LCIO Turns 20

Norman Graf, Tony Johnson, Jeremy McCormick (SLAC)

Frank Gaede (DESY)

CHEP 2023: Sustainable and Collaborative Software Engineering

May 11, 2023

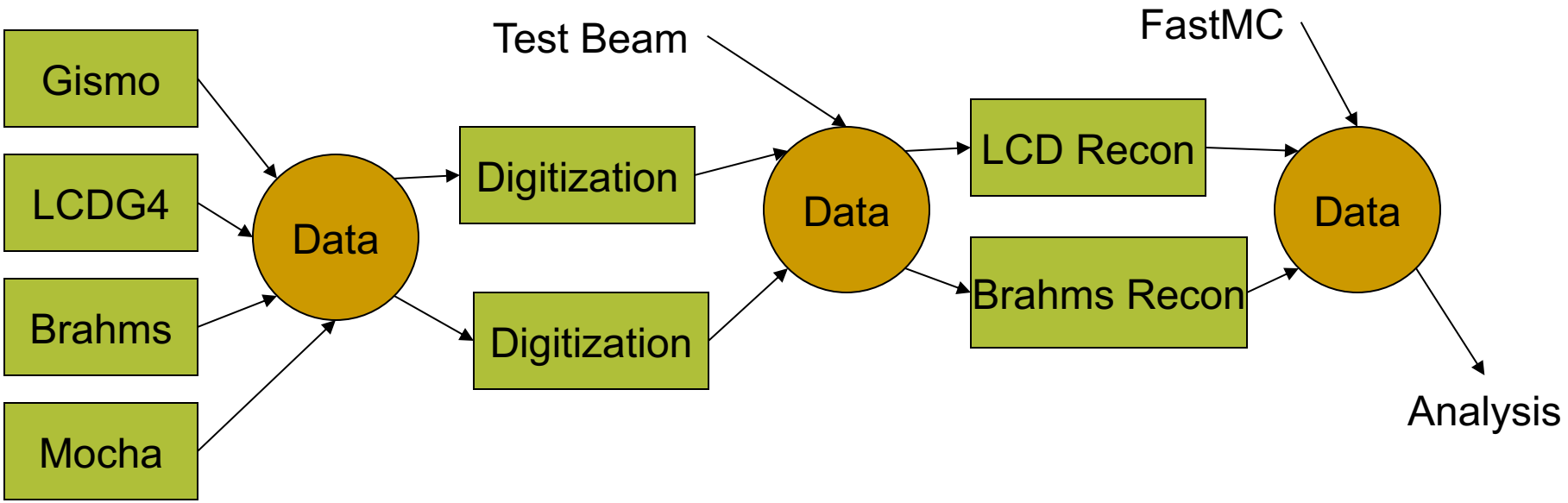
Linear Collider Collaborative Software

- The dawning of the 3rd millennium brought with it a number of ambitious proposals for new e^+e^- linear colliders.
- Preliminary physics and detector studies had relied on reuse of existing software (LEP/SLC).
- HEP transition to object-oriented languages provided opportunity to collaboratively develop common software

Goals

- Long-term goal was to encourage interoperability of worldwide linear collider simulation/reconstruction/analysis programs

First Step – Common IO format



LCIO Philosophy

"Simplify, simplify, simplify"

Thoreau

"Make everything as ***simple as possible***, but not simpler."

Einstein

Identify the key elements for an event data model appropriate to a colliding detector experiment.

Common Detector Response Simulation

- Would need to record input MC hierarchy and all “important” particles created in simulation
- Realized that HEP collider detectors divide naturally into two distinct types:
  - Trackers
    - Non-destructive, position/time sensitive detectors
    - Store GEANT step information at each sensitive detector
    - Important to maintain MC provenance for every hit
  - Calorimeters
    - Destructive, energy sensitive, naturally quantized by size of readout cells
    - Only store energy sum and time for shower energy deposition
    - Need only record MC information for showering particle
- Common detector simulation output would consist of MC particle information, hits in tracker sensitive volumes and energy in calorimeter cells.
- Goal was to develop a single, common Geant4 executable

LCIO v0

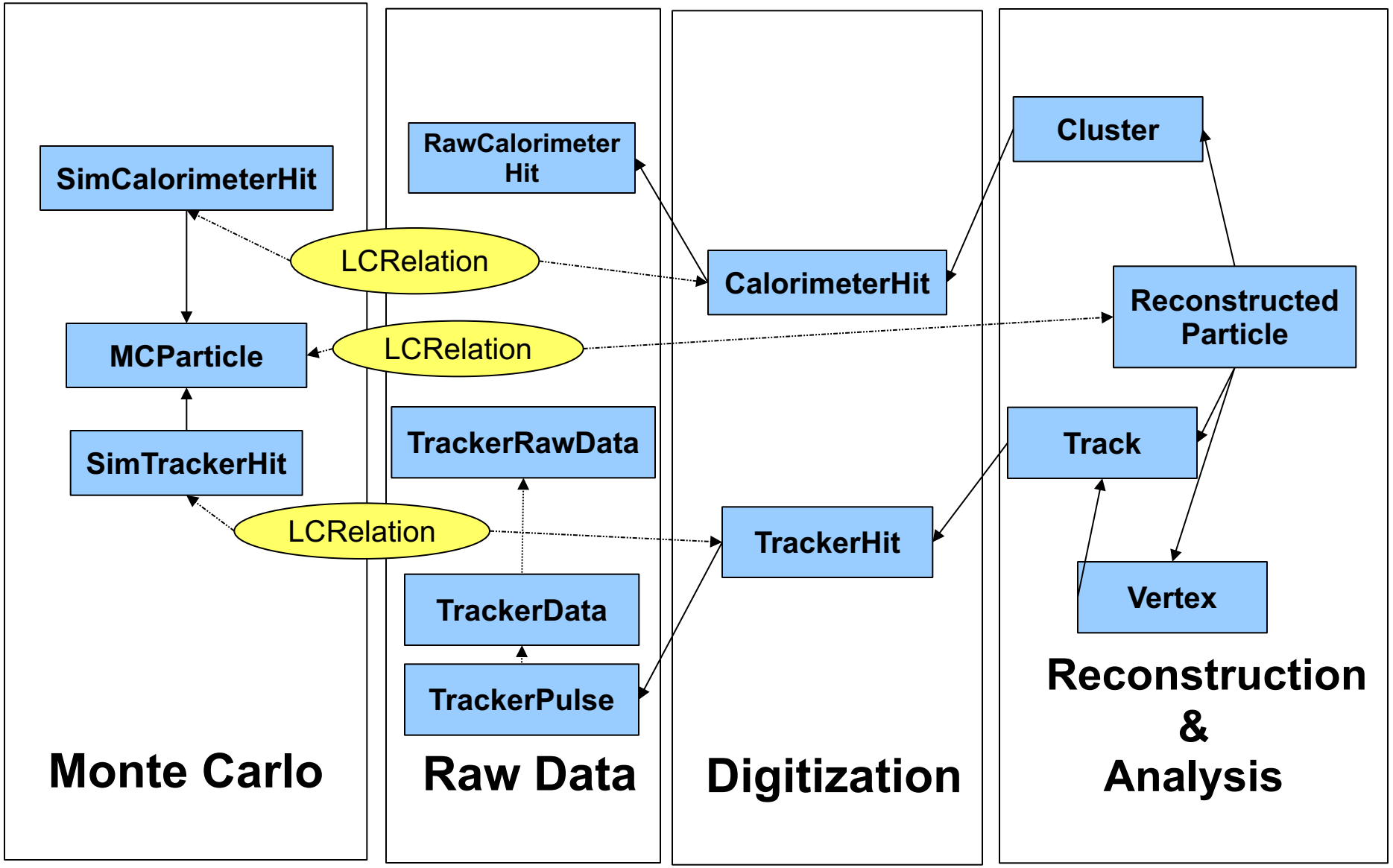
- Original Development Team
  - Norman Graf, Tony Johnson (SLAC)
  - Ties Behnke, Frank Gaede (DESY)
  - Paulo Mora de Freitas (LLR)

- Original Event Record had 4 types of block
  - EventHeader
    - Run #,event #,detector Name, timestamp
  - MCParticle – All pre-shower particles
    - 4-vector, origin, status, parent
  - TrackerHit
    - Position, MCParticle, dedx, time
  - CalorimeterHit – includes pointer to MCParticles(s)
    - Exists in two variants
      - Short
        - stores one entry per cell and per shower which produced this entry, includes "cellID"
      - Long
        - each cell records each individual particle type which crosses the cell. Both the energy and the MCParticle producing this hit are stored. Includes cell position and "CellID"

Reconstruction / Analysis EDM

- Common Simulation EDM soon was expanded to include the reconstruction/analysis EDM
- Again, key was to recognize that particle physics reconstruction objects can be simplified into essentially just tracks and calorimeter clusters
- "Physics Objects" were represented by ReconstructedParticle objects, which were allowed to be composite:
  - Charged particle: track + calorimeter cluster
    - e+/e- : track + EM calorimeter cluster
    - Pion : track + EM + Hadronic calorimeter cluster
    - Muon: track + EM + Hadronic MIP clusters + muon track
  - Photon
    - Single EM cluster or e+/e- conversion pair
  - Jet: assembly of ReconstructedParticles
  - etc.

LCIO Event Data Model



LCIO Extensions

- In addition to the predefined classes, LCIO also allows users to define extensions to classes by:
  - defining collections of primitives
  - defining associations via LCRelations
  - defining new objects via LCGenericObject

LCIO Persistency

- SIO: Simple Input Output → slcio files
  - Tony Waite (SLAC)
  - Architecture independent binary format.
  - A high integrity, self-checking data layout.
  - Multiple simultaneously open input and output streams.
  - Heterogeneous record types on each stream.
  - Pointer relocation at the level of a record.
  - On the fly data compression/decompression.
  - C++, Java and FORTRAN implementations at the start.
  - Recently re-implemented to support multi-threading.
- ROOT → rlcio files
  - Proof-of-principle implemented
  - No real benefit in size or read/write performance
  - Did not gain traction in the community
- Keep it simple!

I have an LCIO File. Now what?

- lcio command-line tool
- Java Analysis Studio (JAS3)
  - LCIO event browser
  - Wired event display
- org.lcsim and ilcsoft
  - Full access to the event data and geometry
  - Drivers give full access to reconstruction and analysis
  - Output LCIO file or AIDA histograms/tuples
- ROOT access via LCIO dictionary
- ROOT access via pyROOT
- python access via pyLCIO
- Jupyter access via Julia

LCIO & ROOT

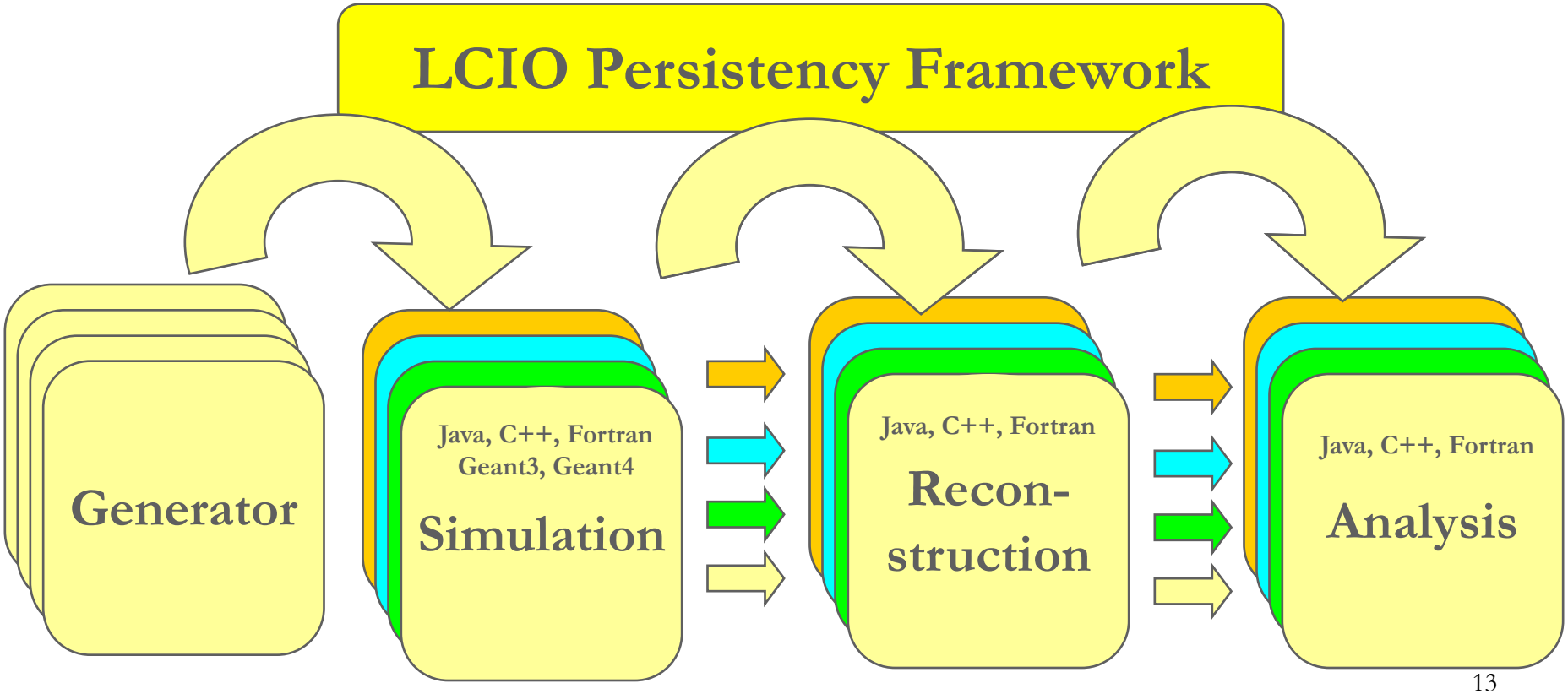
- LCIO allows to optionally create a ROOT dictionary for all LCIO classes. With this one can:
  - Use LCIO classes in ROOT macros
  - Write simple ROOT trees, e.g. std::vector<MCParticleImpl*>
  - Use TTreeDraw for quick interactive analysis of LCObjects, e.g.:

```
//---gamma conversions:
Tcut isPhoton("MCParticlesSkimmed.getPDG()==22" ) ;
LCIO->Draw("MCParticlesSkimmed._endpoint[][0]:
                    MCParticlesSkimmed._endpoint[][1]",isPhoton ) ;
```

  - Write complete LCIO events in one ROOT branch

One EDM to bind them all…

- Defining a common EDM for all linear collider studies allowed heterogeneous software systems, frameworks and languages to interoperate
- Enabled sharing of data as well as software
- Allowed apples-to-apples comparisons of detector performance



LCIO in practice

- The current LCIO EDM has been battle-tested and proven in many large Monte Carlo physics and detector design exercises for future e^+e^- colliders
  - ILC
    - ILD and SiD concepts
  - CLICdp
  - CEPC
  - FCC-ee
- As well as test-beam campaigns
  - Calice, LC-TPC, EU-Telescope,…
- And running experiments
  - HPS
- And is being used to design future experiments
  - REDTOP @ FNAL
- Current LCIO persistency using SIO has I/O performance comparable to ROOT I/O for objects although performance was not one of the main design goals for LCIO

The Future

- EDM4hep is a common event data model for HEP being adopted by a number of particle physics proposals
  - developed within the context of the key4hep software eco system
  - based on the EDM toolkit PODIO
    - See talk by Thomas Madlener
  - essentially a one-to-one correspondence with LCIO
  - Linear Collider community is moving to key4hep and EDM4hep in an adiabatic way
    - LCIO <-> EDM4hep conversion available
- EDM4hep adopted by ILC, CLIC, FCC, CEPC
  - under investigation by EIC
  - LUXE @ DESY
- “A rose by any other name…”

Lessons Learned

- Defining a common EDM for all ILC physics & detector studies provided the basis for developing a common software eco-system
- Adoption by detector concepts at other proposed accelerators (CLIC, CEPC, FCC-ee) enabled them to make rapid progress in their physics and detector simulations
  - Conversely, improvements made e.g. by the CLIC team were then immediately available to the ILC DBD exercise.
- A common EDM, even in the absence of a common software framework or programming language, vastly improves close collaboration
- Defining an EDM is not entirely trivial
  - LCIO was developed by physicists experienced in software development from several different labs (Tevatron, SLC, LEP, HERA) working in close communication with detector and analysis physicists in the wider linear collider community (NLC, Tesla, JLC)

More Lessons Learned

- Keep it simple!
  - Resist the urge to add every bell and whistle
  - Although the LCIO EDM was iteratively extended and improved over the years, it was essentially mature a year after its introduction
- Make it accessible
  - Allow for easy access via multiple languages
- Make it easy to use
  - Make simple things simple to do
    - provide convenience methods, utilities
  - Make complex things possible
    - allow for user extensions (within reason)
- Simplify! Simplify! Simplify!

Code Infrastructure

- Open source https://github.com/iLCSoft/LCIO
- Implementations / bindings in
  - C++, Python, go, Java, Fortran, Julia
    - The straightforward implementation of the libraries allowed the Julia bindings to be implemented basically in a weekend
- API Documentation:
  - https://ilcsoft.desy.de/LCIO/current/doc/doxygen_api/html/index.html