

# The Soft Side of the Global LC Collaboration: Detector Simulation Efforts

---

Dhiman Chakraborty

NICADD/NIU

LC software workshop  
ANL, 02-05 June, 2004

## Outline

- Current status in America, Asia, Europe.
- A compelling case for worldwide collaboration: the LCD Simulation Working Group.
- Toward a common program: stating the requirements.
- Milestones, plans, personpower needs.
- Summary

# Status: LCDG4 (America, NICADD+SLAC)

---

- Derived from LCDRoot
  - Root dependency removed.
  - Bugs fixed, features added.
  - XML for geometry description.
  - MC I/P: binary StdHep, O/P: sio.
- Relative strengths:
  - Lightweight, portable, flexible, fast.
  - Detector hits linked to MC particles.
  - Nice visualization+analysis tools.
- Relative weaknesses:
  - Limited geometry options (only cyl. shapes, projective towers in  $\theta, \phi$ ).
  - Geometry description messy to edit, error-prone.

## Status: Mokka (Europe, LLR+DESY)

---

- Developed with TESLA in mind
  - Cross-checked w/ LCDG4 (DESY+NICADD).
  - MySQL for geometry description.
  - MC I/P: ASCII StdHep, O/P: Icio (untested).
- Relative strengths:
  - Realistic geometry description.
  - Good model to separate Geometry persistency from main program.
  - Wide development base - adopted for CALICE TB (NICADD+DESY).
- Relative weaknesses:
  - Somewhat unwieldy, & slow.
  - Geometry model does not completely adhere to design principles.

## Status: Jupiter (Asia, KEK)

---

- Relative Strengths:
  - Modular (peripheral tasks handled by “satellites”).
  - Good support for hierarchical structures.
- Relative Weaknesses:
  - Completely Root-based (for most persistent I/O, user interface, analysis).
  - We don't know enough about it.

## Common weaknesses

---

- Numerous restrictions due to bottom-up development.
- Not designed to take full advantage of GEANT4, other tools.
- Geometry description limited in run-time flexibility/realism.
- Geometry hard to edit, visualize.
- No modeling of non-uniformity, noise, sensor characteristics.
- Core programs too dependent on I/O persistency models.
- Lacking proper code repository, release mechanism.
- Lacking good data server/model.
- Not well-suited for design optimization.

## But let us not forget that

---

- We're trying to break new ground on many fronts:
  - Simulating a detector with so many detector elements.
  - Using GEANT4 for design optimization.
  - Trying to keep many options open (a price to pay for global acceptance?).
  - Aspiring to implement ambitious design principles.
- We've been working with meager resources.
- People from all around the world are teaming up ...

## The worldwide LCD simulation WG

NIU/NICADD	D. Chakraborty, G. Lima, J. McCormick, V. Zutshi
SLAC	M. Asai, R. Cassell, N. Graf, T. Johnson
DESY	T. Behnke, F. Gaede, R. Pöschl
LLR/IN2P3	P. M. de Freitas, G. Musat, H. Videau
Cambridge	D. Ward
KEK	K. Fujii, A. Miyamoto

- Preparing the Requirements Document:  
Draft v3.0 posted on  
<http://forum.linearcollider.org>
- Next: analysis, design, & implementation of the complete software system.

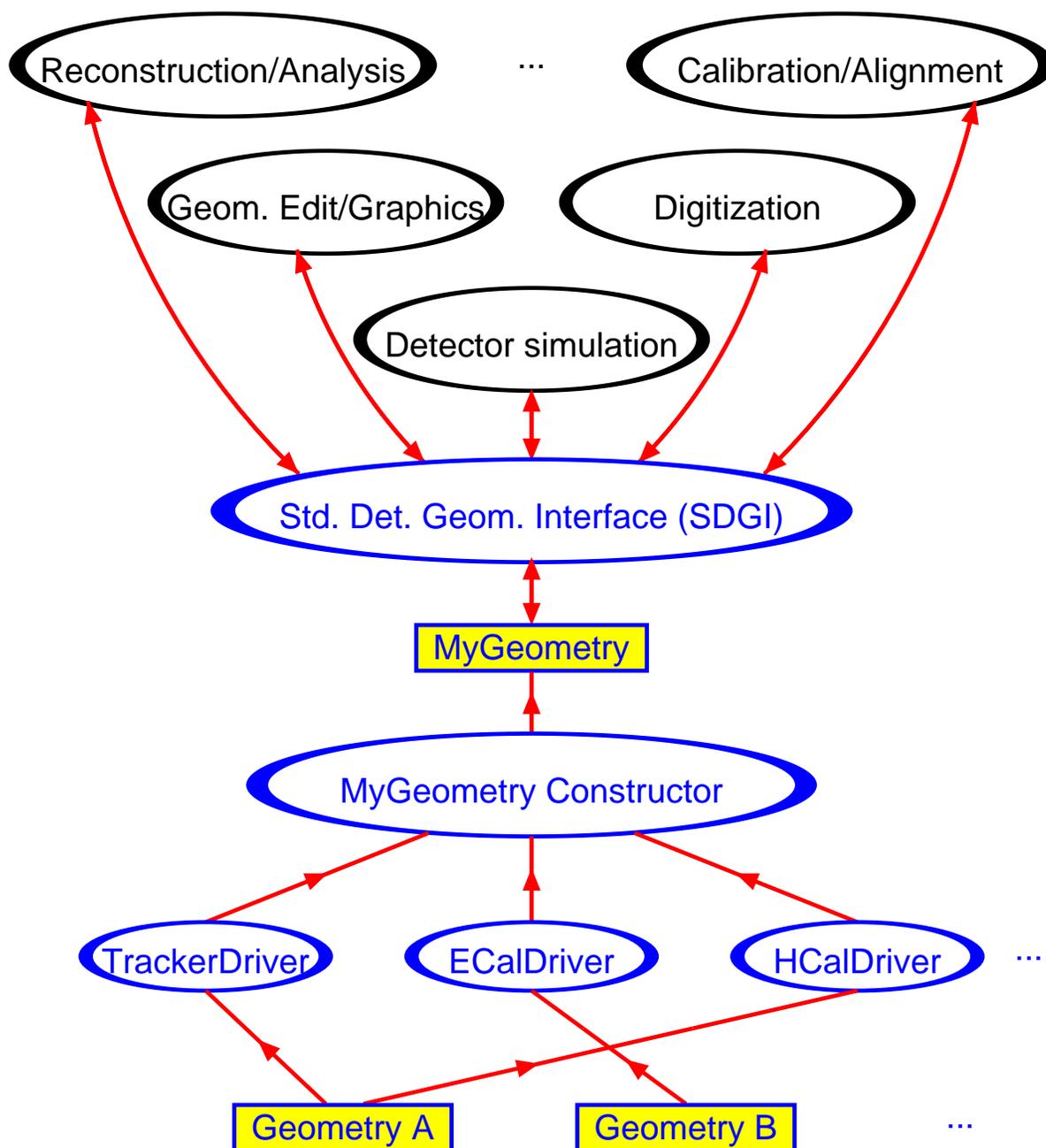
# Requirements for Simulation Software

- **Framework:** Same general structure as other applications:
  - Initialization: select geometry, MC inputs. Set run-time parameters, data processing, filtering/streaming, book histos ...
  - Event loop: select input events, process, select output events, fill histos ...
  - Summary: produce job log, close files.

Use the same framework as reconstruction/analysis? (Design requirements for that is a separate task.)

- **Geometry:**
  - Furnish complete description of detector at run-time: shapes, volumes, materials, fields.
  - All current geoms must be supported.
  - All GEANT4 shapes & volumes must be supported.
  - Hierarchical, recursive structure (like a directory tree) of elements.
  - Element replication arbitrary placement, orientation, independent scaling along principal axes.
  - Must be easily extensible.
  - Tag for each major subsystem.
  - Capable of running in stand-alone mode.

- **Geometry (contd.):**
  - **Volume segmentation:** Store hits in arbitrarily small units, or “virtual cells”. Group these together in different ways to form different shapes and sizes of physical cells.
  - **Common Geometry API:** All main programs should be completely independent of persistency mechanisms. They should access & save data using brokers compliant to a standard interface. Geometry-dependent applications include simulation, digitization, calibration, alignment, reconstruction, analysis, geometry editor, visualization etc.



- **Geometry editing and visualization:**  
Element-by-element editing, verification, and visualization of the geometry should be offered through a menu-driven user-interface that is completely independent of the geometry persistence model. One should be able to choose a reasonable range of GEANT4 shapes for individual detector elements, and put them together through a series of forms and menus. When done, one should be able to check whether the whole geometry is GEANT4-legal, and graphically examine it for correctness.

- **User Interface:**

- GEANT4 command-line interface is powerful, but difficult for novice users.
- Need to add a simple and intuitive GUI for the average user.
- In addition to normal G4 commands, it should offer access to utility functions & macros.

- Digitization:
  - A post-processor to convert the GEANT4 output (spacepoint+energy) to “raw data” format (electronic address+sensor response) for reconstruction.
  - Hooks for user-definable transfer functions (incl. noise, inefficiencies etc.)
  - Preserve MC truth.
  - Options to save output, or pipe into reconstruction on-the-fly.

- **Graphics:**

- GEANT4 native graphics not suitable for virtual cells.
- Display of detector elements with translation, rotation, zoom.
- Measurement of lengths and angles.
- 3D to 2D projections, switching between views.
- MC particle trajectories.
- Interactive association of detector hits & MC particles and vice-versa.
- Attributes of a set of detector elements selected interactively.
- Attributes of a set of MC particles selected interactively.
- Must not be too slow.

- **Persistency:**
  - Should be handled by API-compliant “brokers”.
  - Must support StdHep binary for generator MC input.
  - Root offers rich data structures, but has idiosyncrasies.
  - LCIO (SLAC/DESY) accepted by America, Europe. Asia receptive.

- **Output contents:**
  - Must have, in addition to detector hits, all the MC particle info, and cross-links between particles and det hits.
  - Need to come up with a particle-vertex link scheme (conventions vary between generators).
  - Interactions of generated particles w/ detector need to be taken into account, e.g. bending of tracks in  $\vec{B}$ , scattering in material.
  - Option to save energies in non-sensitive volumes.
  - Time stamps to study pile-up issues.
  - “Run header” to contain info about run control parameters, input data & geometry, software versions . . .

- **Code management**
  - Need a CVS with a good web interface and notification mechanism.
  - Certification, build, and release procedures need to be established.
- **Data processing**
  - Program must be portable to laptops.
  - Production must be distributed.
- **Data archival & access**
  - Must archive and catalog all information needed to reproduce any file.
  - Data server should be distributed (rather than central).
  - Piggyback on an existing metadata system, or get on the GRID (not fully implemented yet)?

- **Forward compatibility**
  - Design should preclude as few options as possible.
  - Changes should be effected in a series of small increments, rather than a few large-scale overhauls.
  - Must have a useful version running at any given time.
- **Parallelization**
  - At some point, will need the ability to split large simulation requests between multiple nodes on a computing farm, and seamlessly merge the events back into one output.
- **GRID-compatibility**
  - All code should be GRID-friendly. Need expert advice on what that entails.

## Milestones, plans, personpower needs

- Milestones:

- 2005 - Test Beam: Opportunity to prototype the software system. Try to take a first shot at as much of the long-term solution as needed for simulation of TB modules. Program should be ready by summer, 2005.
- 2007 - CDR: Must have the capability to perform full-detector simulation with multiple design alternatives. Flexibility and speed more important than accuracy (i.e., go for parametrizations, “virtual cells”). Program should be ready 6-8 months before CDR submission.

- 2009 - TDR: Must be able to perform full-detector simulation one or two design alternatives, but with accurate detector descriptions. Program should be ready 12 months before TDR submission.

- Plans:

- Workable, though not ideal, TB solution in view.
- Will take some time to formulate the rest.

- Personpower needs:

- 12-15 FTE over the next 4 yrs.
- Can count 6-8 FTE right now.

## Summary

- Our ambitions are high, and resources limited.
- There is absolutely no argument against international collaboration on LC software in general, and detector simulation software in particular.
- Should follow a line similar to accelerator development.
- The initiative toward a global software framework is in a very early stage, but the signs look good so far.

- A coherent plan is in the process of being formulated - requirements being agreed upon, action items identified - we'll need to work efficiently and in tandem.
- We're somewhat anemic right now - need to double the effort (at least).
- Help needed especially from software engineers.