

Software Review

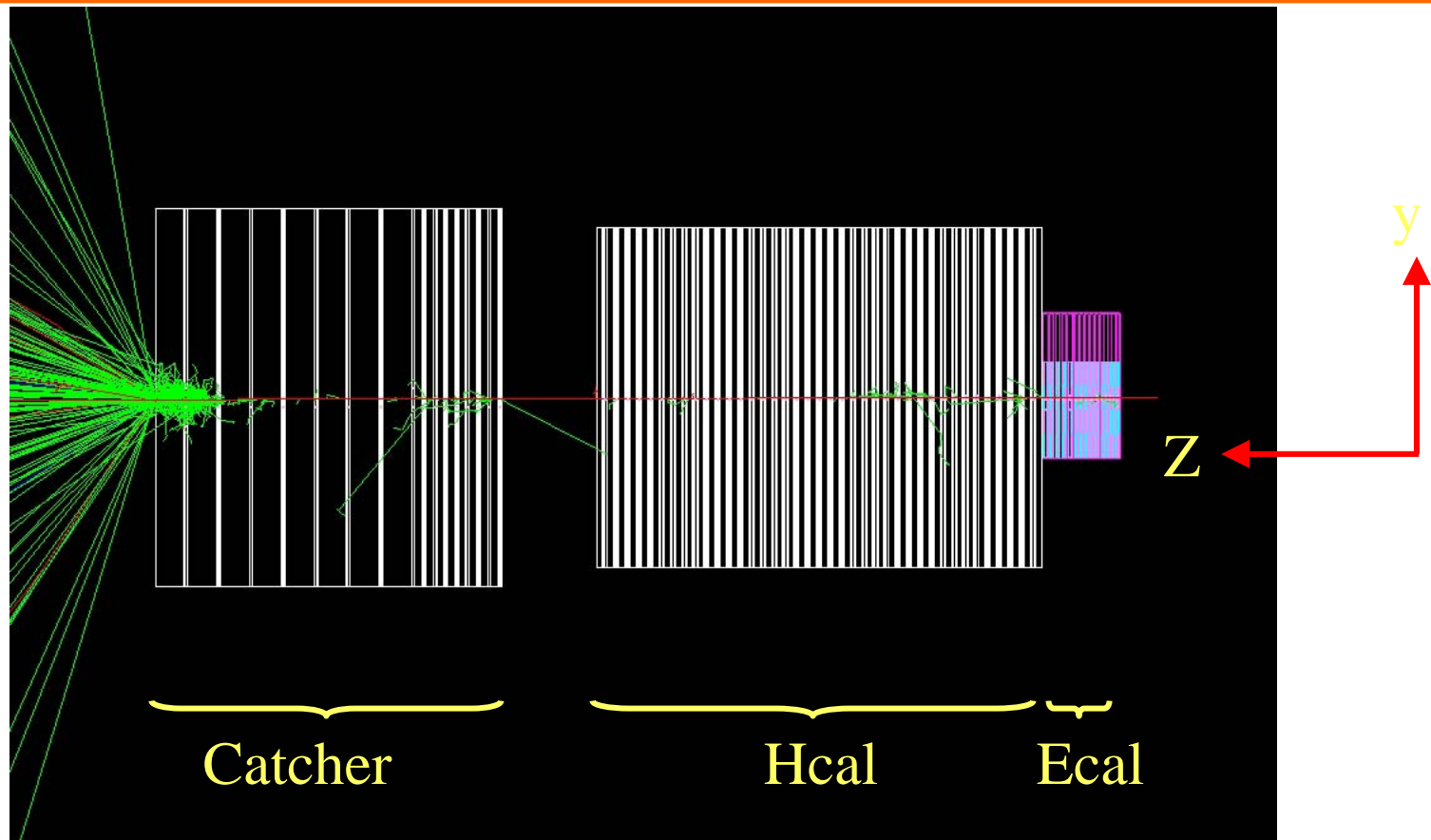
David Ward

- Monte Carlo
- Requirements for data analysis
- Data processing scheme
- Calibration
- Data storage
- Software repository

Monte Carlo

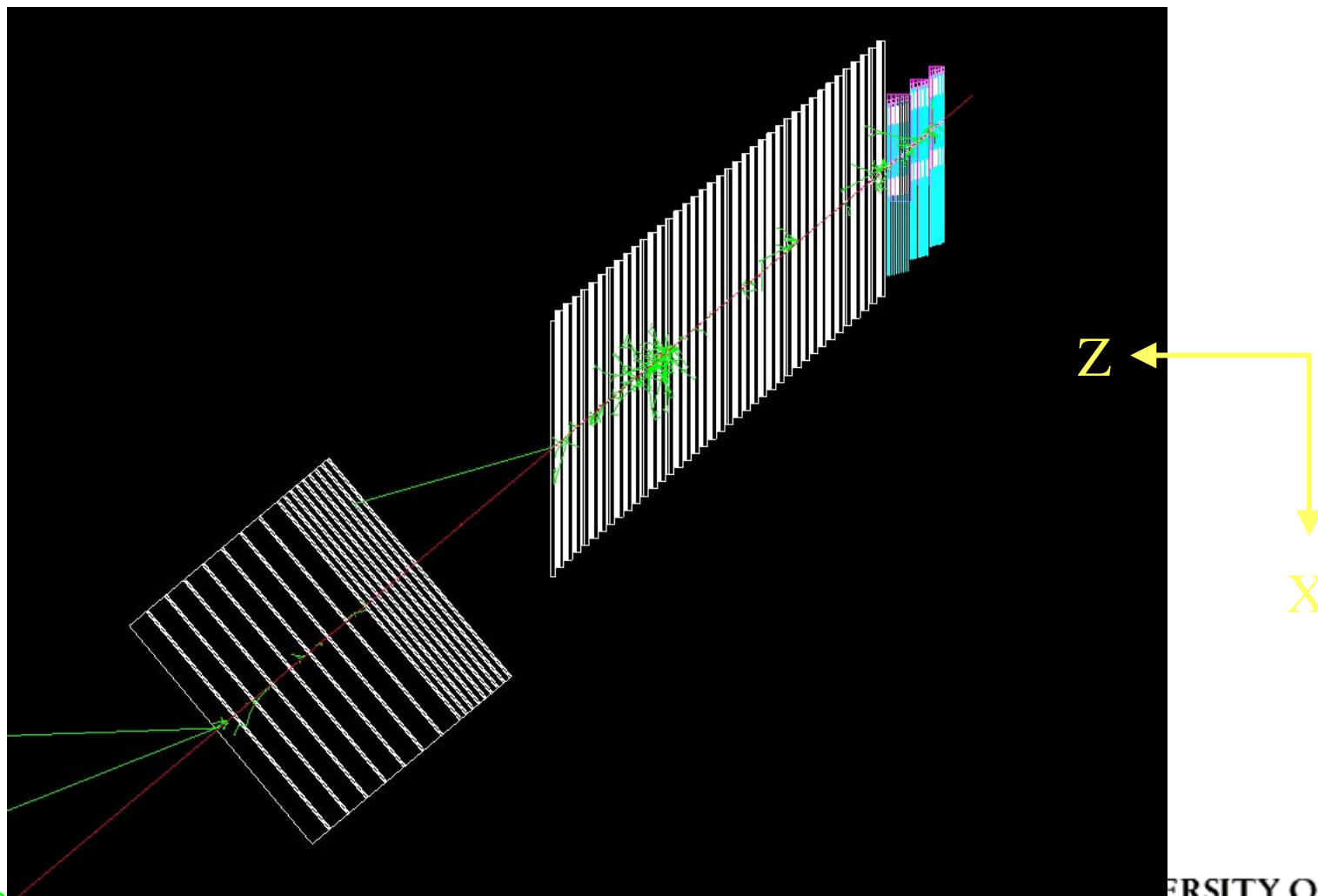
- **Mokka** (Geant4) contains detector geometries for Test Beam. Maintained by LLR, DESY and NIU. Will need maintenance/updating of geometry.
- **Geant4 provides access to many hadronic models.**
- Coordinate system, cell numbering scheme agreed June 2004. See <http://polywww.in2p3.fr/geant4/tesla/www/mokka/ProtoDoc/CoordinatesAndNumbering.html>
- Also need Geant3 MC – A.Raspereza will maintain this. Uses hard coded geometry.
- **Also option in Mokka to write out Geant3 geometry code – not fully up to date.**
- FLUKA – N.Watson has studied Flugg package; in medium term hope FLUKA will be available in Geant4.
- **This is, I think, basically under control.**

Test beam TB03, zy view



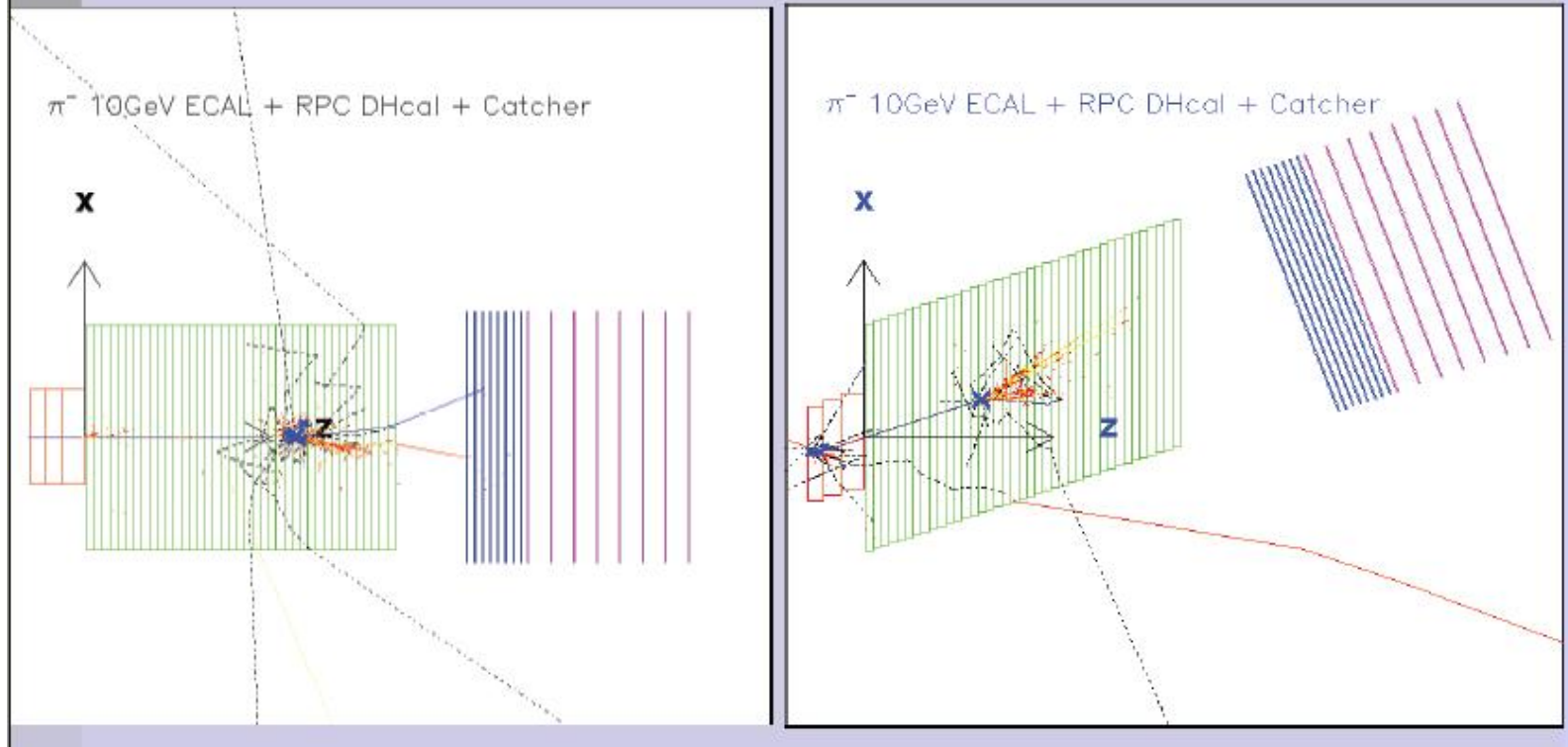
Roman Poeschl (DESY), Jeremy McCormick (NIU), Gabriel Musat (LLR)

Test beam TB03, zx max angle (40°)



Geant3 version (Calopppt)

Configuration Angle Option

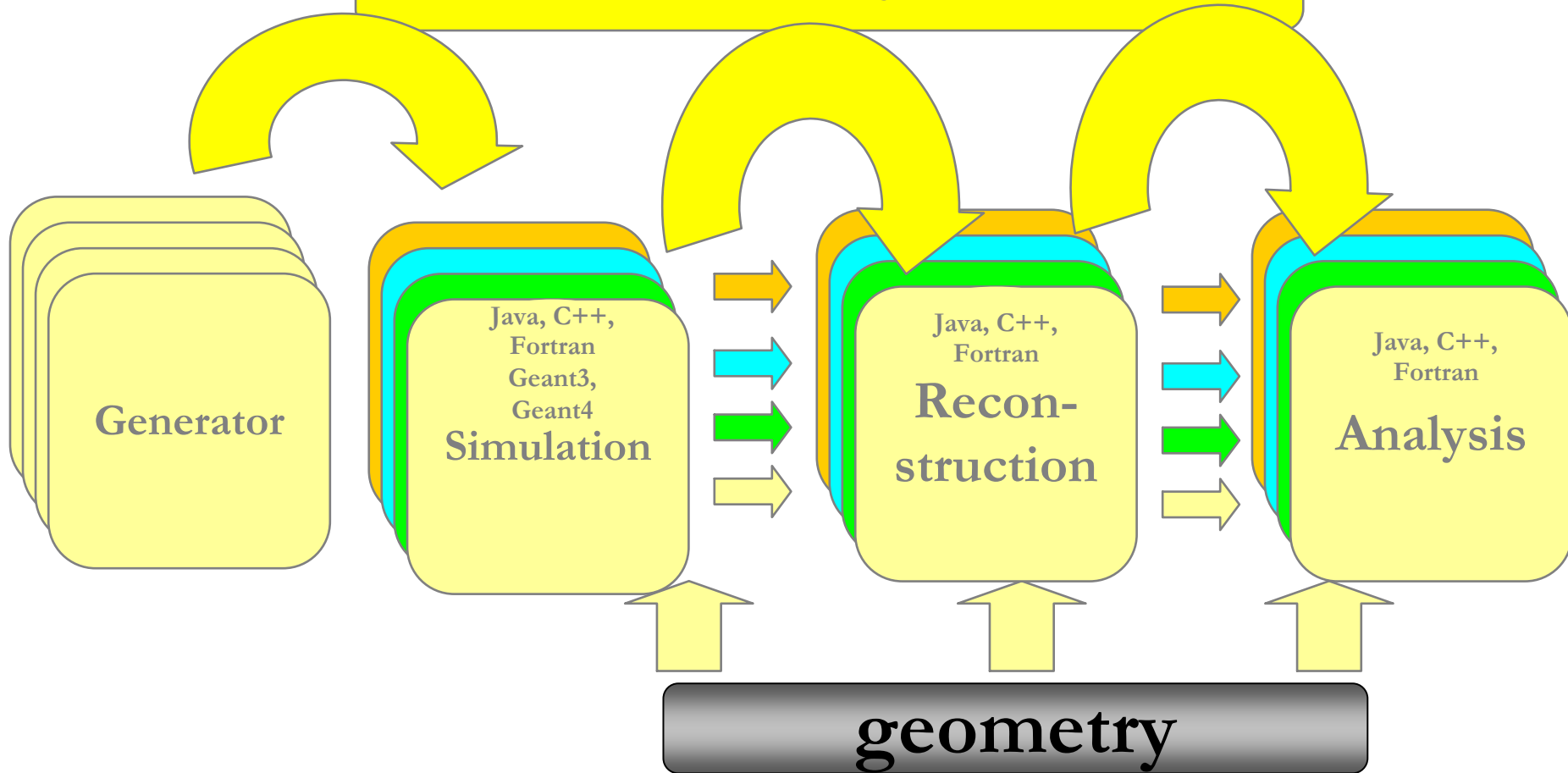


Monte Carlo

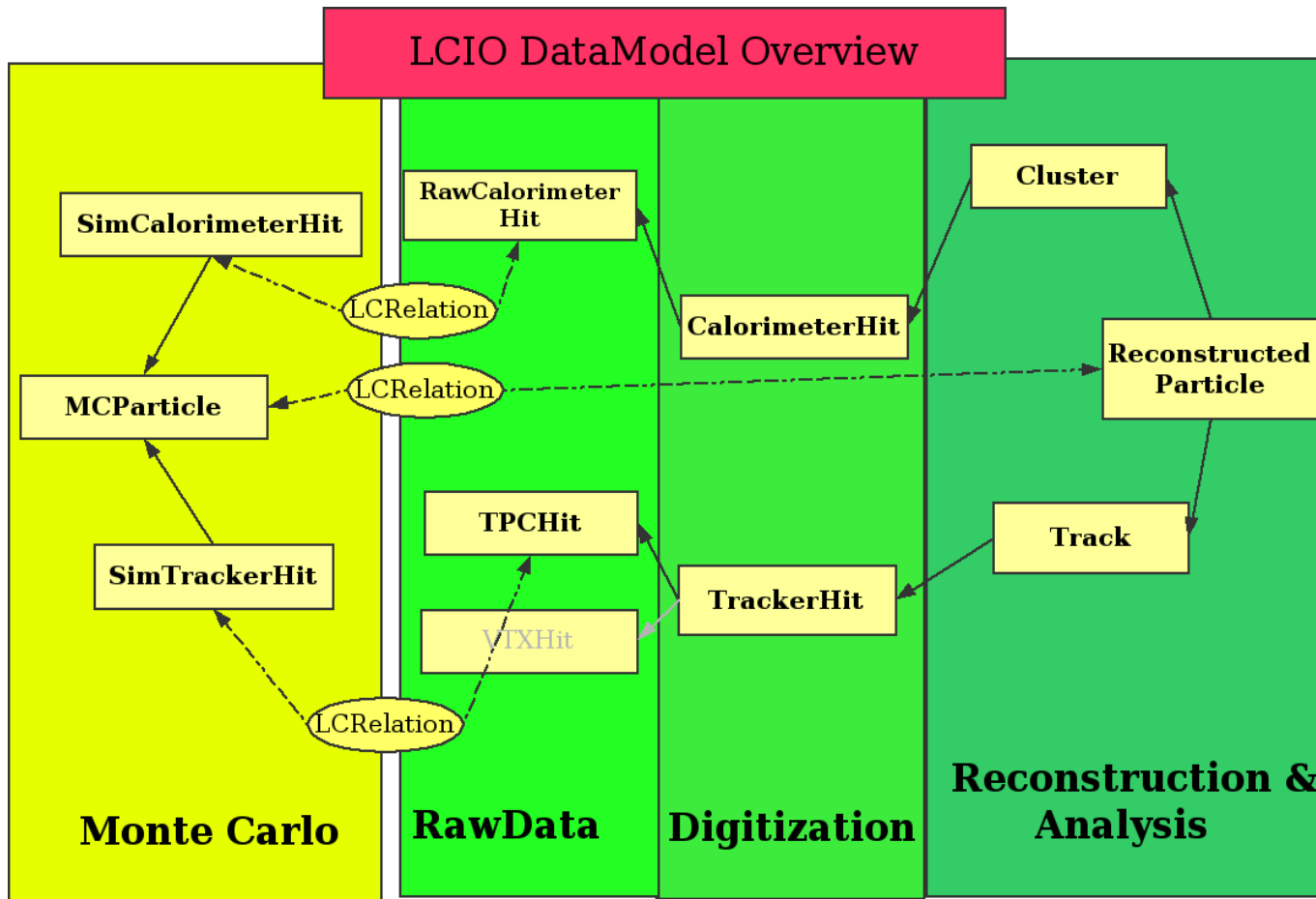
- Need to add upstream material/detectors? Description of beam profile.
- Mokka output is in LCIO format, in the form of **SimCalorimeterHits** (cell ID; hit energy [dE/dx deposited in cell]; MC truth information.)
- What is still needed is simulation of digitization effects (gain; noise; resolution; crosstalk etc.).
- A possible framework for this exists (G.Lima) operating in the LCIO/Marlin framework (DigiSim; G.Lima).
Needs filling with realistic parameterisations by detector experts.
- End result of this should be LCIO **CalorimeterHits** (cell ID; hit energy [in MIPs?]), in a form directly comparable with data, with linkage back to truth info.

LCIO

LCIO Persistency Framework



LCIO data model

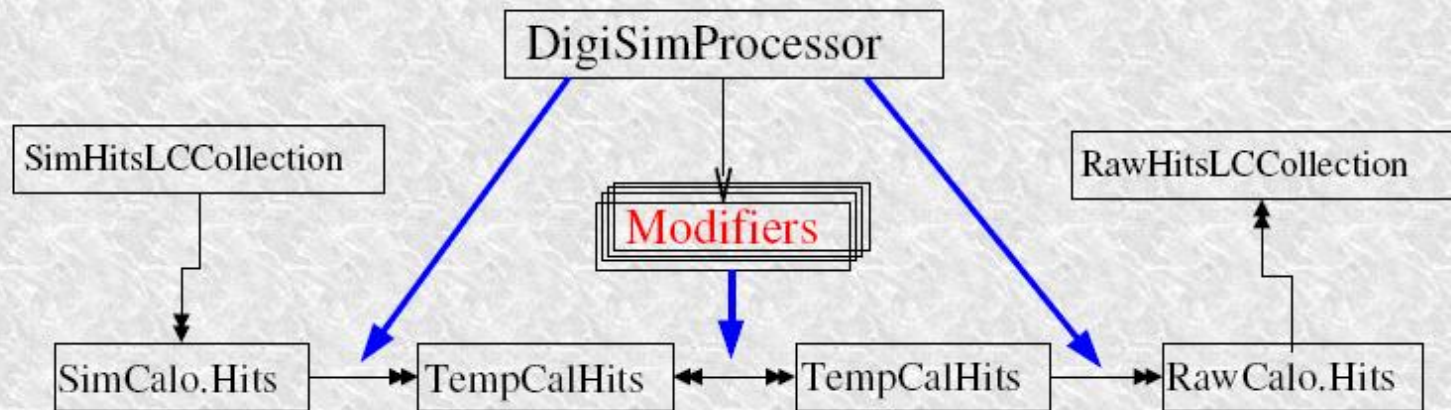


Additional LCIO objects available for storing other types of data:

- IntVec
- FloatVec
- LCGenericObject

Digisim framework (G.Lima)

DigiSim event loop



- Calorimeter hits are shown here, but the same structure could be used for trackers as well
- TempCalHits are both input and output to each modifier
- All processing is controlled by a DigiSimProcessor (one per subdetector)
- Modifiers are configured at run time, via the Marlin steering file
- New modifiers can be easily created for new functionality (more info later)

Guilherme Lima, Calice Meeting, 2004-12-08

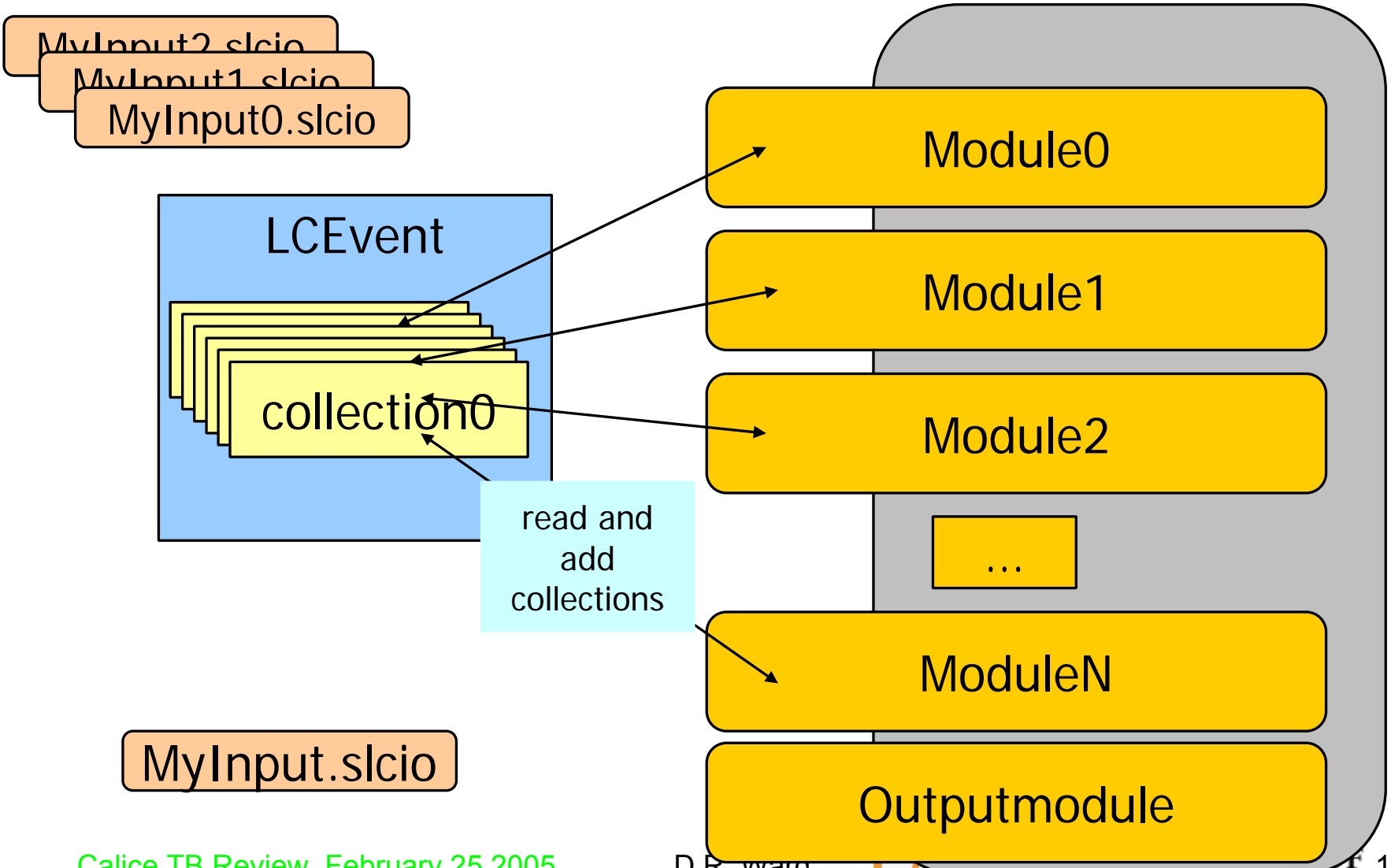


7

Marlin framework

- Designed as a simple framework for LCIO jobs (F.Gaede).
- Code written as autonomous modules, taking LCIO objects (permanent or transient) as their input and output. Should make it easy to combine code written by different authors.
- Modules can be in C++ or Fortran (or Java, though not mixed with C++/Fortran) → access to ROOT/HBOOK.
- Steering file allows control to be defined at run time, and provides a way of passing parameters to modules.
- Experience so far is that this is reasonably straightforward to use. Don't really know yet about efficiency.
- Probably needs some way of controlling event processing – in particular some method for filtering/aborting events.
- I think we should recommend the use of Marlin for CALICE Test Beam analysis.

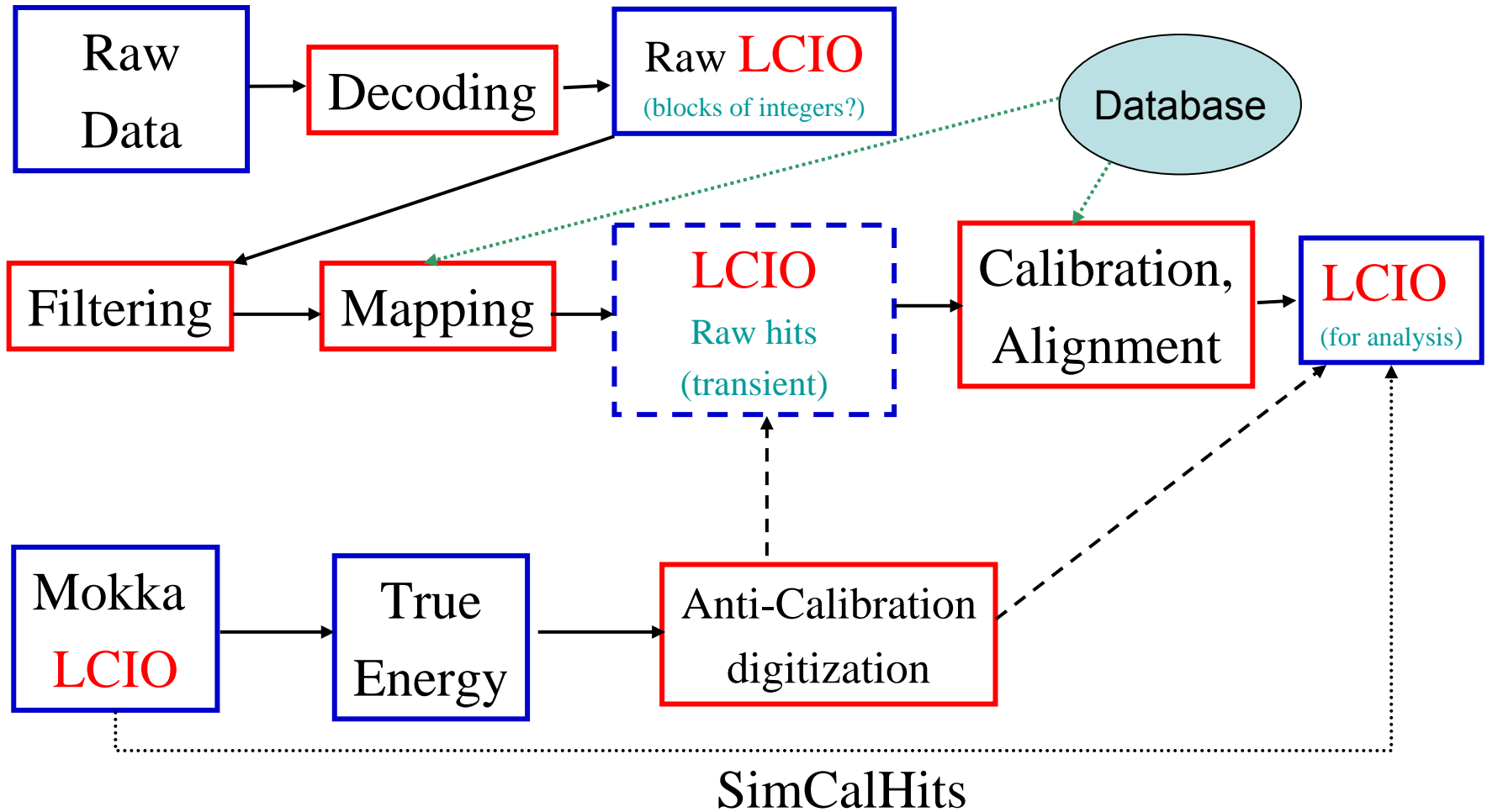
MARLIN – modules and LCIO



Basic steps in data processing

- Filtering (remove bad data or special records like pedestals)
- Cell mapping, i.e. channel # \rightarrow cell/wafer index (I,J,K)
- Alignment, i.e. cell index \rightarrow (x,y,z)
- Calibration – pedestal, gain. Zero suppression?
- Above three steps needed for each detector.
- Process beam data (drift chambers, Čerenkovs etc.)
- End up with LCIO CalorimeterHits for direct comparison with Monte Carlo.
- Analysis, clustering, histograms/ntuples, event display etc.
- Each of these could be incorporated into the MARLIN framework as separate modules.

Dataflow (agreed 16 Feb)



Data processing – points agreed 16 Feb.

- Conversion to LCIO. Agreed on "intelligent unpacking", i.e. raw data of any given type (CRC (calorimeter) hits, TDC data, trigger data, status info etc.) would be stored in separate LCIO objects. In the short term, the CRC and TDC data are the most urgent.
- Ideally the slow controls info should be stripped off at the conversion stage and put into the database. Also pedestals. May want, temporarily, to put this into an LCIO object.
- Mapping and filtering will not be applied before the LCIO stage. (This was the main point of controversy). We could envisage a migration path where some of these features could be introduced later.
- Still need detailed discussion on format of objects (LCIntVec, LCGenericObject?). Come up with a plan in a matter of days, hopefully. Paul, George and Roman seem to be the crucial people here; David would like to be kept involved in discussion.
- Need to press on expeditiously with the LCIO conversion, but can't expect to get useful results out of it before LCWS. Assume ECAL analyses for LCWS will be based on native raw data.

Data processing – points agreed 16 Feb.

- LCCD database package of Frank Gaede was welcomed. Assume as the basis of our planning that we will use this.
- Roman hopes to have realistic experience with its use before the NIU Calice meeting.
- We will use the DESY dCache for the master copy of the native raw data. DESY group will proceed to set this up.
- Implies that conversion of raw data to LCIO will be done in DESY computer centre (at least during data taking at DESY).
- Assume we will use the MARLIN framework for LCIO-based work.

Database

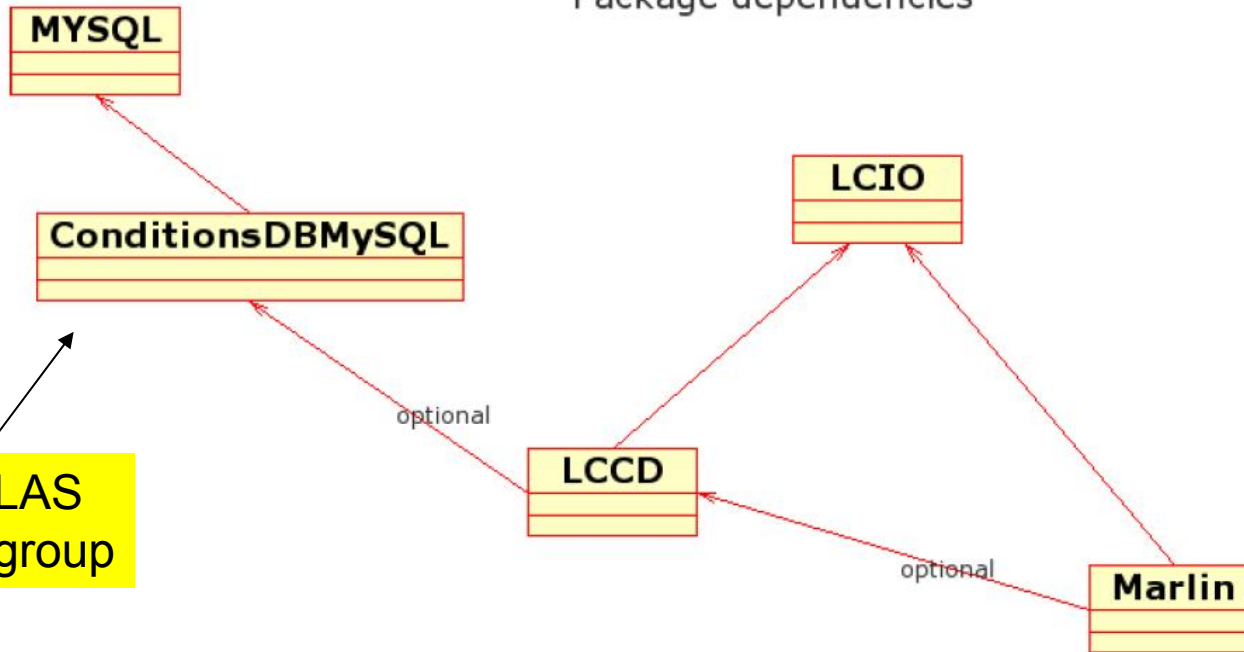
- Frank Gaede has a proposal for LCCD (Linear Collider Conditions Data) framework.
- Would access a MySQL database via a package ConditionsDBMySQL (from the Lisbon Atlas group).
- Would fill LCIO calibration objects – persistent throughout a run (or for some appropriate period of validity). **Need to be LCIntVec, LCFloatVec or LCGenericObject.**
- Calibrations are then easily accessible in code processing collections of LCIO hits etc.
- Frank and the DESY FLC group are keen to implement this; Calice wouldn't have to produce the framework (though we would be guinea pig users).
- A first version is available now. Associated mods made to LCIO and MARLIN.
- We can concentrate on populating the database with useful data, **but we need someone to manage the database.**

LCCD scheme



LCCD dependencies

Package dependencies



From ATLAS
(Lisbon) group

Latest status on LCCD



Summary & Outlook

- LCCD is a simple toolkit for retrieving and storing conditions data transparently through
 - LCIO files
 - conditions database (ConditionsDBMySQL)
- v00-01 released
- need testing, testing, testing
- user input very welcome/needed
- **NB: while LCCD provides a simple user interface to access the conditions database it doesn't deal with setting up and managing that database, e.g.**
 - **security issues (read/write access)**
 - **data integrity**
 - **availability/ quality of service****need a dedicated person for this !**

Data Storage

- The DESY group have a proposal to store the data in the dCache mass storage at DESY.
- Could (and probably would) still maintain copies in UK/France.
- In this case, processing of native raw data to LCIO would probably be done at DESY.
- (Similar data store system available at Fermilab.)
- Organised in three (at least) parallel directories ../native/ ../raw/ and ../reco/ (where raw and reco would be in LCIO format).
- Access via anonymous ftp, dCache client tool (dccc) or Grid-ftp (preferred).
- Important that all members of Calice have read access by one of these mechanisms; write access limited to a handful of people.
- DESY group setting this up following TB's agreement last week.
- How should we document the data recorded (beam energy, position, data quality etc.)? Web pages ...

Code Repository

- Roman Pöschl has been setting up a CVS repository for Calice code at DESY Zeuthen. Now exits; web interface to come soon.
- **Already used for Brahms, LCIO, Marlin, LCCD.**
- Encourage CALICE users to check in code which others will find useful.
- Do we also need a web page to collect information on software tools etc.?
- **Of course we still need to do better with documentation.**