GEANT4 for Future Linear Colliders

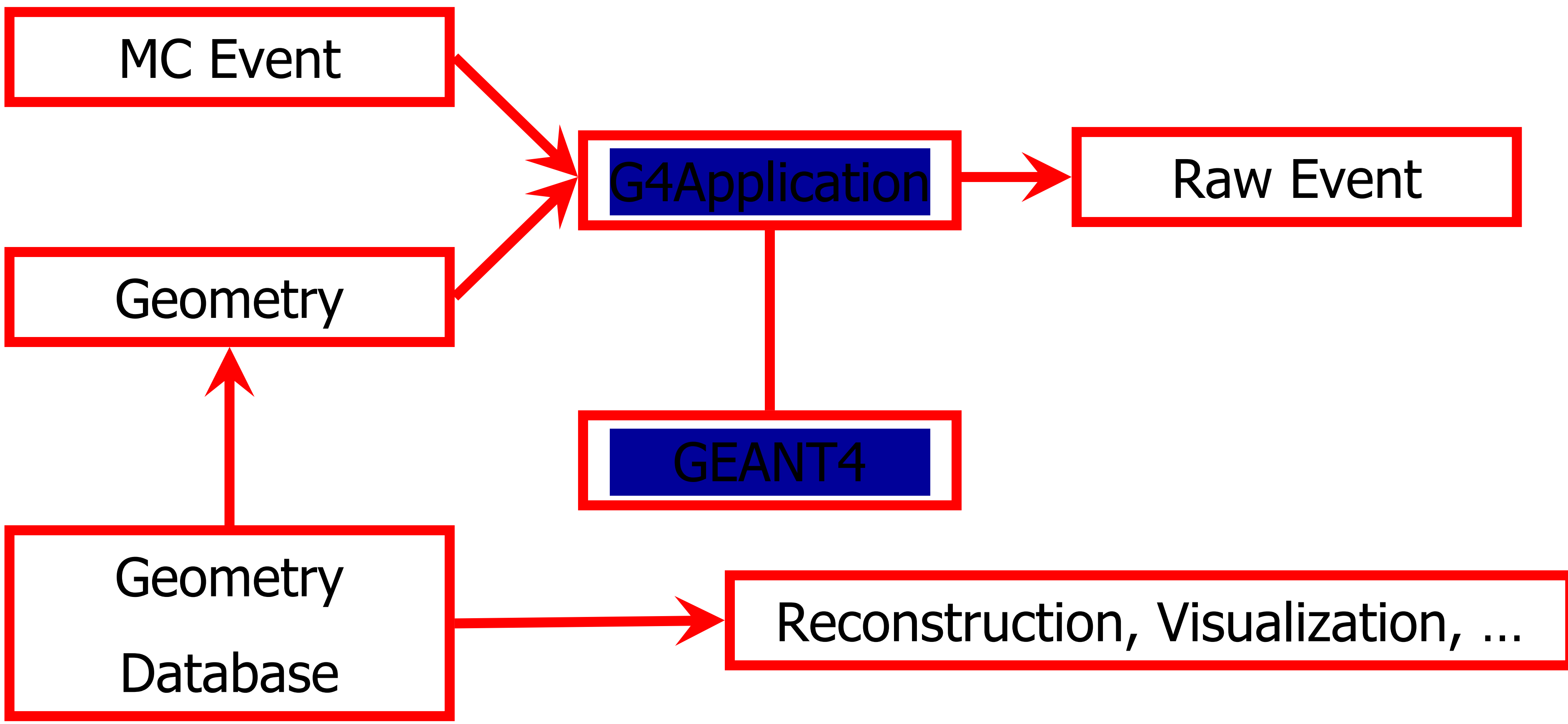
Geant4 Workshop @ DESY

October 2, 2003

Linear Collider Environment

- Exploit the physics discovery potential of e^+e^- collisions at $\sqrt{s} \sim 1$TeV.
- Precision measurements of complex final states require detectors with:
  - Exceptional momentum resolution & vertexing.
  - Imaging calorimetry for “Energy Flow” analysis.
- Common simulation environment for all LC studies would allow sharing of detectors, algorithms, and code.
- The system should be flexible, powerful, yet simple to install and maintain.

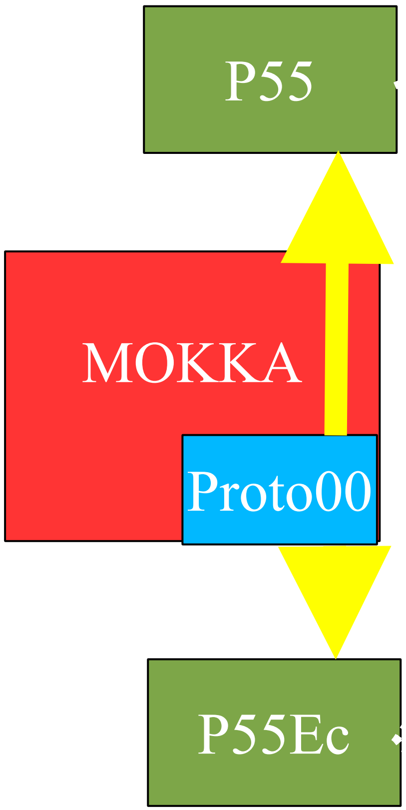
LC Detector Full Simulation

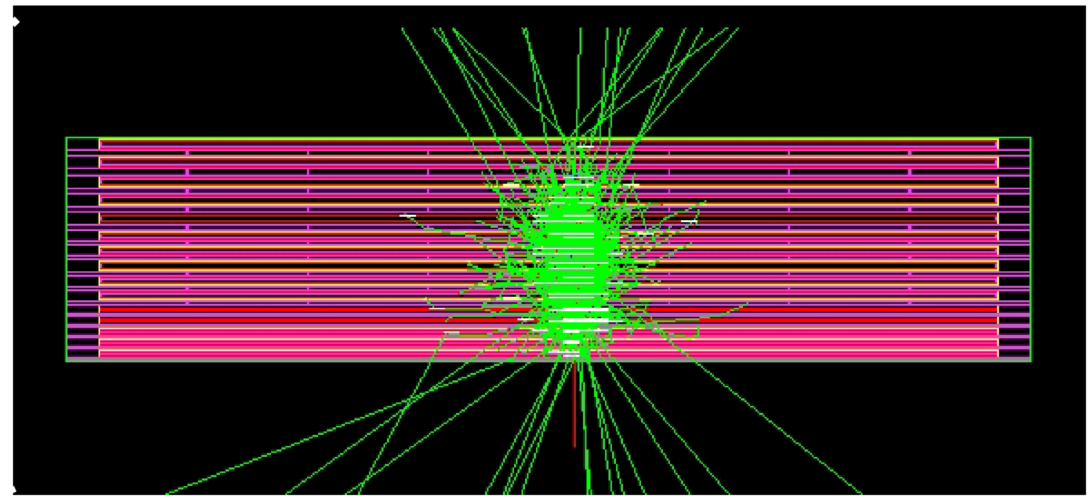


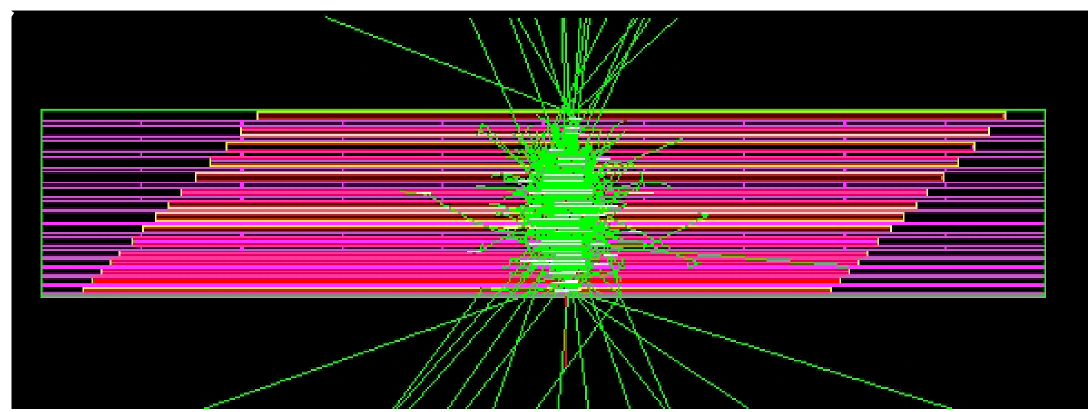
Mokka

- Geant4 full simulation for the Tesla detector.
- Uses subdetector-specific geometry drivers.
  - Relevant parameters stored in MySQL database.
  - Tight coupling between Sensitive Detector and geometry volume definitions.
- LCIO persistence for generic hits & MC chain

The Proto00 geometry driver



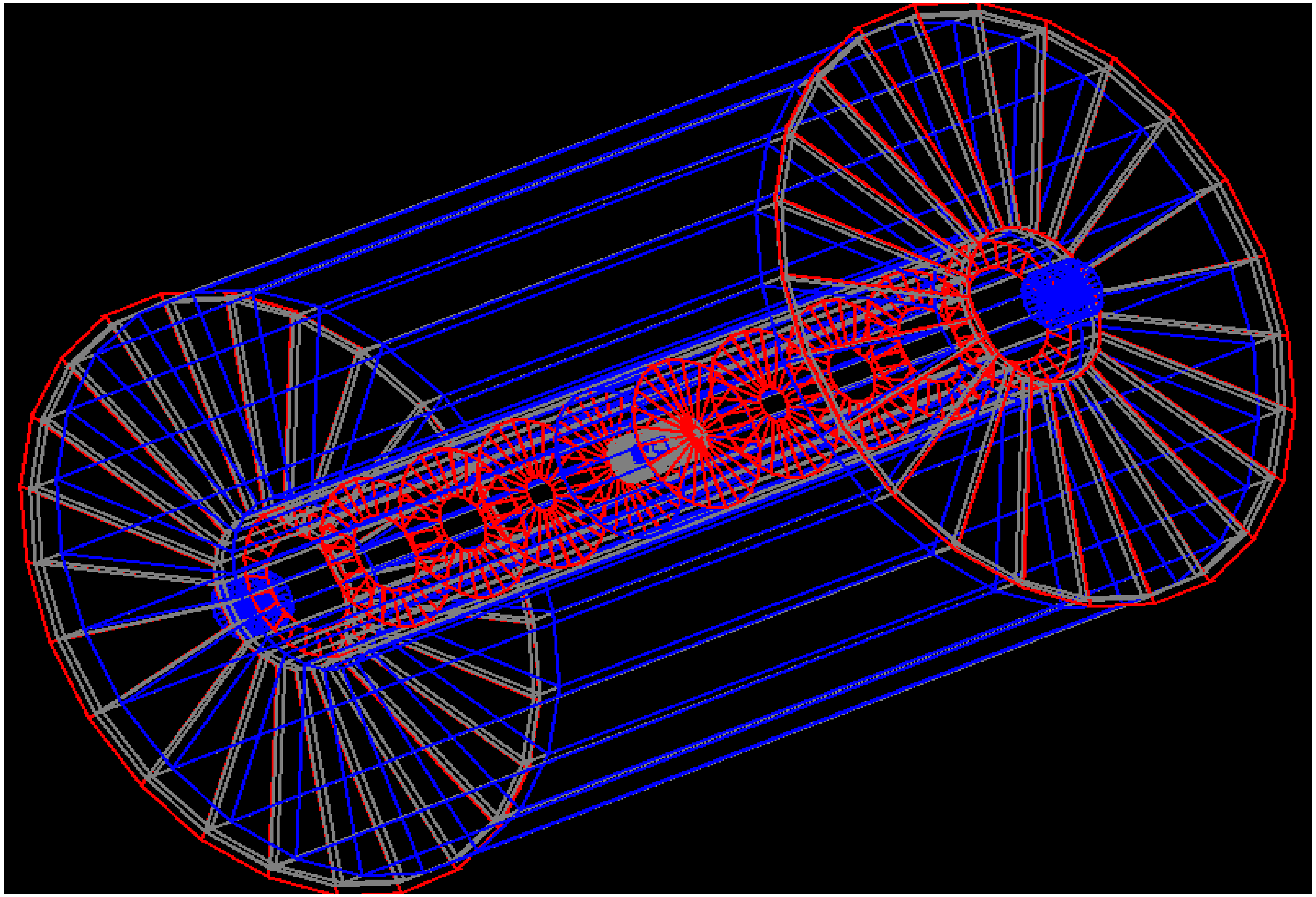




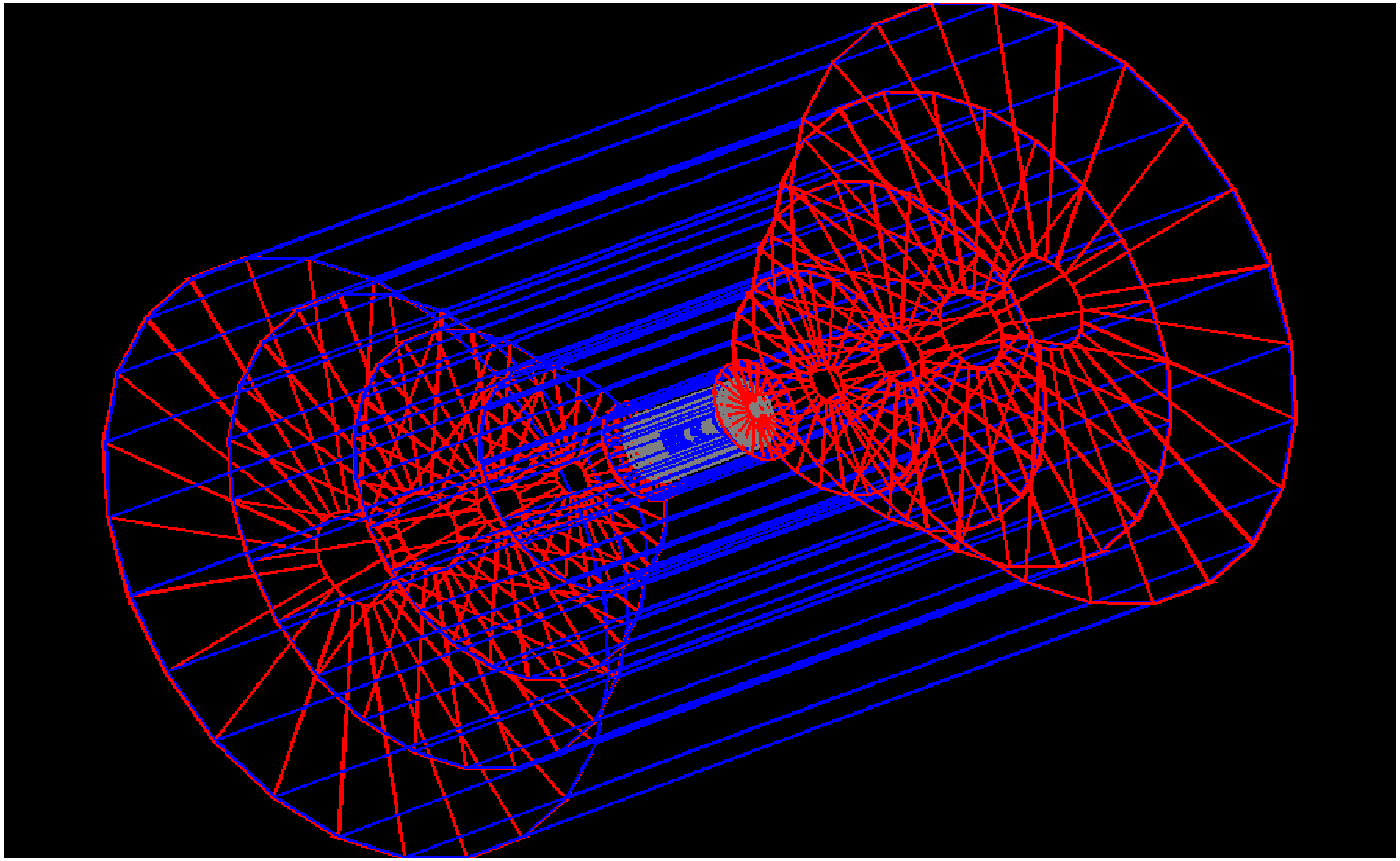
LCD Full Simulation

- Geometry defined in XML.
  - Flexible, but simplified volumes.
  - Projective readout of sensitive volumes.
- Dynamic topology, not just parameters.
- Have defined generic hit classes for sensitive tracker and calorimeter hits.
- Root and LCIO bindings for I/O.

TPC Tracker, Si Disks, CCD VTX



All Si Tracker, CCD VTX



Generic Hits Problem Statement

- We wish to define a generic output hit format for full simulations of the response of detector elements to physics events.
- Want to preserve the “true” Monte Carlo track information for later comparisons.
- Want to defer digitization as much as possible to allow various resolutions, etc. to be efficiently studied.

Types of Hits

- "Tracker" Hits
  - Position sensitive.
  - Particle unperturbed by measurement.
  - Save "ideal" hit information.
- "Calorimeter" Hits
  - Energy sensitive.
  - Enormous number of particles in shower precludes saving of each "ideal" hit.
  - Quantization necessary at simulation level.

Hits Summary

- Storing "ideal" hits gives detailed information about MC track trajectory.
- Deferring digitization allows studies of detector resolution to be efficiently conducted.
- Can approximate the same in calorimeter by defining small cells, then ganging later.

LCIO

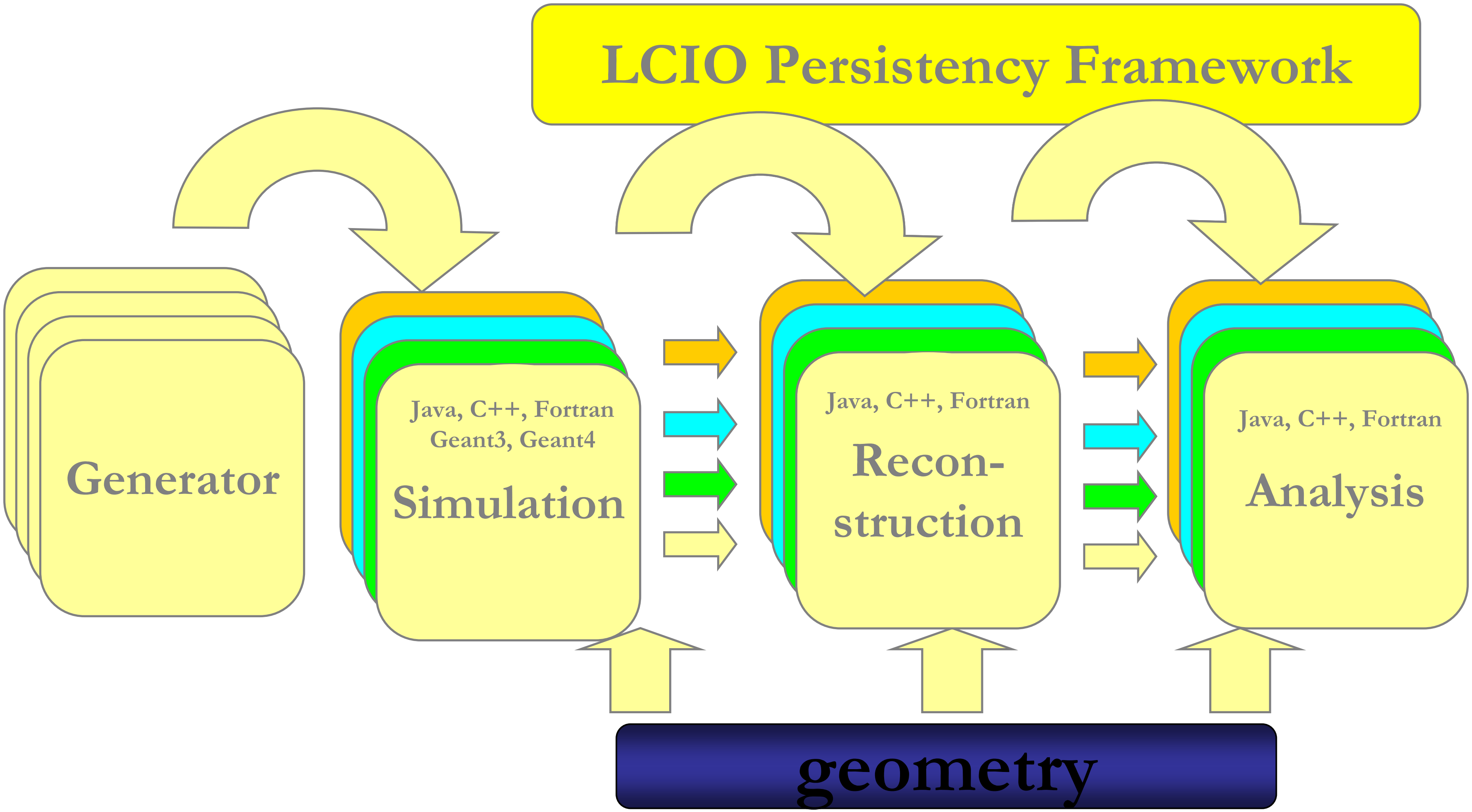
- Persistency framework for LC simulations.
- Currently uses SIO: Simple Input Output
  - on the fly data compression
  - some OO capabilities, e.g. pointers
  - C++ and Java implementation available
- Changes in IO engine designed for.
- Extensible event data model
  - Generic Tracker and Calorimeter Hits.
  - Monte Carlo particle heirarchy.

LCIO (II)

- Persistency framework for LC simulations.
- Java, C++ and f77 user interface.
- LCIO is currently implemented in simulation frameworks:
  - hep.lcd
  - Mokka/BRAHMS-reco

  -> other groups are invited to join

LCIO Motivation



Towards Internationalization

- Suggest that Tesla, NLC and JLC full simulation groups could run a single GEANT4 executable.
- Geometry determined at run-time (XML).
- Write out common “ideal” hits.
- Digitize as appropriate with plug-ins.
- Enormous savings in effort.
- Makes comparisons easy.

Full Simulations

LCDROOT/LCDG4 MOKKA JUPITER

Common GEANT4 executable

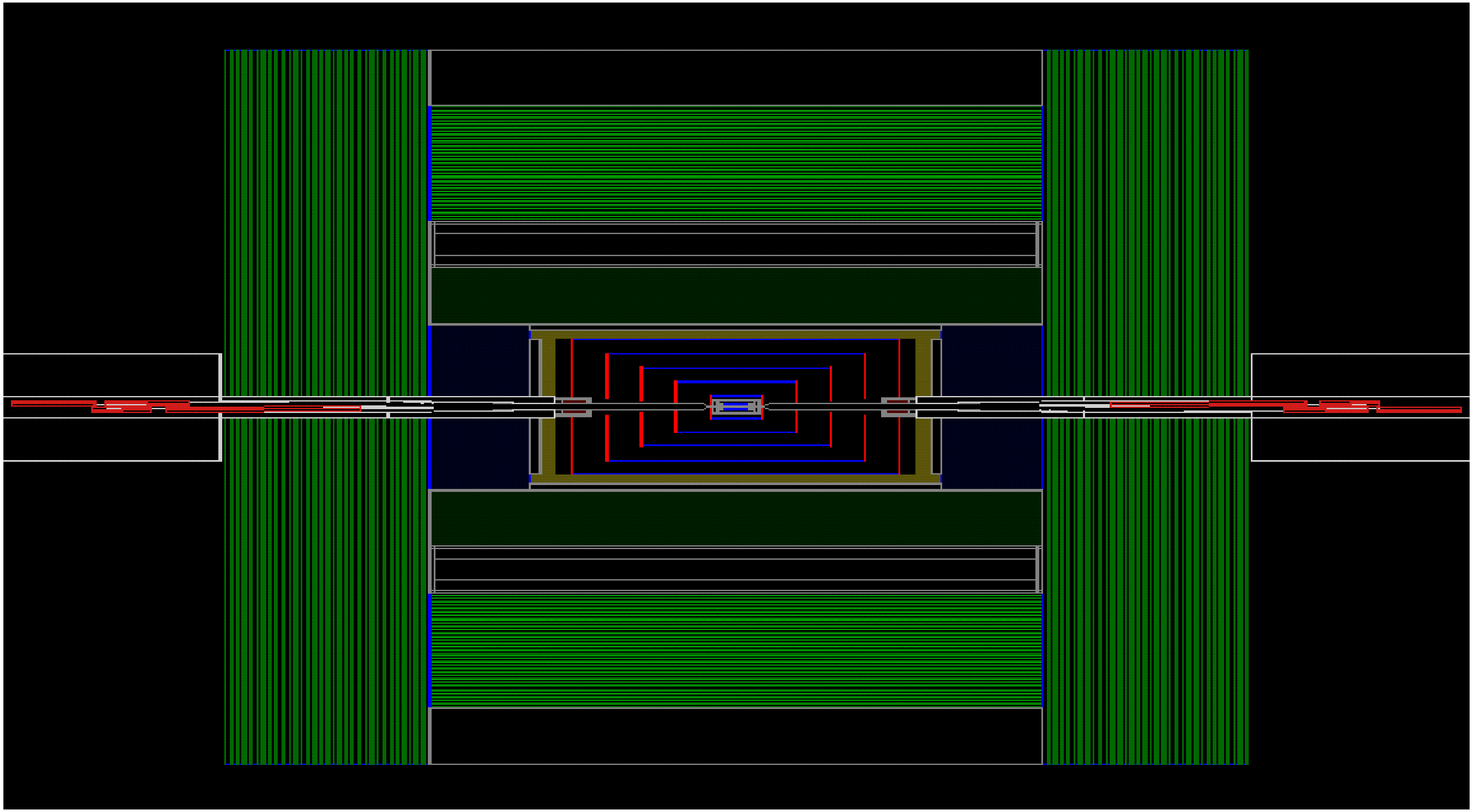
Runtime geometry

Generic Hit output

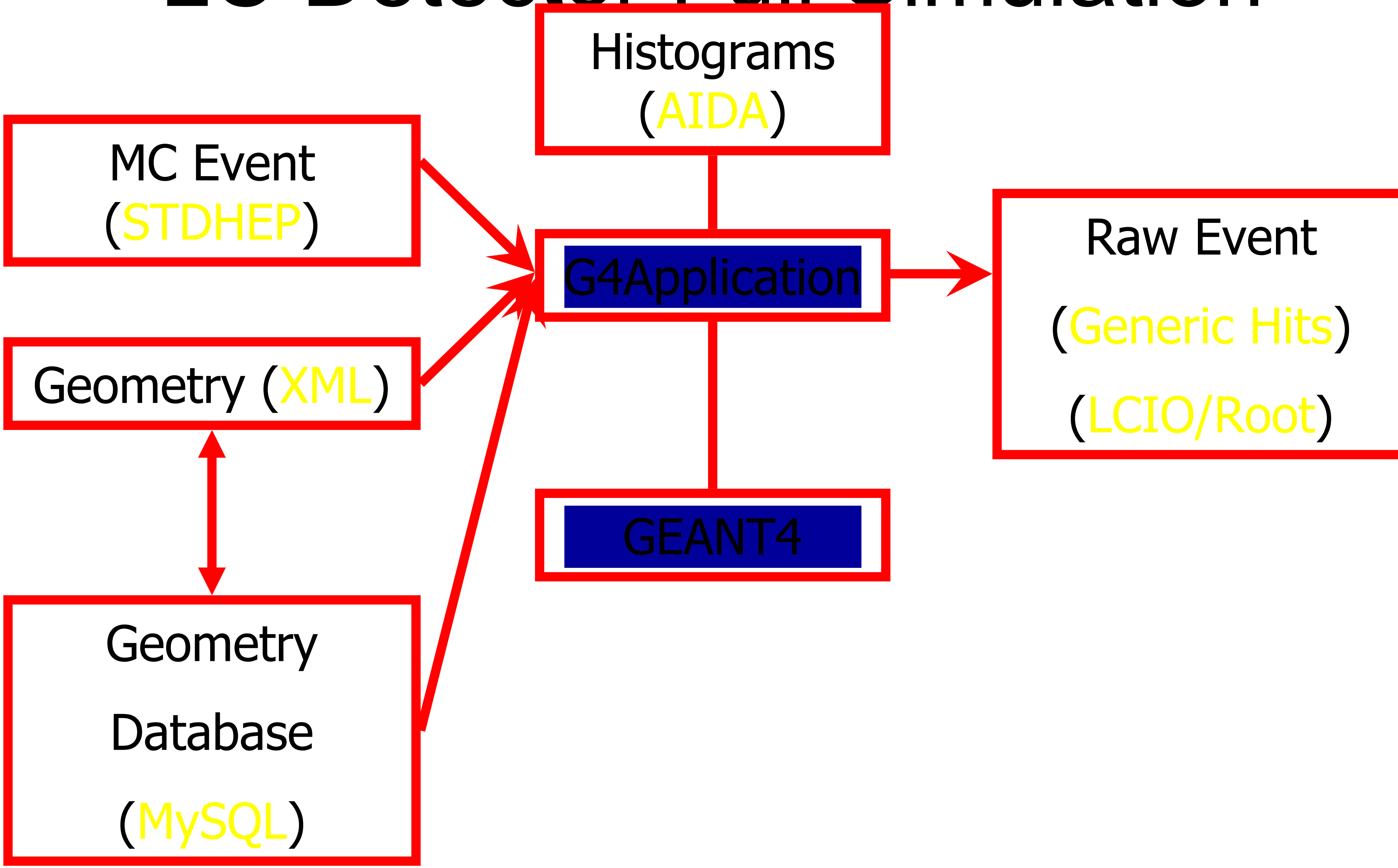
LCD/Mokka

- First version of mysql / xml interface exists
- SD detector fully modelled including beamline elements.
- Several TESLA detector versions modelled.
- LCIO output implemented in beta version.
- Interfaces to HEPEVT and STDHEP and background files implemented.
- Interface to AIDA integrated.

SD in Mokka



LC Detector Full Simulation



Main Simulation Issues

- Need flexible method to describe geometry.
  - Prefer G4 supported geometry input (GDML?)
- Beam Delivery System requires arbitrary magnetic fields, excellent tracking precision.
- Tracking System: $\delta(1/pT) \leq 5x10^{-5}$ GeV/c
  - Multiple Scattering, tracking precision.
- Jet Reconstruction: $\delta E/E \sim 30\%/\sqrt{E}$
  - Excellent hadronic shower simulations.

Highlights of LC Geant4 Effort

- Common executable, with runtime geometry.
  - Detector designs compared on equal footing.
- Generic hits for trackers and calorimeters.
  - Simplifies Sensitive Detector implementation.
  - Post-GEANT digitization $\rightarrow$ design flexibility.
- Lightweight persistence format (LCIO).
  - Allows interchange of data between communities.
  - Common target for Java, C++ & Fortran analyses.

Why XML?

- Simplicity: Rigid set of rules, plain text
- Extensibility: Add custom features, data types
- Interoperability: between OS and languages
- Self-describing data
- Hierarchical structure $\leftrightarrow$ OOP
- Open W3 standard, lingua franca for B2B
- Many tools for validating, parsing, translating
- Automatic code-generation for data-binding

Why G4 XML?

- XML Schema very useful for “compile-time” type safety and bounds checking.
- Prefer a G4-supported XML-based solution.
  - Had hoped for common LHC solution.
  - Investigated GDML.
    - Looks promising.
    - Sensitive detector definitions needed.
    - Support?