Software/calibration

David Ward

- Monte Carlo
- Requirements for data
- “DESY scheme”
- Paul’s scheme
- Calibration / data storage

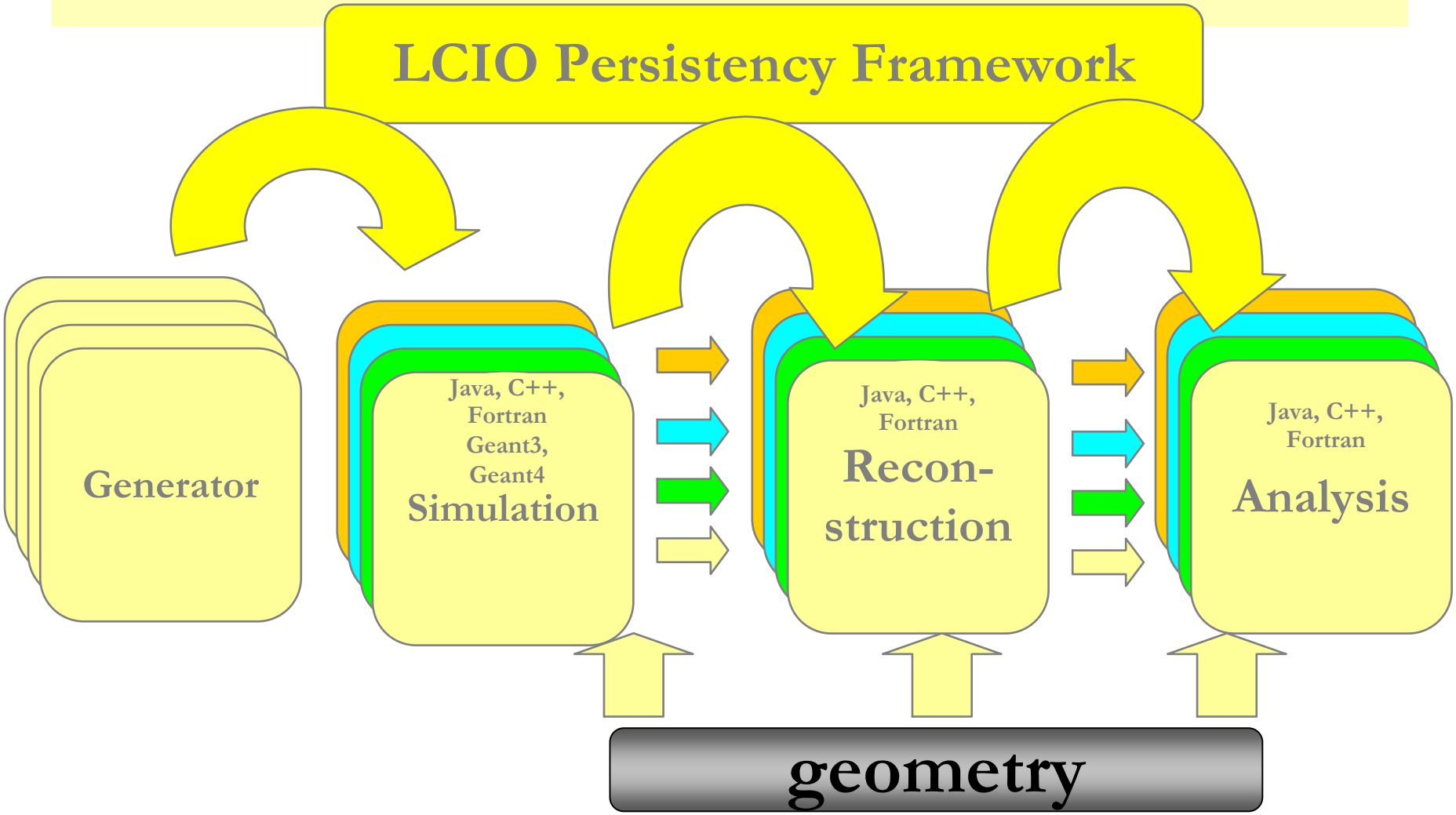
Introduction

- Biggest (or most controversial) issue is how to convert raw data from DAQ to a format suitable for analysis. Presumably LCIO (compatibility with MC; plus political reasons) rather than ROOT (say).
- Obviously ties in with many other open questions; allude to briefly.
- Urgency of a decision. Problem became apparent at December CALICE meeting. We have been taking ECAL data for three weeks now, and analysis is paralysed because nobody wants to invest effort in the wrong framework.

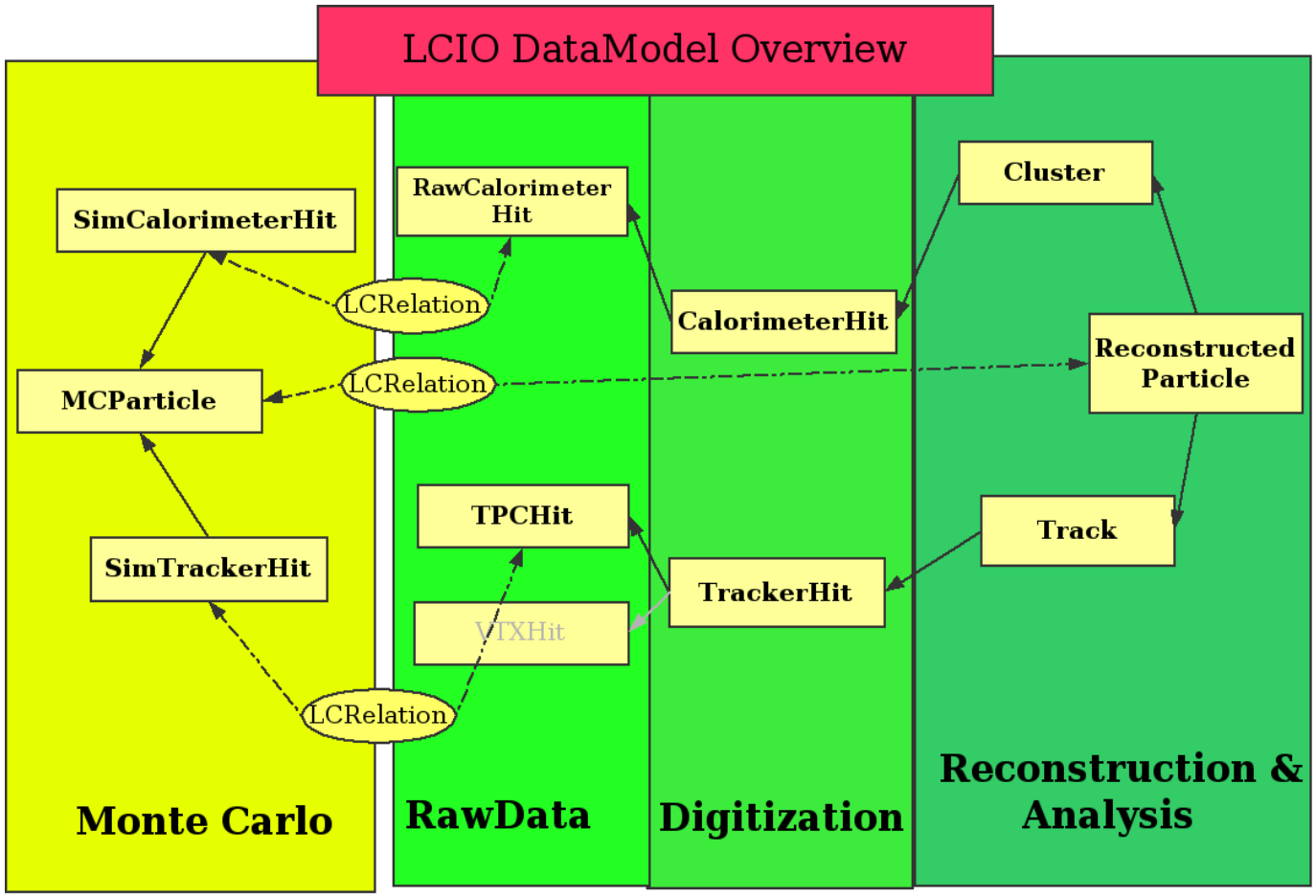
Monte Carlo

- Mokka (Geant4) contains detector geometries for Test Beam. Maintained by LLR, DESY and NIU. (Also Geant3 MC – A.Raspereza).
- Coordinate system agreed June 2004.
- Need to add upstream material/detectors?
- 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 CalorimeterHits (cell ID; hit energy [in MIPs?]), in a form directly comparable with data, with linkage back to truth info.
- Most of this is, I think, under control.

LCIO


LCIO Persistency Framework
Generator
Java, C++,
Fortran
Geant3,
Geant4
Simulation
Java, C++,
Fortran
Recon-
struction
Java, C++,
Fortran
Analysis
geometry

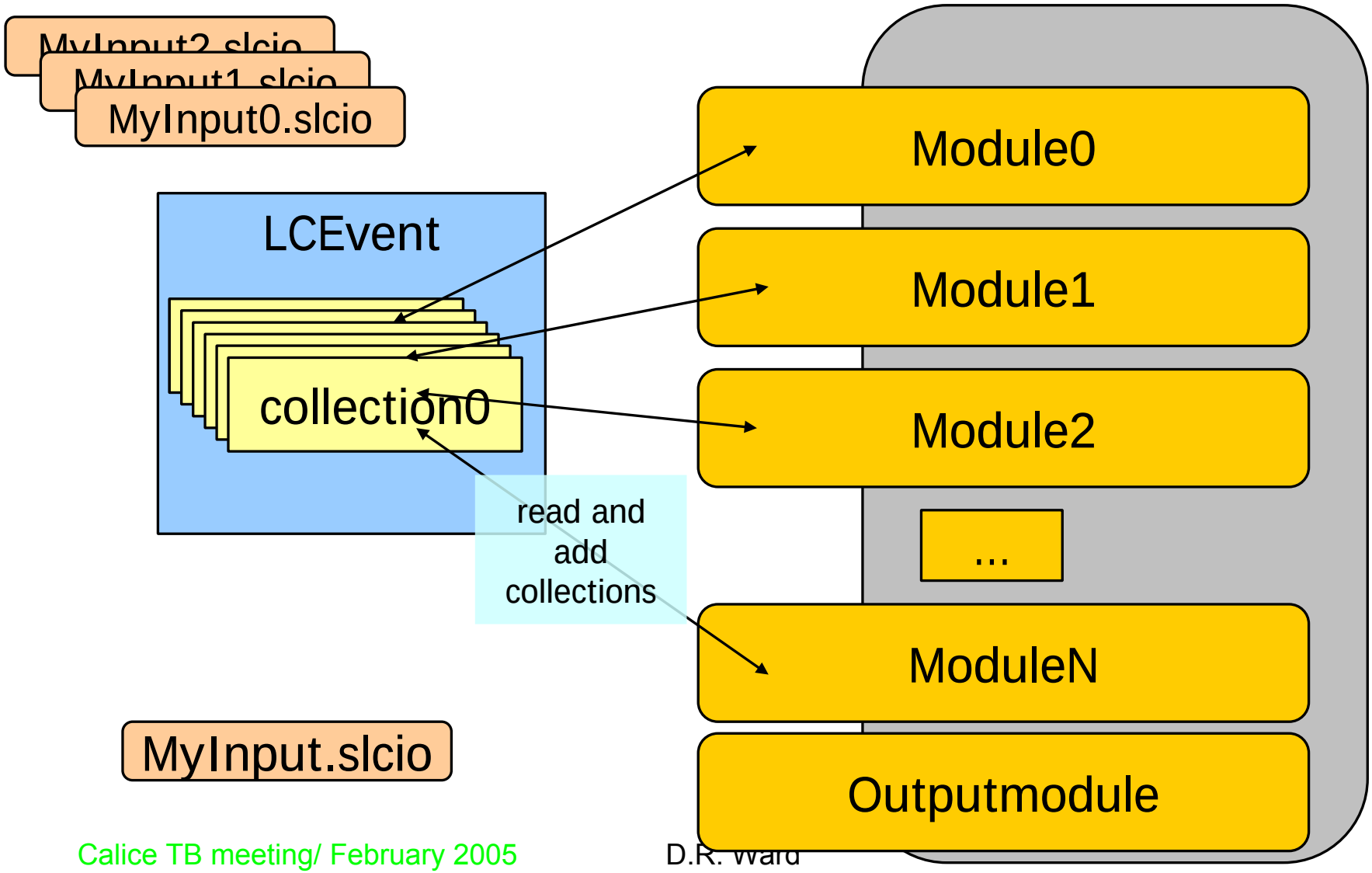
LCIO data model



Additional LCIO objects available for storing other types of data:

- IntVec
- FloatVec
- LCGenericObject

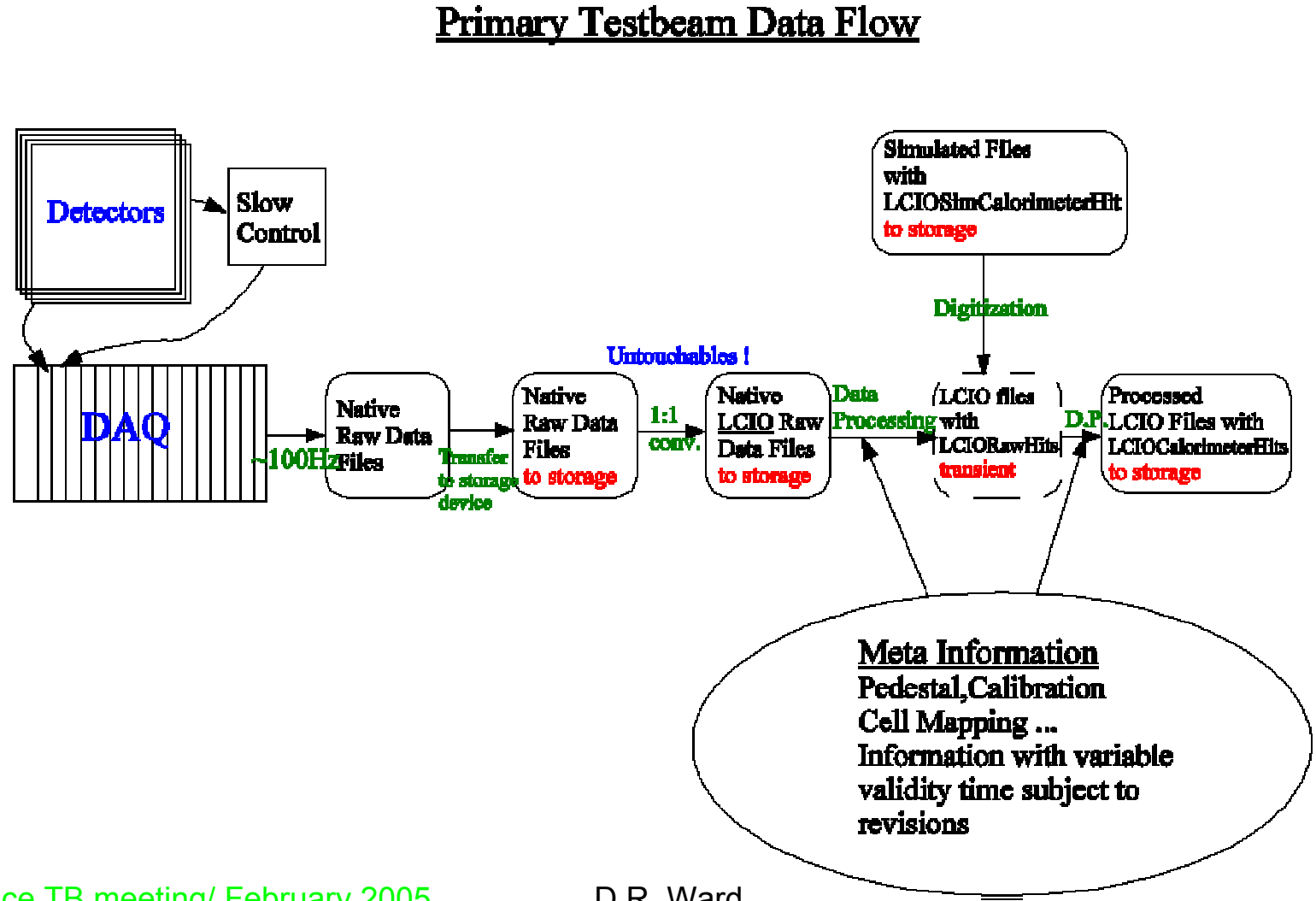
MARLIN – modules and LCIO



Basic steps in data processing

- Cell mapping, i.e. channel # → cell/wafer index (I,J,K)
- Alignment, i.e. cell index → (x,y,z)
- Calibration – pedestal, gain. Zero suppression?
- Above steps needed for each detector.
- Process beam data (drift chambers etc.)
- End up with LCIO CalorimeterHits for direct comparison with Monte Carlo.
- Analysis, clustering, histograms/ntuples, event display etc.

DESY scheme (R.Pőschl)



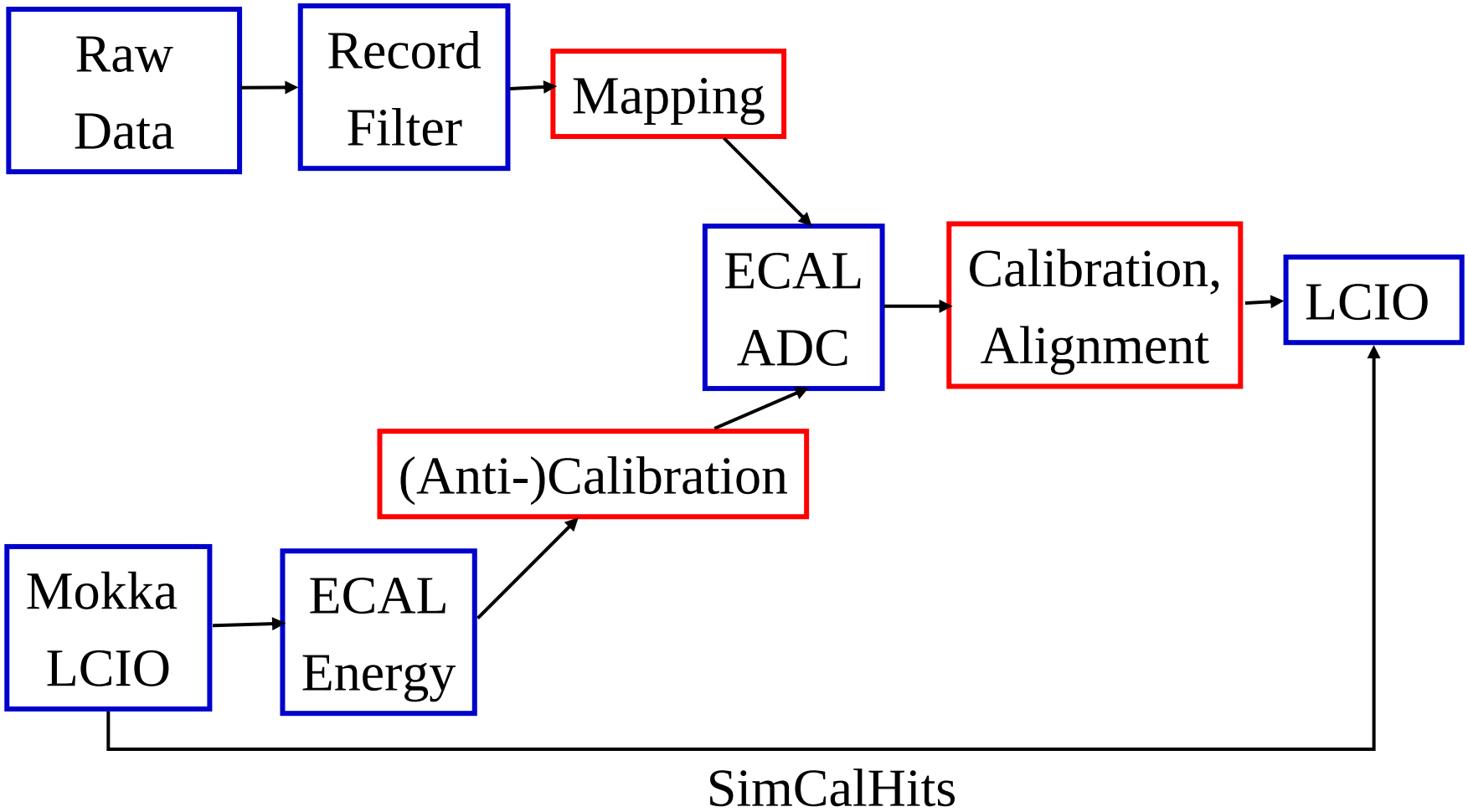
Pro

- One-to-one translation of native raw data – stored as vector of integers – simple; minimise likelihood of error.
- All processing within LCIO/MARLIN framework.
- Code can be written by many users in a modular way; modules communicate cleanly through LCIO objects.
- LCIO promises to provide a convenient database system. (see later)

Con

- Decoding of raw data block is complex. Data organisation linked to needs of hardware, rather than users.
- Needs to use classes from DAQ software in LCIO framework (maintenance issues), or else to recode them (compatibility problems).
- Duplicates stored data volume.
- No natural framework for non-calorimeter hit data (beam counters; trigger info; slow controls data …)

P.Dauncey (as of Dec'04)



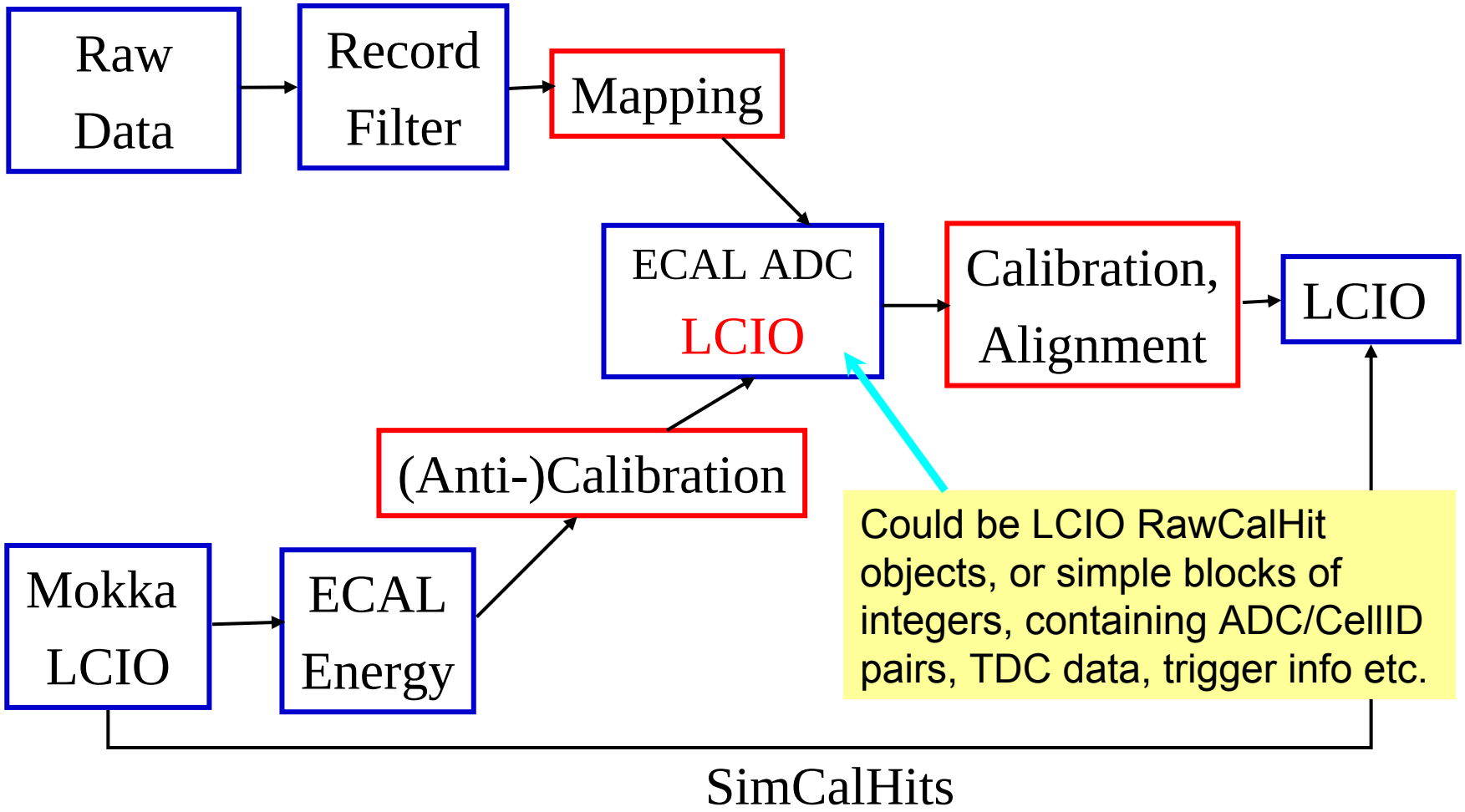
Pro

- All calibration done before conversion to LCIO. Most users don't need or want to know the details.
- End users all use same calibrated data.
- Few users need database access.
- No need to duplicate or copy stored data.
- No need to duplicate or copy code already available in DAQ suite.
- All "analysis" (data/MC comparisons) still performed within LCIO/MARLIN framework. (though it depends what you regard as analysis)

Con

- Calibration is part of analysis.
- Anyone working on calibration/alignment needs to use DAQ framework (lack of documentation), and follow changes in DAQ code (maintenance issue).
- Code only works in LINUX and C++ framework. May not be a big problem.
- Fewer people will be able to effectively participate in calibration work. (Pro or con?)
- If some of the same code to be run on MC, then MC converted from LCIO to some other format or framework and then back to LCIO.

Possible variant



Pro

- Separates mapping (intrinsically linked to DAQ) from calibration (arguably part of analysis).
- More user friendly, I think.
- MC digitization code all in LCIO; share relevant code for data with Monte Carlo.

Con

- Not a one-to-one copy of native data. Mistakes will occur; reprocessing will be needed. Or messy patch-up procedure in LCIO.
- Lose access to hardware configuration during subsequent analysis.
- Duplicates stored data volume.
- No natural framework for non-calorimeter hit data. Need to define this.

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).
- 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). Claim it can be done fast. We could concentrate on populating the database with useful data.

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 (dccp) or Grid-ftp (preferred).
- Important than all members of Calice have read access by one of these mechanisms; write access limited to a handful of people.
- DESY group would like TB's agreement to set this up.