

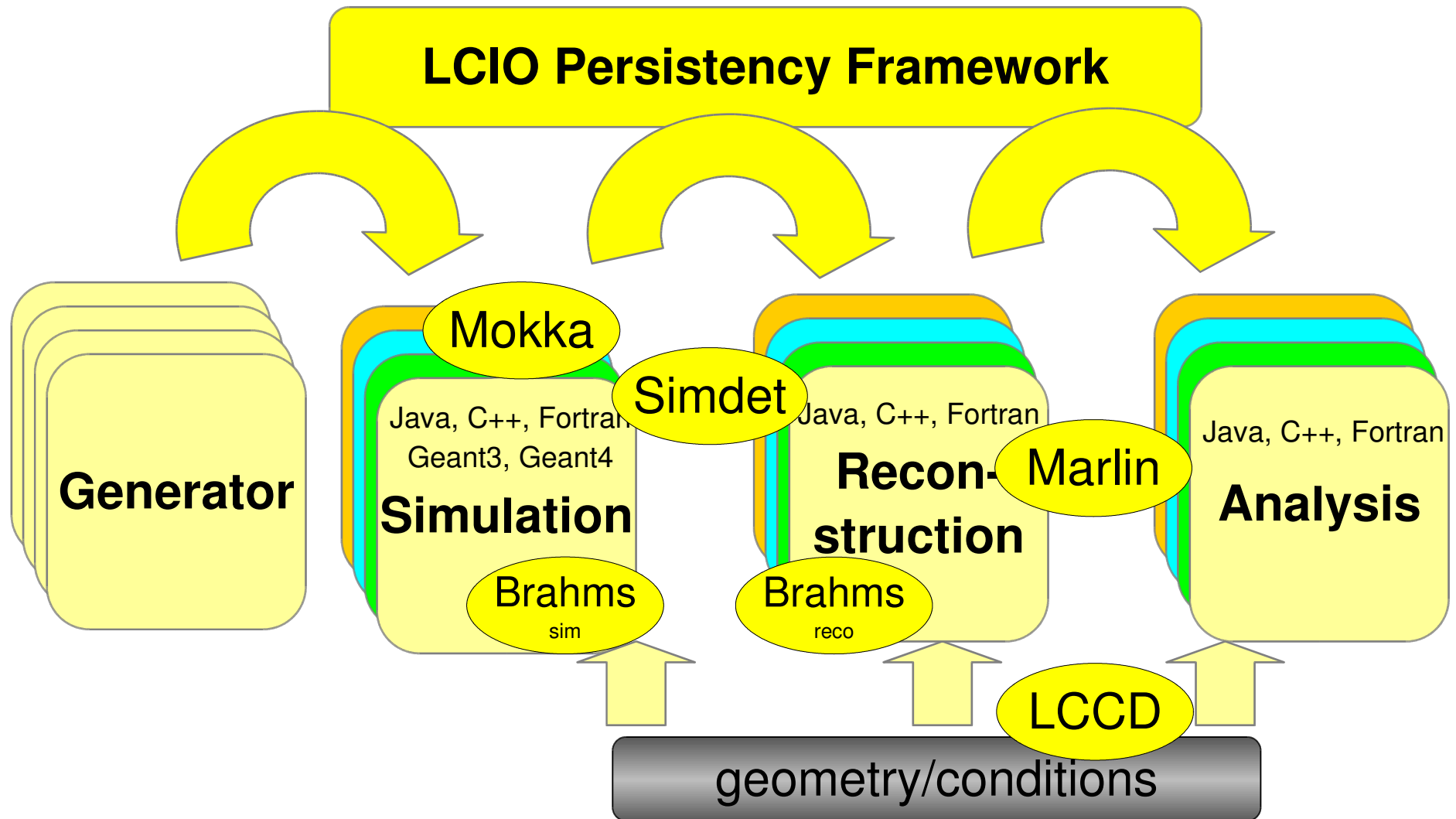
Overview of Simulation and Reconstruction Tools in Europe

Frank Gaede, DESY
LCWS 2005, Stanford, CA
March 18-22 2005

Outline

- Introduction
- LCIO – data model & persistency
- Simulation
 - SIMDET – fast simulation
 - Mokka – geant4
 - BRAHMS – geant3 & reconstruction
- MARLIN – C++ reconstruction framework
- LCCD - conditions data toolkit
- Summary

Overview of software tools

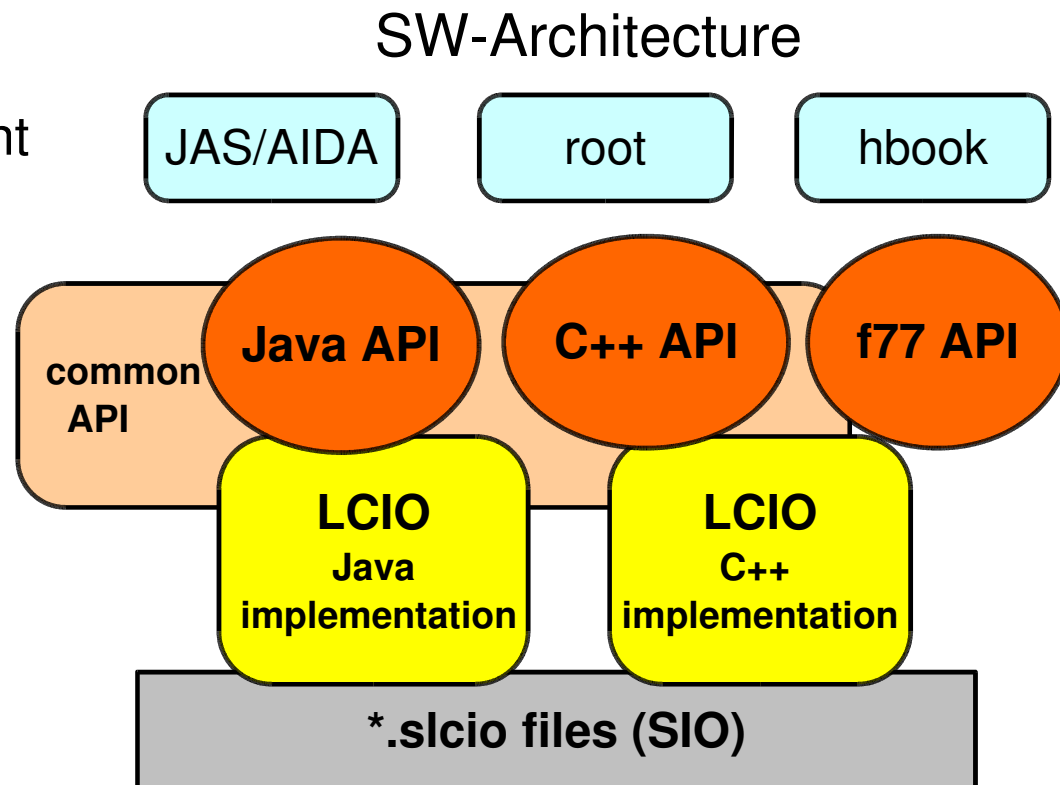


LCIO overview

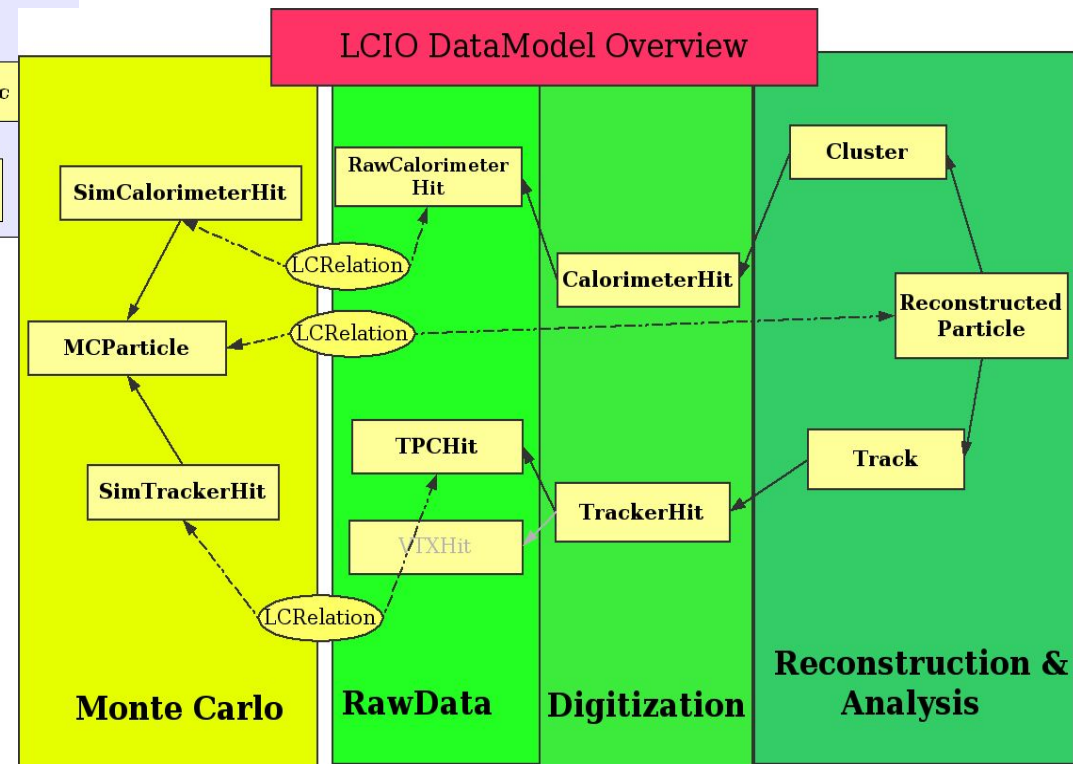
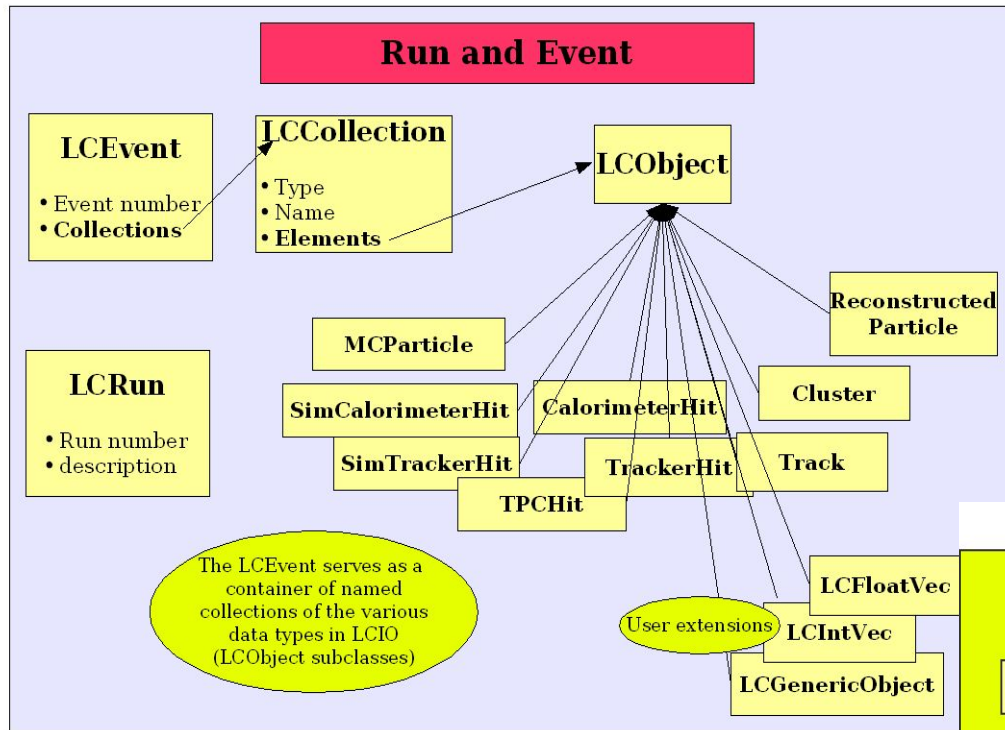
- DESY and SLAC joined project:
 - provide common basis for ILC software
- Features:
 - Java, C++ and f77 (!) API
 - extensible data model for current and future simulation and testbeam studies
 - user code separated from concrete data format
 - no dependency on other frameworks

simple & lightweight

now de facto standard
for ILC software



LCIO data model



LCIO status-new features

- new release v01-04 (March 2005)
- features:
 - support and definition of 64bit time stamps
 - **ns since 1/1/1970 (UTC)**
 - multiple I/O streams in C++
 - LCGenericObjects in Java (user defined data objects)
 - subset collections
 - hold pointers/references to objects already existent in the event, e.g. LeptonCandidates from ReconstructedParticles
 - transient and persistent
 - if persistent, only pointers/references are stored in the file
 - files are downward compatible, LCIO 1.3 can read new files
 - bug fixes
 - improved documentation

LCIO on the web

Frank Gaede, DESY, LCWS 2005, Stanford, CA March 19 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	Default IO implementation.
hep.lcio.implementation.io	SIO specific LCIO implementation.
hep.lcio.io	Interfaces for IO library.
hep.lcio.test	
hep.lcio.util	Utilities for use with LCIO.

Mozilla Firefox

http://lcio.desy.de/v01-03/doc/

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
```

IMPL::ReconstructedParticleImpl Class Reference

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

```
#include <ReconstructedParticleImpl.h>
```

Inheritance diagram for IMPL::ReconstructedParticleImpl:

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

ReconstructedParticleImpl()

Default constructor, initializes values to 0.

virtual ~ReconstructedParticleImpl()

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

Simulation tools

- **SIMDET**

- parameterized fast Monte Carlo (f77) writes LCIO
- hard coded geometry: TESLA TDR Detector

- **Brahms**

- geant3 simulation (f77) writes LCIO
- hard coded geometry: TESLA TDR Detector
- full standalone reconstruction part (pflow) reads + writes LCIO

- **Mokka** (see talk by H.Videau)

- geant4 simulation (C++)
- uses MySQL database for geometry definition writes LCIO
- flexible geometry setup on subdetector basis using C++ drivers, e.g.
 - Tesla TDR Detector with new masks
 - CALICE testbeam prototypes

for download (cvs web interface)
and more information:

http://www-zeuthen.desy.de/linear_collider

full simulation and reconstruction with Mokka & Brahms for Tesla/LDC !

LCCD

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

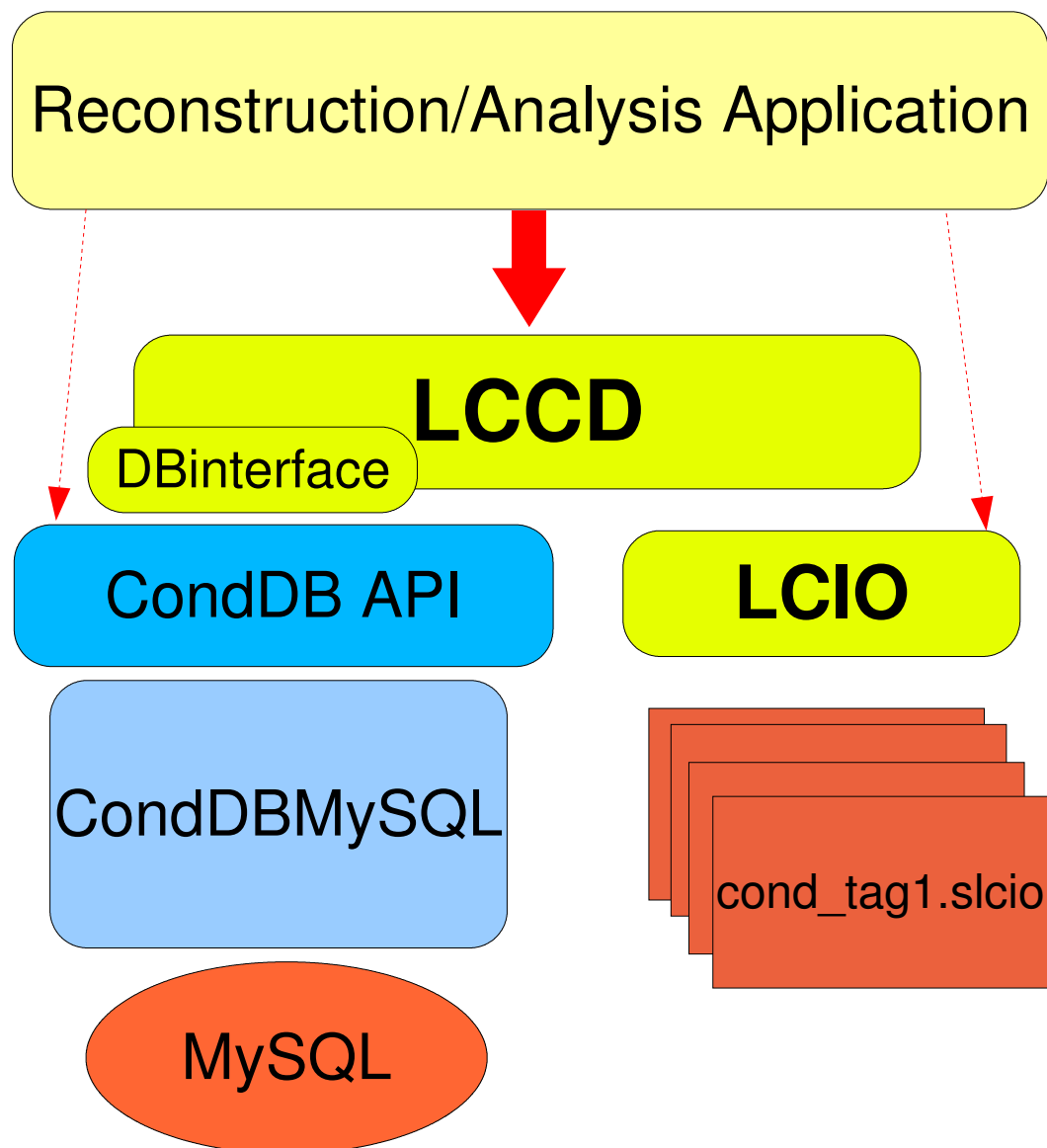
- handles access to conditions data transparently from
 - conditions database (CondDBMySQL)
 - LCIO files
- **Conditions Data:**
 - all data that is needed for analysis/reconstruction besides the actual event data
 - typically has lifetime/validity range longer than one event
 - can change on various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

ConditionsDBMySQL

- open source implementation of CondDB API
 - conditions data interface for LHC (Cern IT)
- developed by Lisbon Atlas group
- features
 - C++ interface to conditions database in MySQL
 - data organized in folder/foldersets
 - objects stored as BLOBs (binary large objects)
 - tagging mechanism similar to CVS
 - outperforms implementation based on Oracle
- status
 - no active development - but bug fixes
- remark: first tests suggest that software runs stable
 - need extended tests before used in production environment

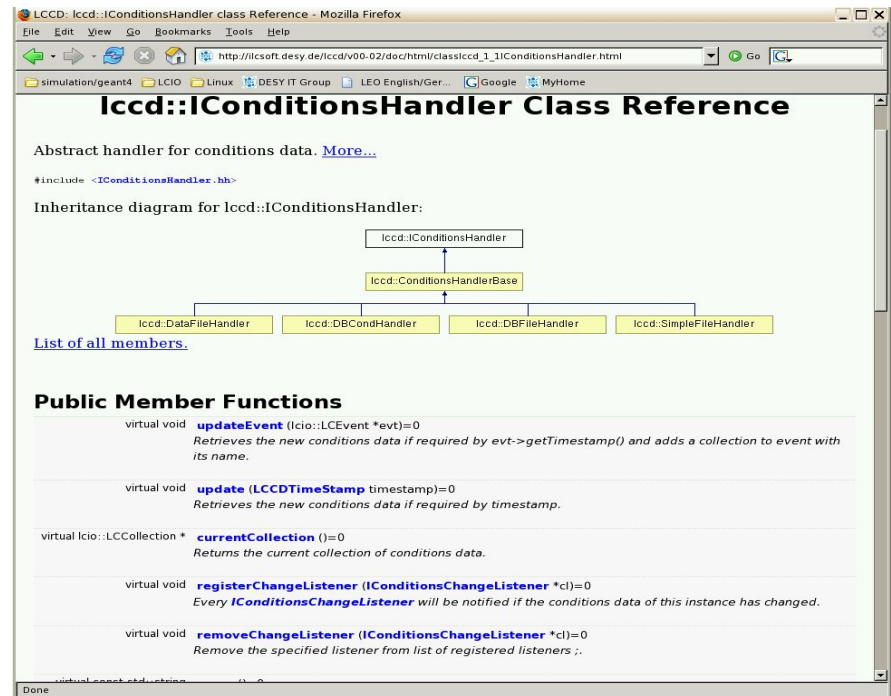
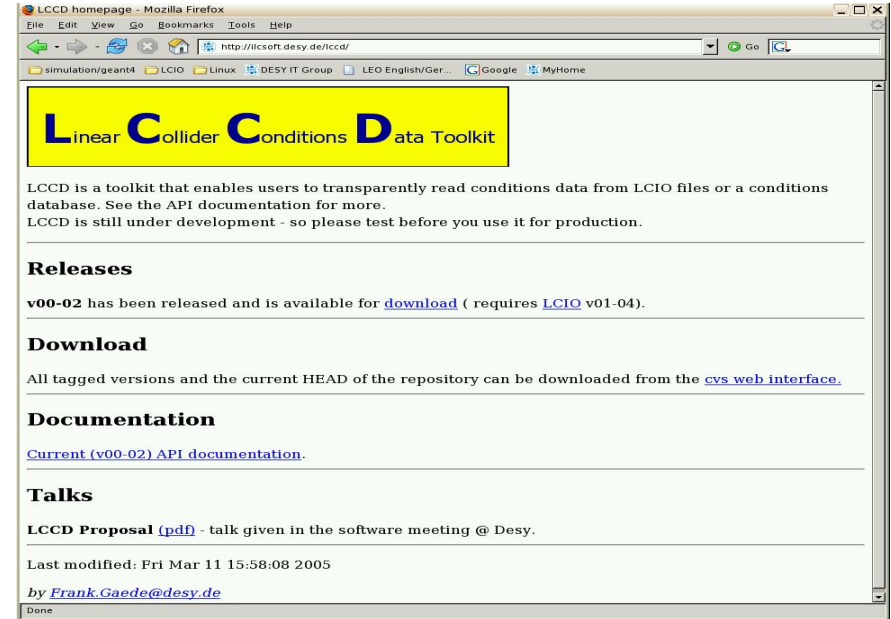
LCDD features

- **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 Status

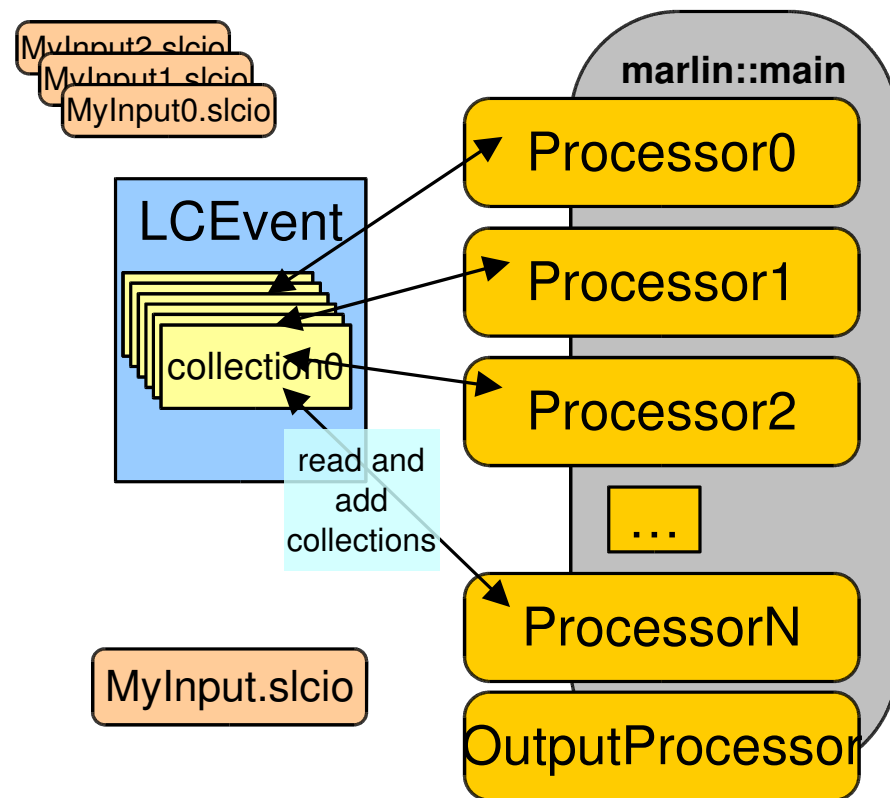
- v00-02 released
- fairly complete functionality
- passes simple tests
- example/test code
- complete API documentation
- available via cvs web:
 - <http://ilcsoft.desy.de/lccd>
- feedback welcome
- CALICE will use LCCD for testbeams



Marlin

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**
- software modules called Processors
- provides main program !
- provides simple user steering:
 - program flow (active processors)
 - user defined variables
 - per processor and global
 - input/output files



Marlin overview

- core functionality

- **AIDAProcessor**

- for easy creation of histograms, clouds, ntuples

- **OutputProcessor**

- **ConditionsProcessor**

- read conditions transparently with LCCD

- **OverlayProcessor**

- event mixing (under development)

- **MyProcessor**

- simple example – serves as template for user code

```
marlin::Processor  
init()  
processRunHeader(LCRunHeader* run)  
processEvent( LCEvt* evt)  
check( LCEvt* evt)  
end()
```

UserProcessor

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



Marlin serves as a framework for the distributed development of a full suite of reconstruction algorithms !

It can also be used for small standalone analysis jobs.

Marlin users

- CALICE testbeam software
 - DigiSim (G.Lima)
 - Ganging and Calibration (R.Poeschl)
- Analysis software
 - LCLeptonFinder (J.Samson)
 - JetFinder (Th.Kuhl)
 - ThrustFinder (Th. Kraemer)
- Reconstruction software
 - wrapper for Brahms-Tracking code (S.Aplin)
 - clustering and pflow – SNARK in C++ (A.Raspereza)
 - clustering algorithms (Ch. Ainsley, G. Mavromanolakis)
- probably others ...
- **aim: have (at least one) complete set for standard reconstruction in C++ soon !**
- need common repository or web portal to provide entry point for users to download and configure their marlin application !

Marlin status

- v00-08 released
 - ConditionsProcessor
 - improved Makefiles
 - improved processor parameters
 - available via cvs web @
- new homepage:
 - <http://ilcsoft.desy.de/marlin>
 - (old download page obsolete!)
 - improved documentation
 - overview & API doc

The image shows two browser windows. The top window is the MARLIN homepage at <http://ilcsoft.desy.de/marlin/>. It features a yellow banner with the text "Modular Analysis & Reconstruction for the LINear Collider". Below the banner are sections for "Releases" (announcing v00-08), "Download", "Documentation" (with a link to "Current API documentation"), and "Talks".

The bottom window shows the "Marlin (v00-08)" documentation page. It includes an "Overview" section describing the framework's purpose and structure. A diagram illustrates the data flow: "MyInput.slcio" feeds into "LCEvent", which then feeds into "collection0". "collection0" is connected to a "marlin::main" block containing "Processor0", "Processor1", "Processor2", and "ProcessorN". "collection0" also has a "read and add collections" output. The "marlin::main" block outputs to "OutputProcessor".

The documentation page also includes sections for "Installation", "Running Marlin", and "Steering files".

Summary & Outlook

- fairly complete software chain exists for Tesla/LDC studies:
 - fast simulation - SIMDET
 - full simulation geant4/geant3 - Mokka/Brahms
 - reconstruction workhorse (Brahms) still f77
 - new C++ reconstruction framework (Marlin) under distributed development
 - conditions data toolkit – LCCD
 - Calice testbeam will exercise the software chain
- all tools use LCIO !
- still a lot of work to do:
 - have complete Marlin based reconstruction
 - geometry description for reconstruction
 - make tools more flexible (other detector concept studies)
 - investigate options for interoperability with other frameworks
 - geometry description /Java-C++ interfacing / conditions ...