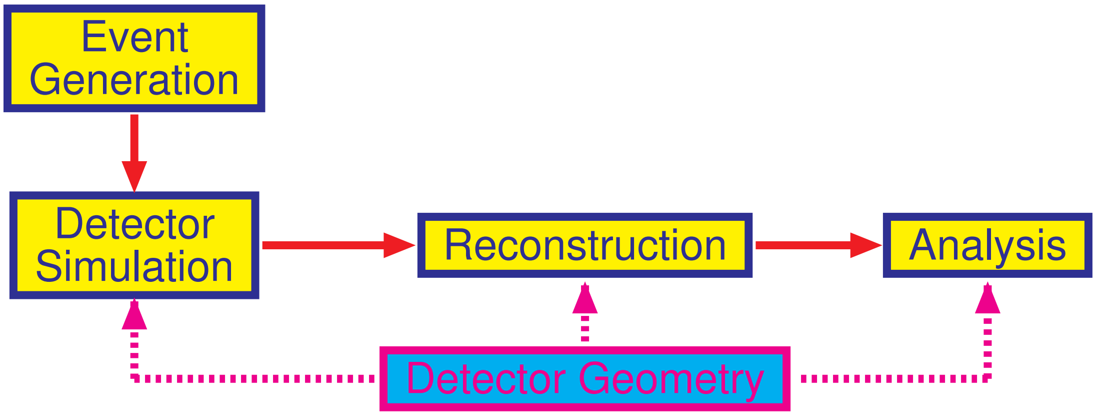
NLC simulation software:
needs, status, plans(?), . . .

Dhiman Chakraborty
NICADD/NIU

LCD muon mtg, 03 May, 2002

General scheme and guidelines



- Keep each step as independent of the previous ones as possible.
- Coupling between boxes only through data persistence, (I/O files) $\Rightarrow$ format choices (although combining reconstruction and analysis into a single box could be a good idea, if achievable).
- Maintain maximum possible flexibility.
  - within each box,
  - between boxes.
- The detector must be an external plug-in.

State of affairs in US, Europe, Asia

- US: LCD
- Europe: Tesla, CALICE
- Asia: JLC
- Event generation:
  - Components:
    - Parton generators (e.g. Pandora)
    - Parton evolution (e.g. Pythia)
    - Evolution plug-ins (e.g. Tauola)
  - World's experts converging to provide StdHEP output.

- Detector simulation:
  - Parametrized:
    - TESLA: SIMDET
    - LCD: fastMC
    - JLC: ?

- Full:
  - TESLA: Brahms
    - GEANT3,
    - Fortran,
    - not general/flexible (det params hard-coded),
    - $\sim$final, dead-end,
    - move to OO with LCD (?)
  - CALICE: Mokka
    - GEANT4,
    - C++,
    - general,
    - ASCII output,
    - promising candidate

* LCD: 1. GISMO
  - EGS/Geisha,
  - Fortran,
  - general/but not flexible (e.g. projective towers),
  - sio output,
  - final, dead-end.

  2. RootLCD
  - GEANT4,
  - ROOT for persistence,
  - general,
  - no output (sio output in the works),
  - merge with Mokka?

- Reconstruction:
  - TESLA:
    - Fortran,
    - move to OO with LCD (?)
  - CALICE:
    - ASCII output from Mokka reade by independent Fortran programs with no structure sharing,
    - must change soon.
  - LCD: 1. JAS
    - Java ($\Rightarrow$ platform-independent),
    - IDE,
    - reads data files in sio format,
    - easy to learn and use,

  * many nifty features,
  * many serious limitations (current version lacks good editor, memory management/persistence, non-projective geometry, ntuple-making, . . .),
  * promising candidate (limitations must be overcome).

2. RootLCD
  * ROOT/C++ (unfortunate coupling, to be removed),
  * a step or two behind JAS,
  * offers memory management, persistence via ROOT.

– JLC: ROOT chosen for both framework and persistence.

- Detector Geometry:
  - TESLA: mostly hardcoded.
  - CALICE: Mokka uses a GUI-equipped geometry database (MySQL).
  - LCD: ASCII for GISMO (soon to be abandoned), XML for JAS (hopefully to become a single standard for all?)
  - JLC: ?

Plans:

- Enormous amount of work is needed in many fronts.
- A reasonably complete system is needed in $\sim$ 6 months.
- US is struggling due to lack of person-power: Europeans are eager to collaborate, but more participation is needed in both development and testing.
- Common denominator for all efforts: hardware prototyping, algorithm development, physics analyses . . . without sound simulation software, everything will stall sooner or later.
- Can only succeed if all groups (physics, detectors, id, . . .) must pitch in.
- Everyone can make valuable contributions irrespective of his/her taste/expertise.

- NICADD (NIU/Fermilab) is taking an initiative out of interests in Calorimetry+Muon systems. We are eager to join hands with others to reach the goals.