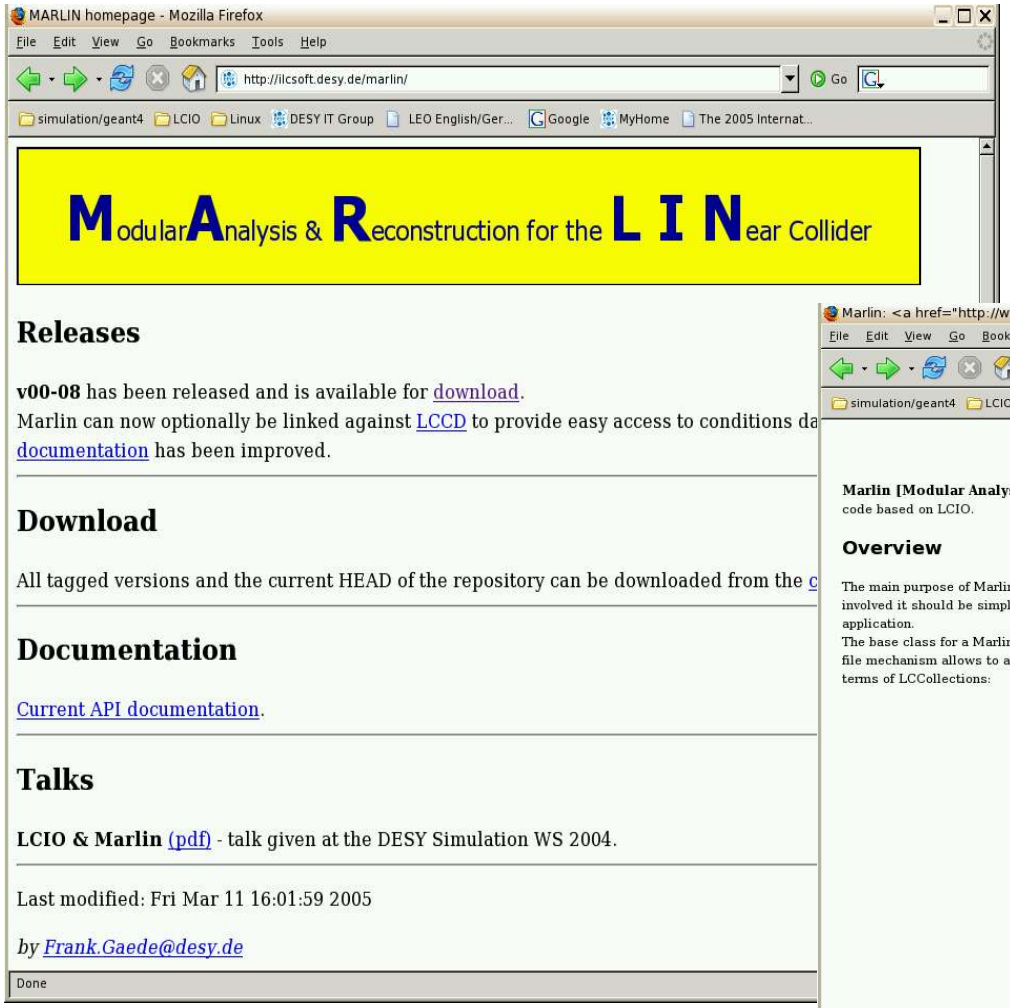
Marlin - new features III

- improved Makefiles
  - make it easier to build against an installed version of Marlin:
    - INCLUDE += `. $(MARLIN)/bin/marlin_includes.sh`
    - LIBS += `. $(MARLIN)/bin/marlin_libs.sh`
    - ensures same optional dependencies are used (LCCD, CLHEP, CondDBMySQL,...)
    - USERINCLUDES, USERLIBS to define additional dependencies
  - global GNUmakefile:
    - allows to build marlin with several packages of processors
      - links all packages in $MARLIN/packages directory
      - packages need to follow $MARLIN/examples/mymarlin structure
  - see workshop DVD

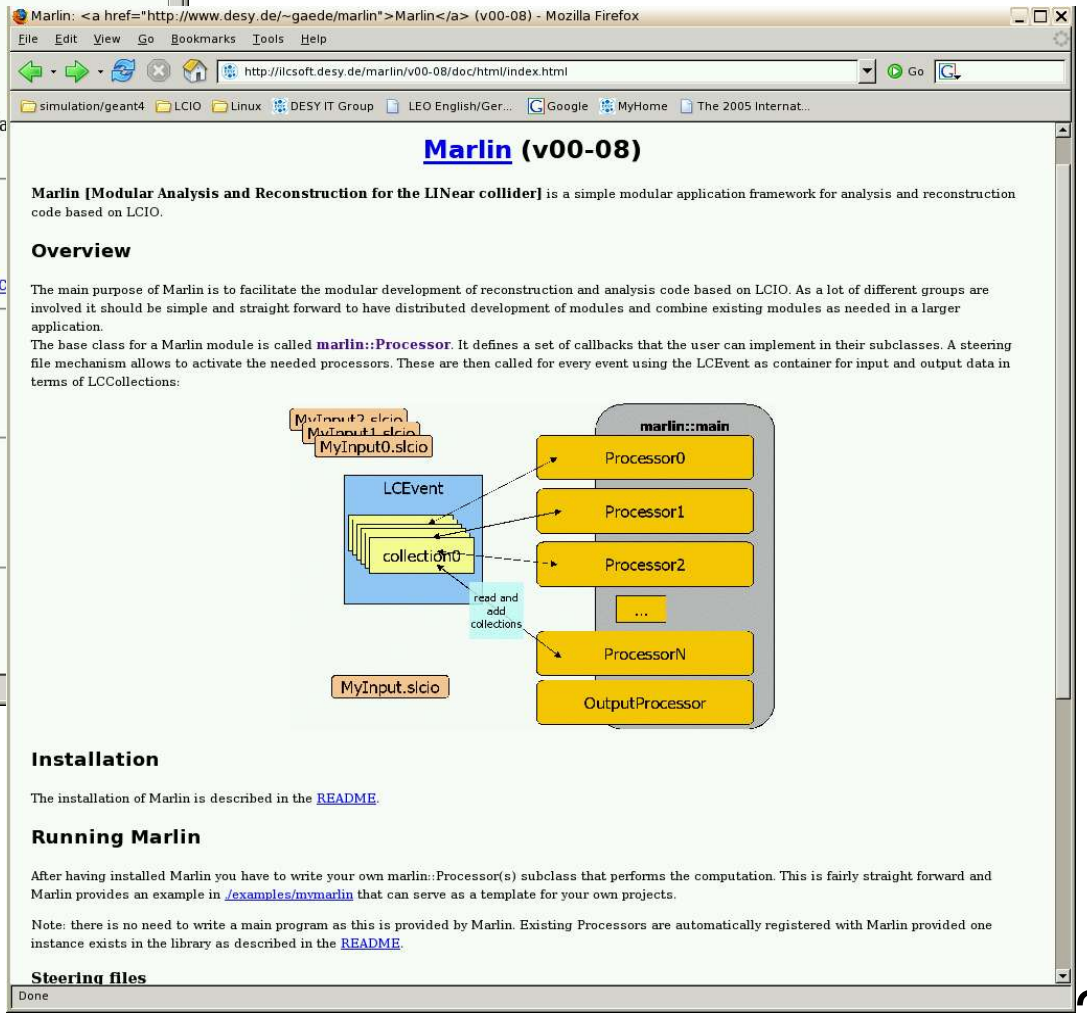
MarlinReco

- **Marlin serves as a framework for the distributed development of a full suite of reconstruction algorithms !**
  - **-> your input is welcome !**
- **(almost) complete set of standard reconstruction in Marlin available: MarlinReco** (see talk by S. Aplin)
  - first version available on DVD “ILC software for LDC”
- uses first implementation of GEAR geometry description
  - Marlin v00-09-01

Marlin on the web



http://ilcsoft.desy.de/marlin



LCCD

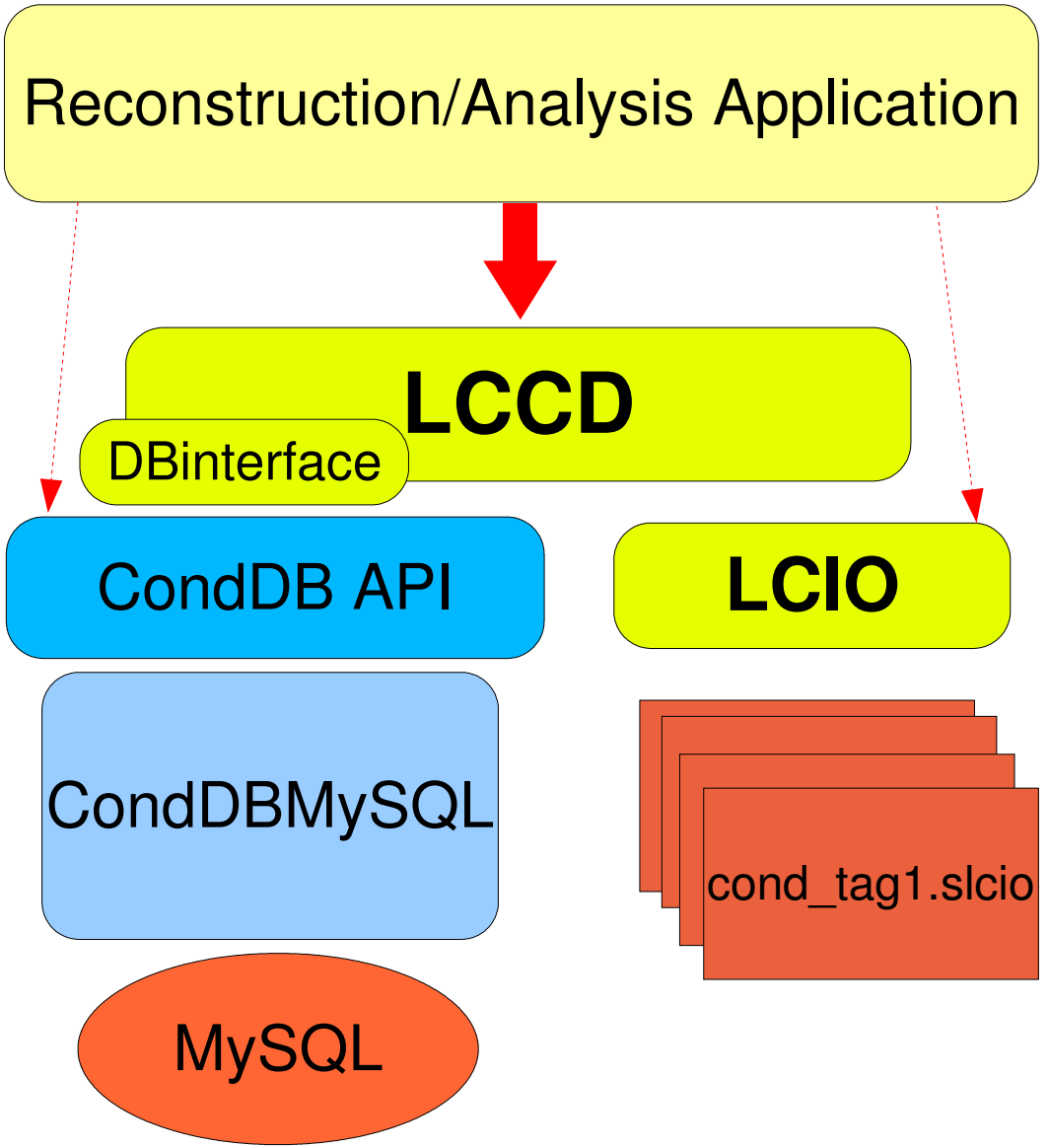
Linear **C**ollider **C**onditions **D**ata Toolkit

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

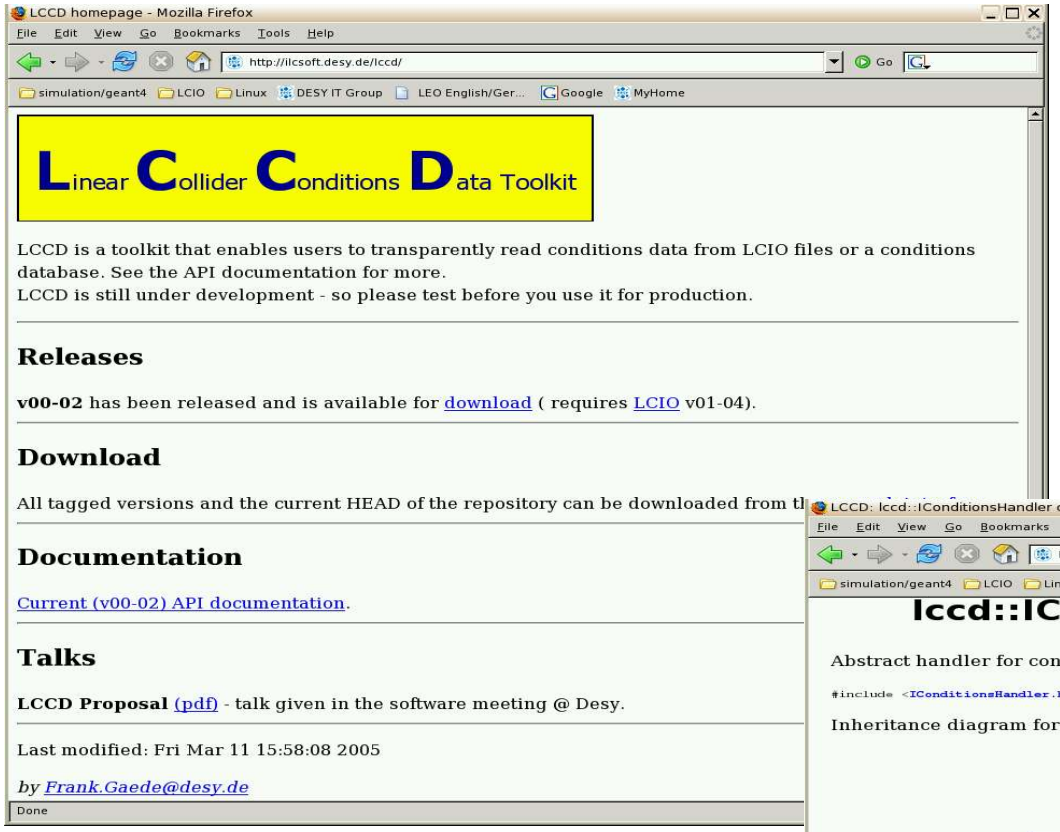
- Conditions Data:
  - all data that is needed for analysis/reconstruction besides the actual event data
    - typically has lifetime/validity range longer than one event
      - can change on various timescales, e.g. seconds to years
    - need for versioning (tagging) (changing calibration constants)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

LCCD features

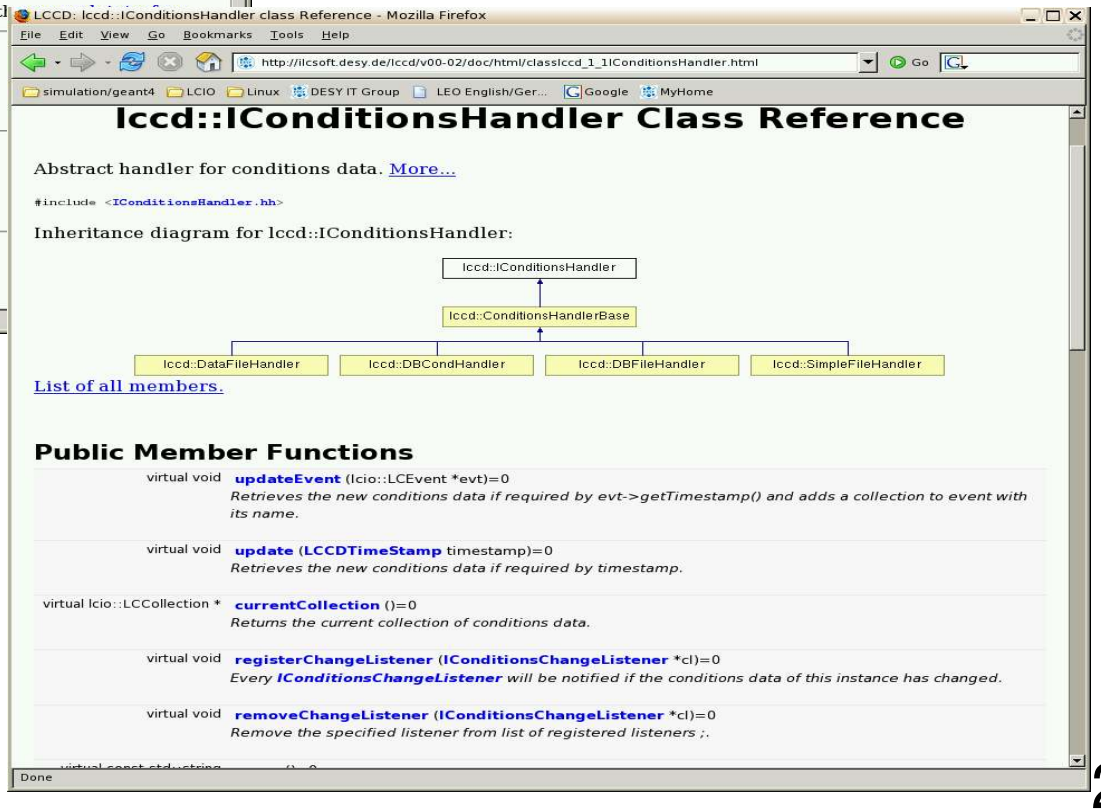
- Reading conditions data
  - from conditions database
    - for given tag
  - from simple LCIO file
    - (one set of constants)
  - from LCIO data stream
    - e.g. slow control data
  - from dedicated LCIO-DB file
    - has all constants for given tag
- Writing conditions data
  - as LCGenericObject 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



GEAR Overview

GEometry **A**PI for **R**econstruction

- Motivation:
  - need well defined geometry definition that
    - is flexible w.r.t different detector concepts
    - has high level information needed for reconstruction
      (different from detailed local description for simulation !)
    - supports 'plug & play' philosophy of processors implementing different algorithms
    - provides access to material properties (radiation/interaction lengths)
- Idea:
  - define abstract interface (a la LCIO C++ and Java ?)
  - concrete implementation based on XML files and CGA

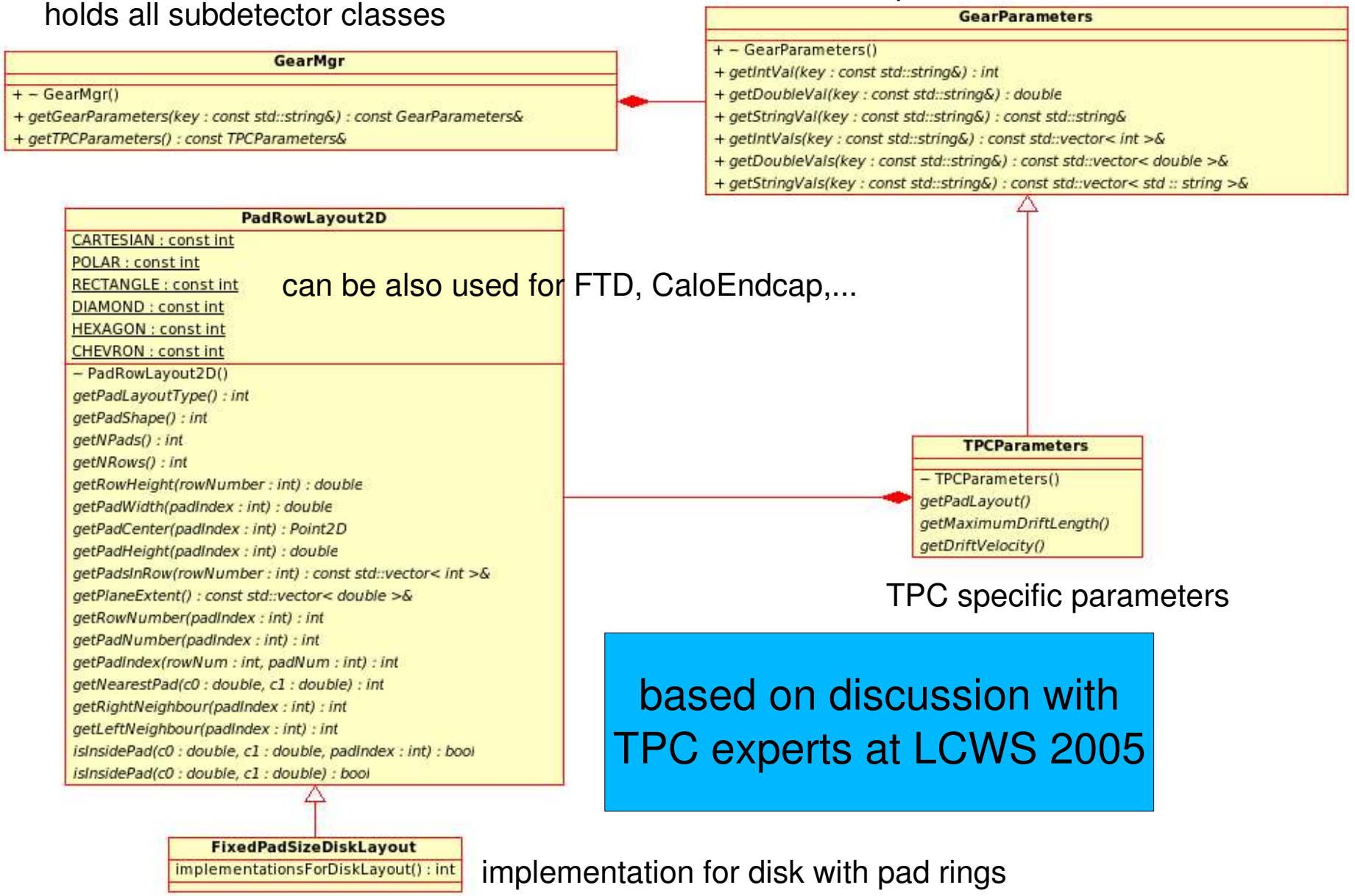
GEAR – Two class types

- Subdetector description
  - high level description of detector shape and readout geometry – one class for every subdetector type, e.g.
    - TPC, Ecal, Hcal (MainCalorimeter), FTD, VTX, SIT, ...
    - defines required attributes - as detailed as necessary but as abstract as possible
    - allows to add additional named attributes
  - use XML files
- Material properties
  - point properties (density, material, radlen,...)
  - distance properties integrated along (straight!?) path
  - use Mokka-CGA interface to geant4 geometry

GEAR – TPC description

holds all subdetector classes

named parameters for additional attributes



can be also used for FTD, CaloEndcap,...

TPC specific parameters

based on discussion with TPC experts at LCWS 2005

implementation for disk with pad rings

GEAR – XML file example

```
- <gear>
  - <!--
      Example XML file for GEAR describing the LDC detector
    -->
  - <detectors>
    - <detector id="0" name="TPCTest" geartype="TPCParameters" type="UNKNOWN" insideTrackingVolume="yes">
        <maxDriftLength value="2500."/>
        <driftVelocity value=""/>
        <readoutFrequency value="10"/>
        <PadRowLayout2D type="FixedPadSizeDiskLayout" rMin="386.0" rMax="1626.0" padHeight="6.2" padWidth="2.2"
        maxRow="200" padGap="0.0"/>
        <parameter name="tpcRPhiResMax" type="double"> 0.16 </parameter>
        <parameter name="tpcZRes" type="double"> 1.0 </parameter>
        <parameter name="tpcPixRP" type="double"> 1.0 </parameter>
        <parameter name="tpcPixZ" type="double"> 1.4 </parameter>
        <parameter name="tpcIonPotential" type="double"> 0.00000003 </parameter>
      </detector>
    - <detector name="EcalBarrel" geartype="CalorimeterParameters">
        <layout type="Barrel" symmetry="8" phi0="0.0"/>
        <dimensions inner_r="1698.85" outer_z="2750.0"/>
        <layer repeat="30" thickness="3.9" absorberThickness="2.5"/>
        <layer repeat="10" thickness="6.7" absorberThickness="5.3"/>
      </detector>
    - <detector name="EcalEndcap" geartype="CalorimeterParameters">
        <layout type="Endcap" symmetry="2" phi0="0.0"/>
        <dimensions inner_r="320.0" outer_r="1882.85" inner_z="2820.0"/>
        <layer repeat="30" thickness="3.9" absorberThickness="2.5"/>
        <layer repeat="10" thickness="6.7" absorberThickness="5.3"/>
      </detector>
    </detectors>
  </gear>
```

additional user defined parameters

compatible with US – compact format

complete file for LDC as used in MarlinReco !

GEAR – material properties

GearDistanceProperties

- GearDistanceProperties()
getMaterialNames(p0 : const Point3D&, p1 : const Point3D&) : const std::vector< std :: string >&
getMaterialThicknesses(p0 : const Point3D&, p1 : const Point3D&) : const std::vector< double >&
getNRadlen(p0 : const Point3D&, p1 : const Point3D&) : double
getNIntlen(p0 : const Point3D&, p1 : const Point3D&) : double
getBdL(pos : const Point3D&) : double
getEdL(pos : const Point3D&) : double

integrated along path:
- straight line or
- true path in B-Field ?

GearPointProperties

- GearPointProperties()
getCellID(pos : const Point3D&) : int
getMaterialName(pos : const Point3D&) : const std::string&
getDensity(pos : const Point3D&) : double
getTemperature(pos : const Point3D&) : double
getPressure(pos : const Point3D&) : double
getRadlen(pos : const Point3D&) : double
getIntlen(pos : const Point3D&) : double
getLocalPosition(pos : const Point3D&) : Point3D
getB(pos : const Point3D&) : double
getE(pos : const Point3D&) : double
getListOfLogicalVolumes(pos : const Point3D&) : std::vector< std :: string >
getListOfPhysicalVolumes(pos : const Point3D&) : std::vector< std :: string >
getRegion(pos : const Point3D&) : std::string
isTracker(pos : const Point3D&) : bool
isCalorimeter(pos : const Point3D&) : bool

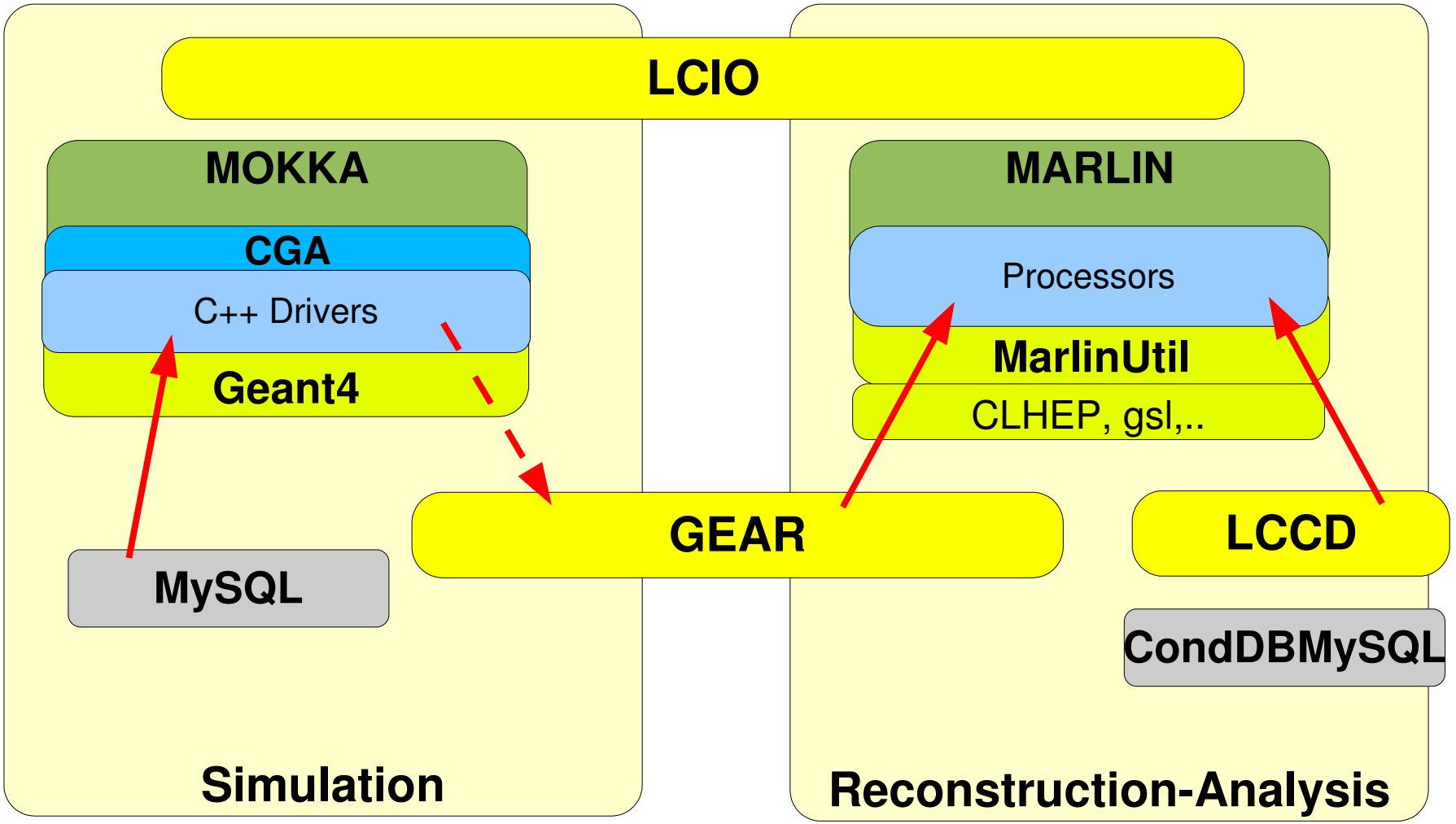
properties at point from geant4 (CGA)

based on discussions at Argonne Simulation Meeting 2004

GEAR status and outlook

- first mini implementation with XML files for current MarlinReco: TPC, Ecal, Hcal
  - XML format 'compatible' with US compact description
  - integrated in Marlin v00-09-01 (see DVD)
  - soon to be released !
- Plans:
  - have discussions at snowmass with other subdetector groups about abstract interface needed: **started with VTX group !**
  - see if other groups are interested in collaborating !?
  - provide implementation of material properties based on CGA/Mokka
  - investigate option of creating GEAR XML files in Mokka geometry drivers

LDC simulation framework



Summary & Outlook

- a fairly complete OO-software framework exists for the LDC study based on Mokka, Marlin, LCIO, LCCD and GEAR
  - ready to start using it for detector concept study !

current version of this software can be found on the DVD "ILC software for LDC "

- has been used to produce LDC events on "International ILC Event DVD" on the DESY – computing **Grid !**
  - LCIO files for SID, LDC, GLC of Zpole, Zh and ttbar events
  - also at http://www-flc.desy.de/snowmassdatadvd

To Do:
• investigate interoperability with other frameworks (ongoing !)
• apply software to other comcepts (ongoing !)
• exchange ideas and collaborate
improve software ...

all software available at portal:
http://www-flc.desy.de/ilcsoft