Software for the CALICE Project(s)

Roman Pöschl
DESY Hamburg
CALICE Collaboration

Applying/Testing ILC Software Tools in ILC Detector Development

LCWS05 Stanford/USA March 2005

Introduction

CALICE collaboration is preparing/performing large scale testbeam
(See Calo Session)
Primary sites DESY and FNAL

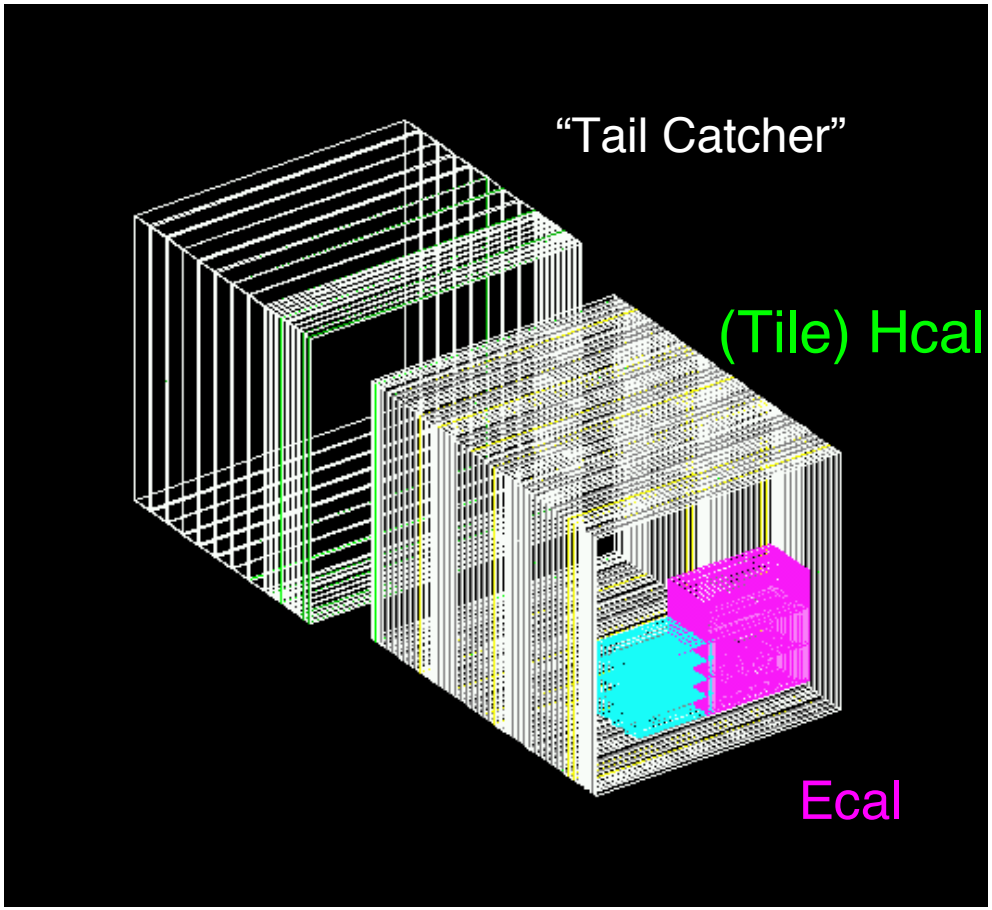
Testbeam program poses software “challenges”

- Detailed simulation of testbeam setup

- Data processing from Raw Data to final Clusters in user friendly way

- Handling of Conditions Data

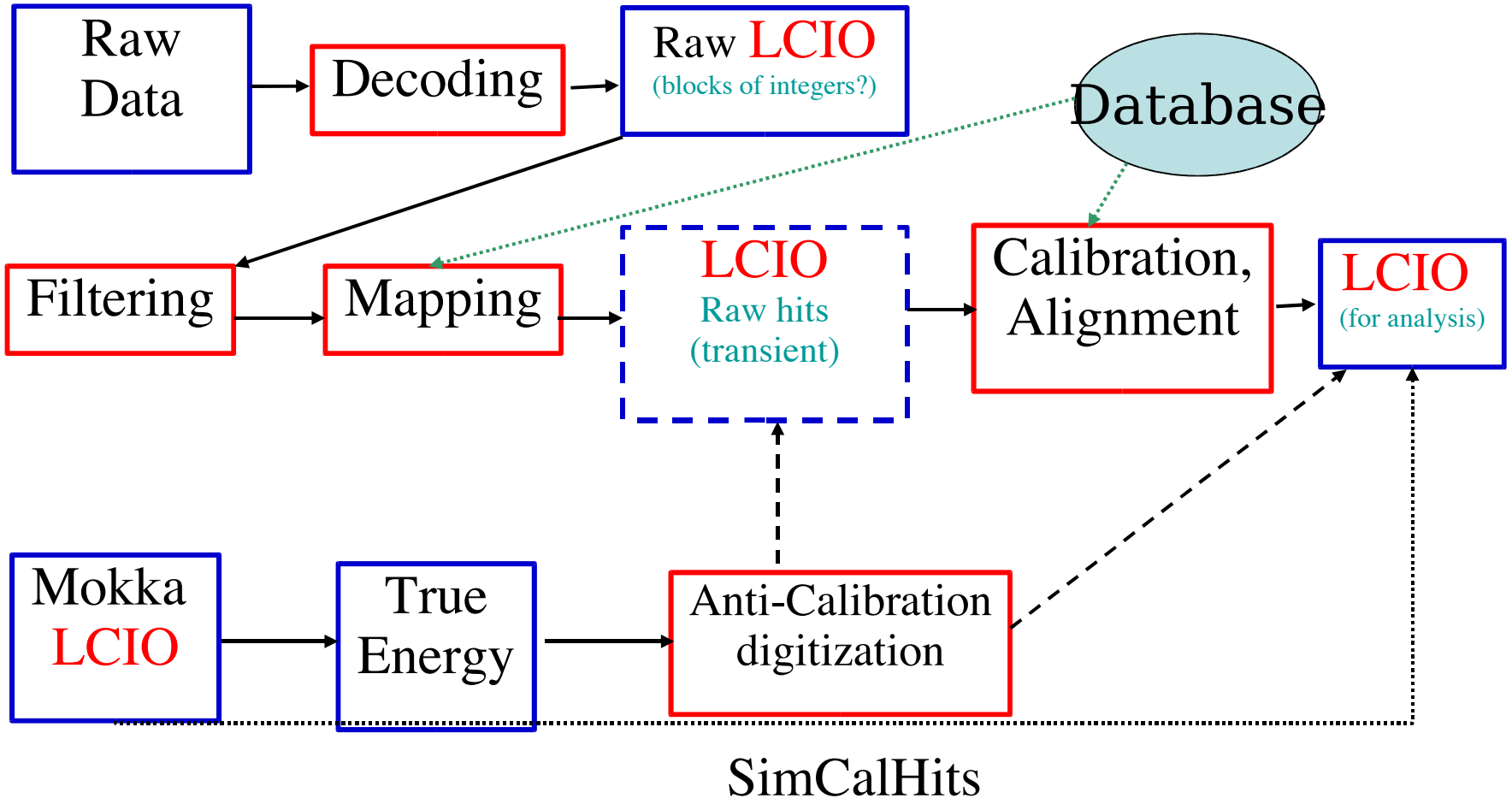
Testbeam software is to be developped within ILC framework



Complete testbeam setup available in Mokka

Dataflow in CALICE Testbeam

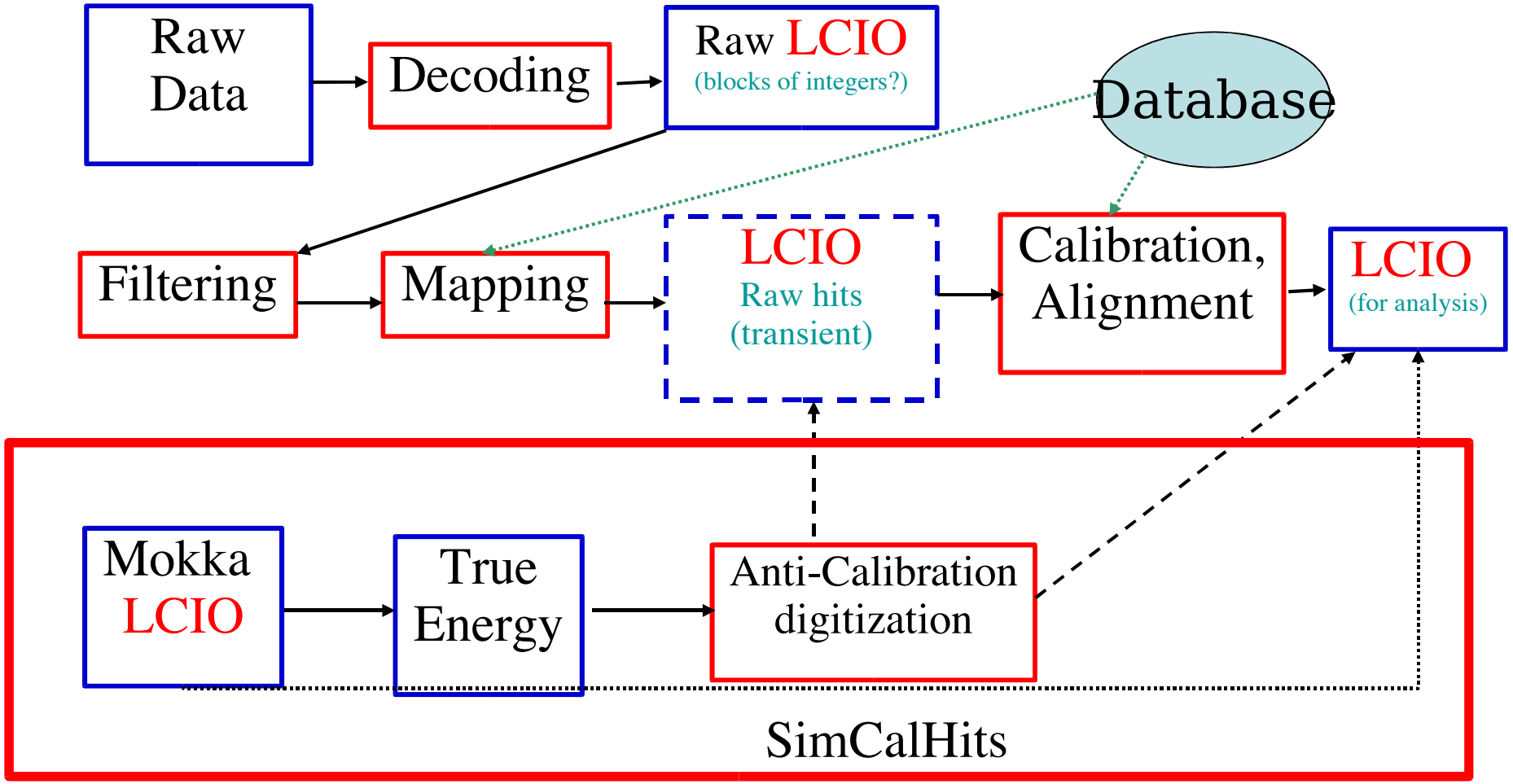
LCIO as backbone of Testbeam Analysis



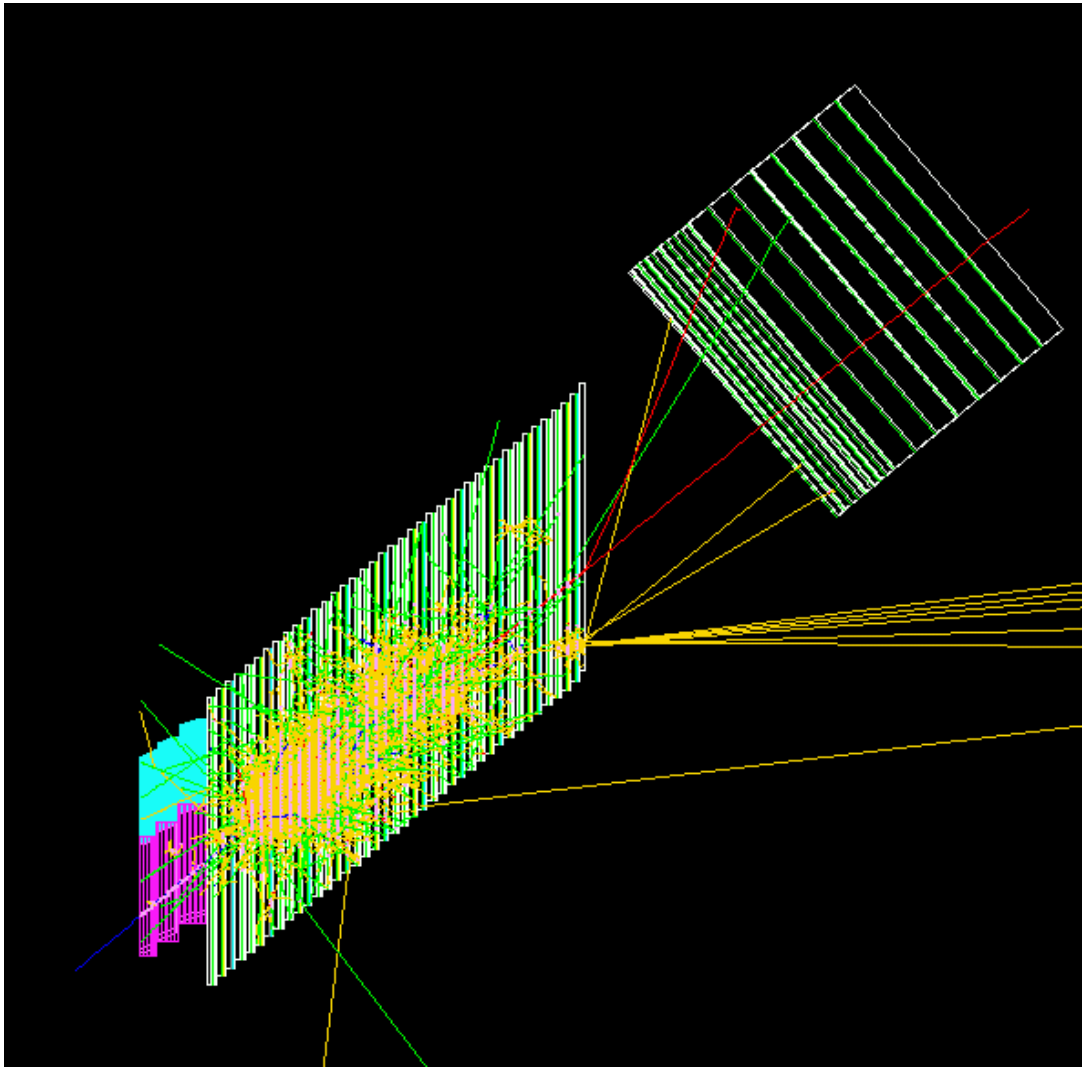
Realization of Scheme ?

Dataflow in CALICE Testbeam

LCIO as backbone of Testbeam Analysis

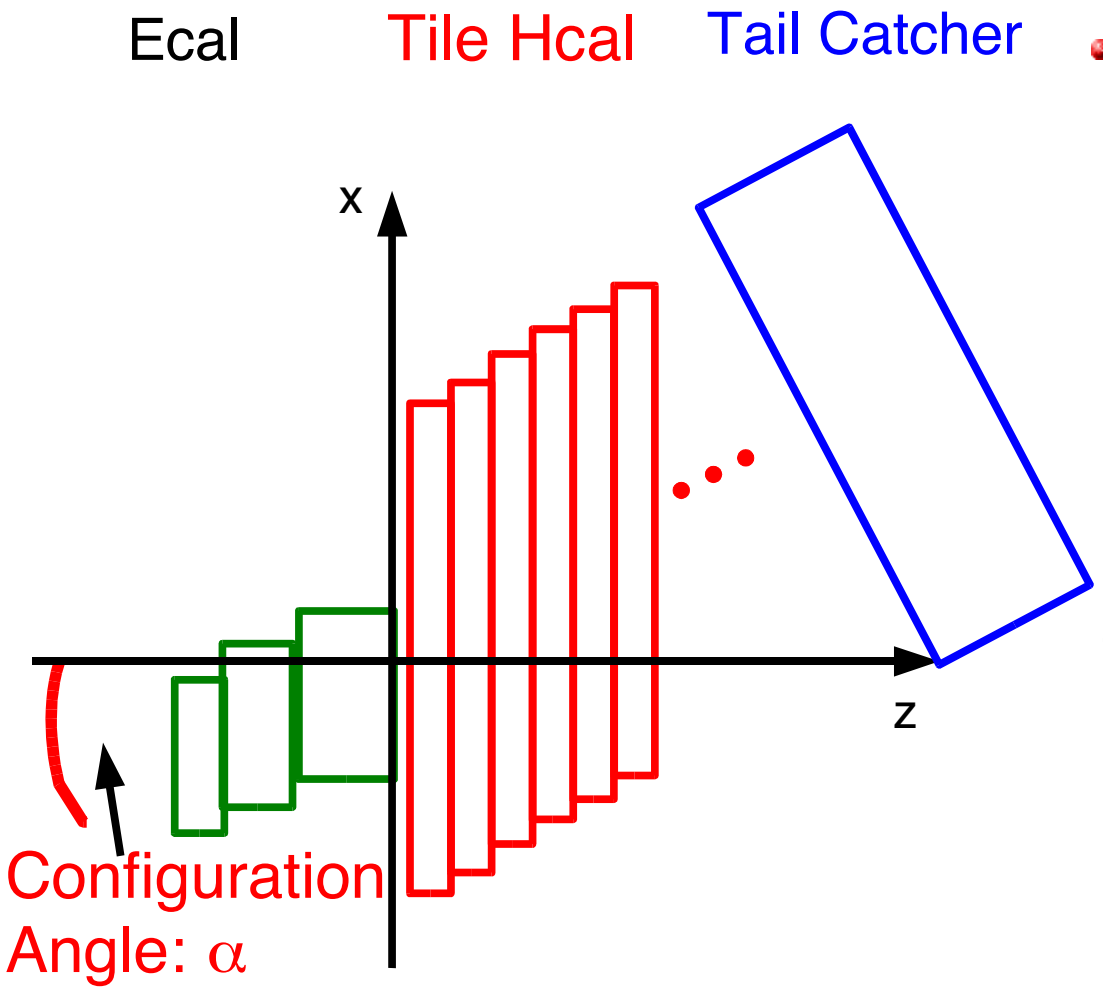


Testbeam Simulation with Mokka



Study of different impact angles

Implementation Issues



Detectors aligned along +z

- Implementation allow for different configuration angles

The layers of the detectors are shifted and the beam is rotated

Implementation is part of Mokka releases since Mokka 03-02

Geometry/tbeam area:
names ...03... (e.g. Tbhcal03.cc)

- Communication of parameters between drivers ?

=> Testbeam needs have driven current major update of Mokka Release 4.0

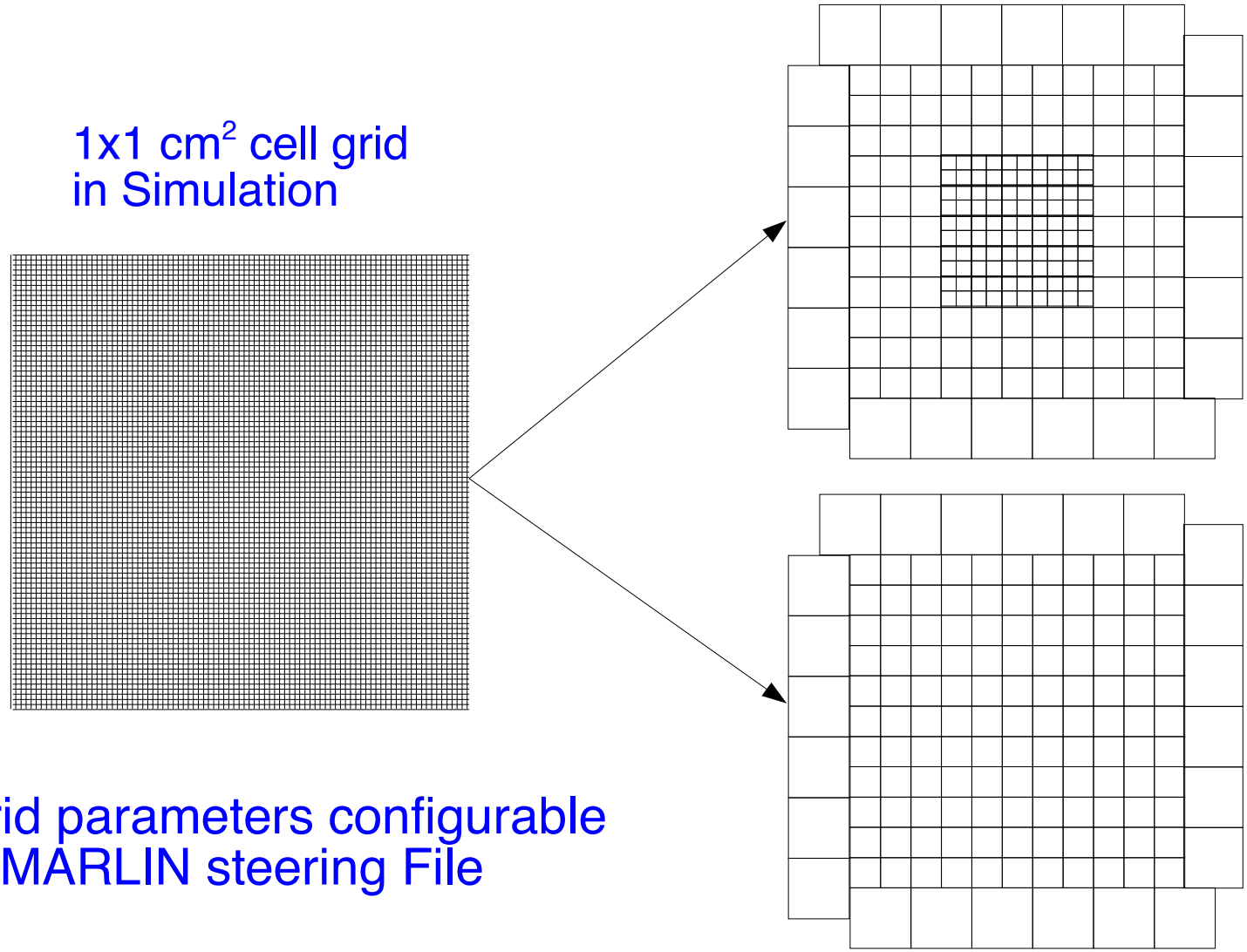
Relation between Subcomponents can be defined in SetupObject (see talk by Henri Videau)

MARLIN in CALICE Project – The Ganging Processor

Combines calo cells from simulation output into 'real' cells

Real Cell Grid(s) of Tile Hcal

1x1 cm^2 cell grid
in Simulation

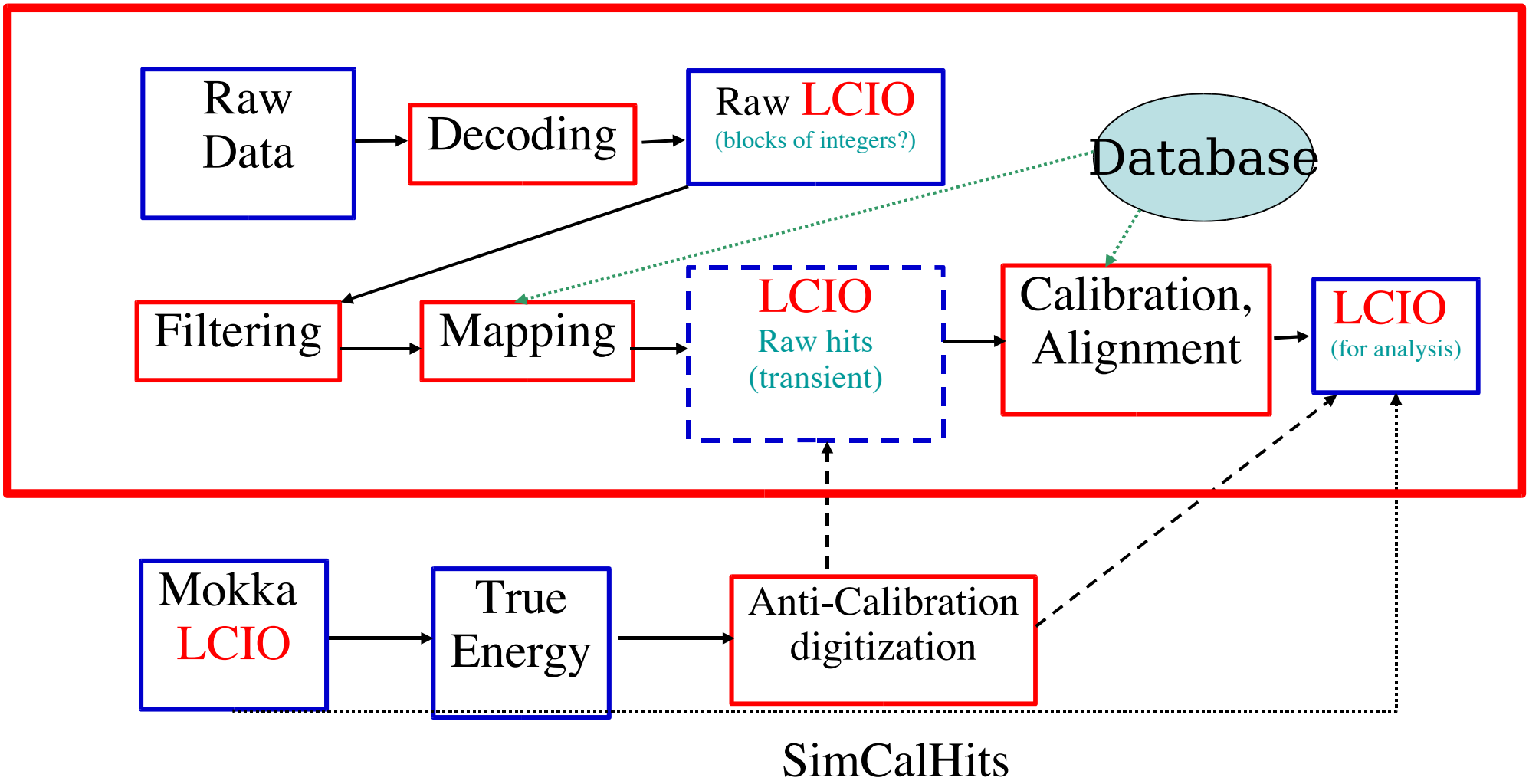


Grid parameters configurable
in MARLIN steering File

All analysis steps are to be realized within MARLIN frame

Dataflow in CALICE Testbeam Program

LCIO as backbone of Testbeam Analysis

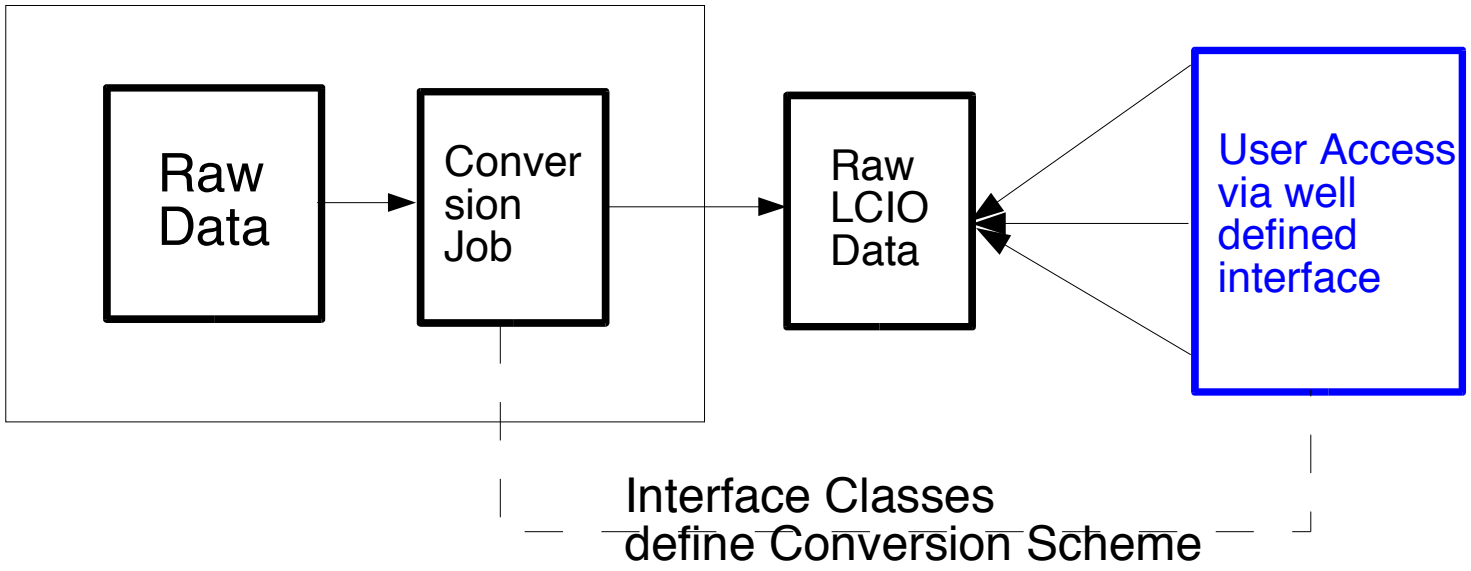


Raw Data Conversion to LCIO

Collaboration: G. Mavromanolakis,D. Ward, P. Dauncey, F.Gaede, R.P.

Main Strategic Decision: Raw Data should be available in LCIO Format

Requirements: - 'Intelligent' Conversion from Raw Data to LCIO Raw Data
- Provide all Info on Raw Data also in LCIO Raw Data in a user friendly way



Interface is to be completely decoupled from online software

Dedicated Interface Classes are defined using LCIOGenericObjects

Handling of Conditions Data

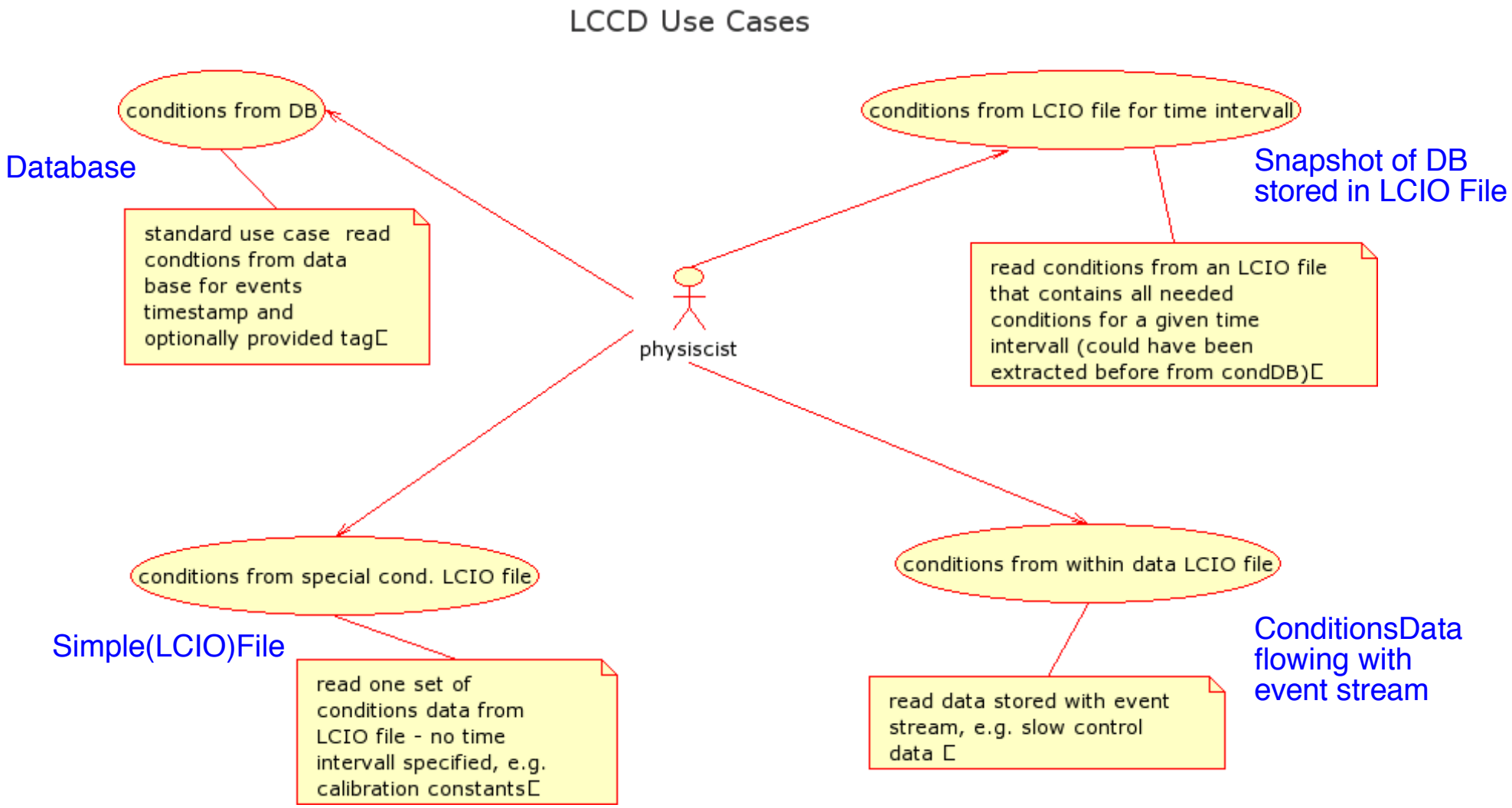
Application of LCCD for Testbeam Data Processing

- LCCD – Linear Collider Conditions Data Framework:
  - Software package providing an Interface to conditions data
    - database
    - LCIO files

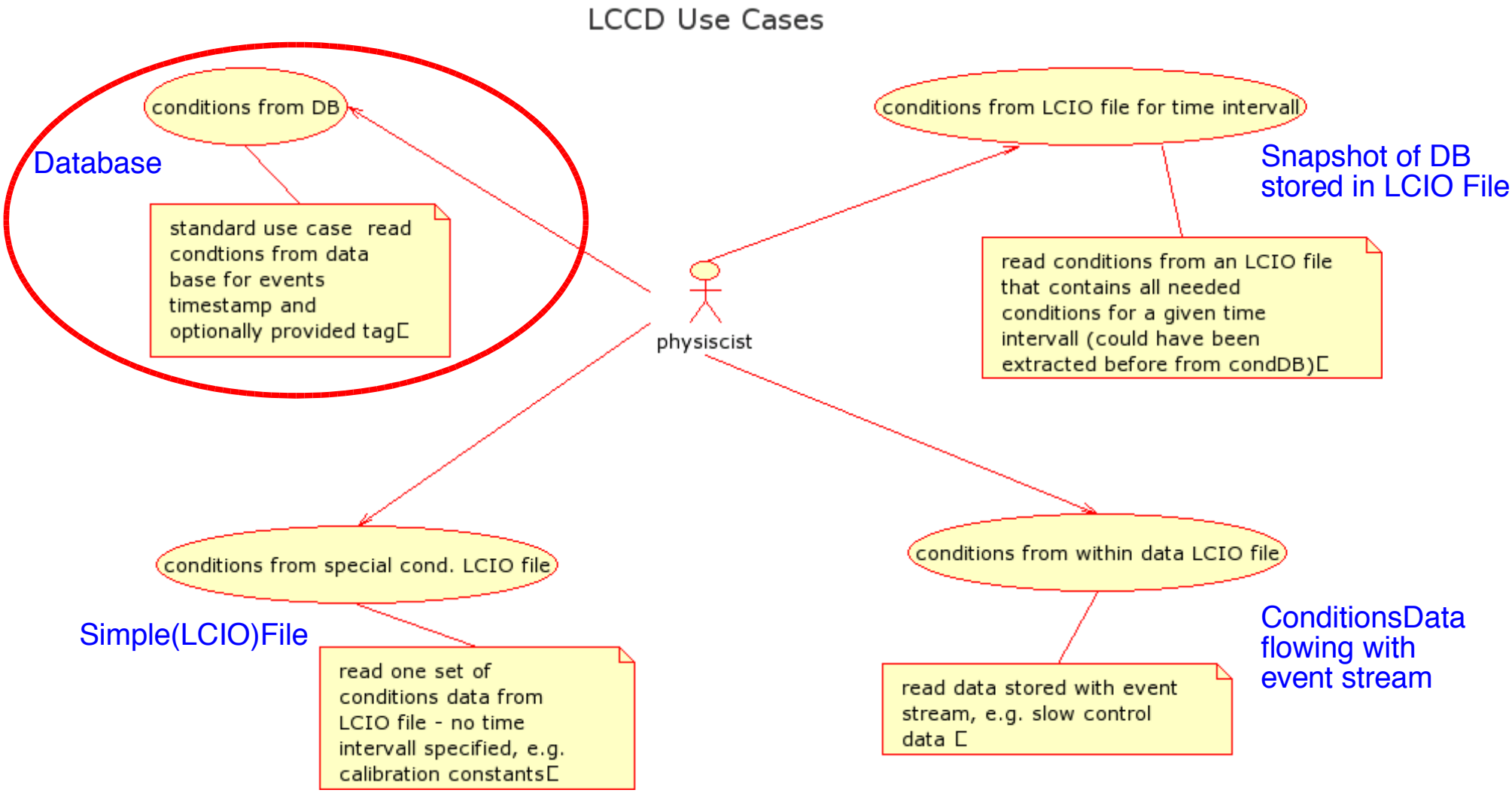
    Author Frank Gaede, DESY

- 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 tagging mechanism, e.g. for calibration constants

Sources of Conditions Data – Use Cases



Sources of Conditions Data – Use Cases



ConditionsDBMySQL – Overview

Interfaced to LCCD by Frank Gaede

- Open source implementation of CondDB API
  - Conditions data interface for ATLAS

- 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)
    e.g. LCIO objects or std::vector .....
  - tagging mechanism similar to CVS
  - scalability through partitioning options
  - outperforms implementation based on Oracle

ConditionsDBMySQL – Versioning of Conditions Data

Figure taken from CondMySQL Documentation

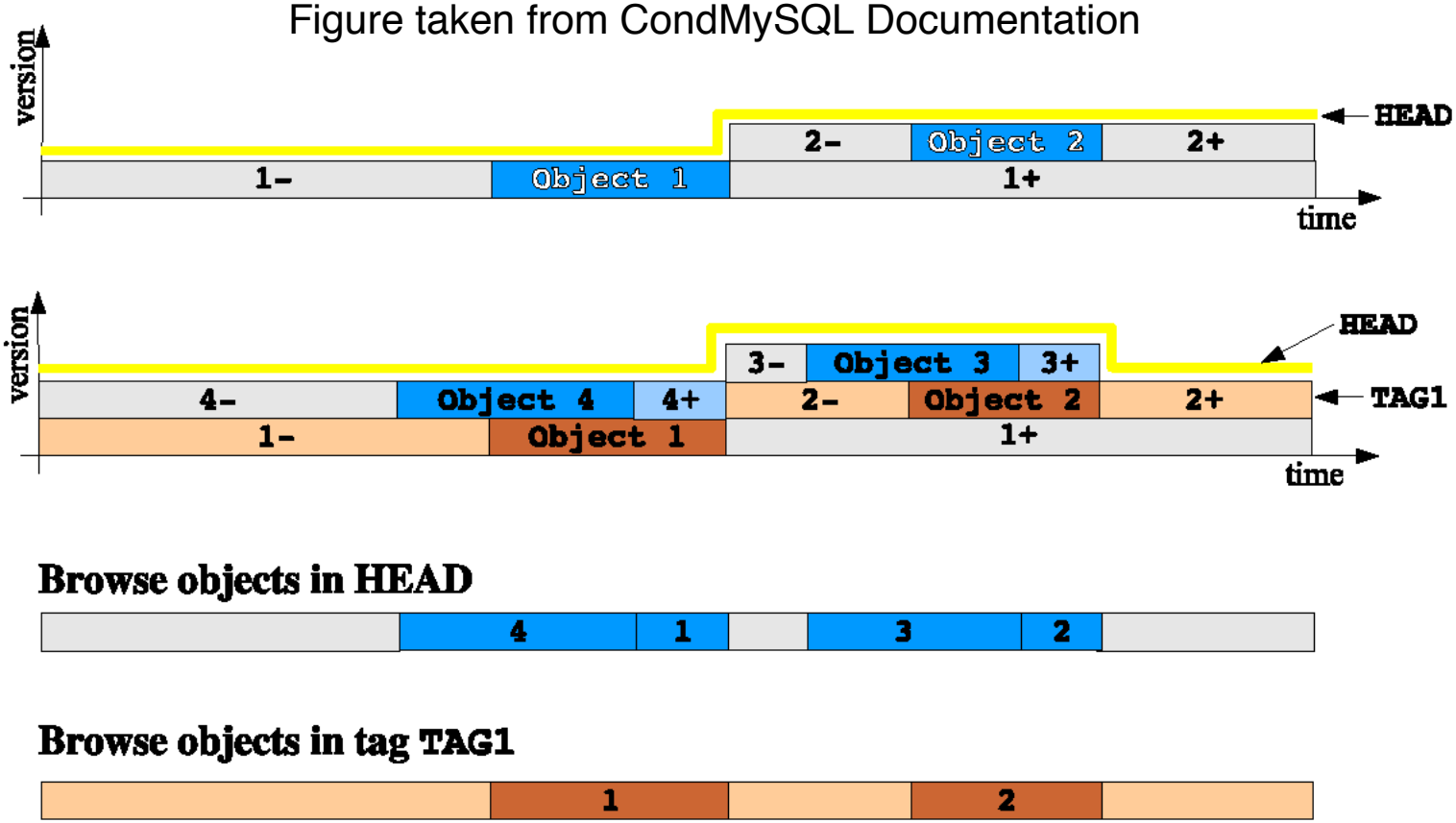


Figure 3: tagging and browsing example in the ConditionsDB mySQL's implementation.

CVS-like versioning/tagging system

'Horizontal' and vertical browsing in time possible

Time Stamp (by LCCD) in units of nanoseconds

ConditionsDBMySQL – Folder Organization

fld_id	fparent	insert_t	fpath	fdesc	fattr	ddtype	db_id	is_se
1	0	20050210202708	/	romans new folder		0	1	1
2	1	20050210202708	/roman	romans new folder		0	1	1
3	2	20050210202708	/roman/mapfolder	romans new folder		0	1	0
4	2	20050210202943	/roman/calibfolder	romans new calib folder		0	1	0
5	1	20050214183955	/lccd			0	1	1
6	5	20050214183955	/lccd/myhcal			0	1	0
7	1	20050301151849	/lccd_calice			0	1	1
8	7	20050301151849	/lccd_calice/CellMap			0	1	0

- UNIX-Like Tree Structure
- Each Folder Contains one set of ConditionsData
  CellMap relation Electronic Channel ↔ CellID
  Stored as set of LCFixedObjects
  LCCD provides Streamer Methods to read LCIO data types back from DB
- Access via Folder Name

Acessing ConditionsData Using LCCD – Users Point of View

New version of MARLIN prepared to deal with Conditions Data
(Note: LCCD does not depend on MARLIN and vice versa)

- Source of ConditionsData defined in MARLIN steering File
  e.g. ConditionsData for Cell Mapping from DB

```
DBCondHandler channelmap /lccd_calice/CellMap V00-01
```

- Handling of Conditions Data (updating etc.) within a ConditionsProcessor (provided by MARLIN)

- Code is completely transparent to Conditions Data Source
  a) Register Pointer to a CellMap and its name

```
lccd::LCConditionsMgr::instance()->registerChangeListener( _cellMapPtr , “channelmap” );
```

  b) Obtain CellMap within event

```
int cellID = _cellMapPtr->find( elec_channel ).getCellID() ;
```

  c) That's all ...

Conclusion and Outlook

- CALICE Testbeam software follows closely and puts demands on software development for ILC related studies
- Conversion of Raw Data to LCIO Format
- LCCD is powerful tool to handle Conditions Data
  Storage of Conditions Data in MySQL Database !?
  LCCD allows also for other sources
- First (small) software chain to process testbeam data using LCIO/MARLIN/LCCD exists
- Reconstruction:
  Clustering algorithms can be run within MARLIN
  (e.g. Talk by Chris Ainsley)

Next Step: Assemble pieces into working machinery