January 20, 2003

Summary of Meeting at LLR-Ecole Polytechnique, 14 Jan.-15 Jan. 2003

People:
Frank Gaede (author of summary)
Paulo Mora de Freitas
Henri Videau (partly)
Jean-Claude Brient (partly)
Ties Behnke, Norman Graf, Tony Johnson (phone meeting)

1. Goal

The goal of this meeting was to continue the discussion on the common persistency framework and the data model for Linear Collider simulation studies (LCIO) started at SLAC in December and the integration of LCIO into the Mokka framework. This is a summary of the discussions during those two days at LLR and the phone meeting. It also incorporates some communication via email that happened since.

2. Topics

The following topics were discussed:

- Data model for generator output (simulation input)
- Data model for simulation output
- Data model for reconstruction output
- Implementation
- Future steps

3. Outcome

3.1. Data model for generator output

The input to the Mokka simulation is based on the HepEvt-interface of Geant4. This interface definition consists of a subset of the HepEvt common block. In particular it doesn’t have any vertex information. From this it is clear, that generator events cannot contain secondary vertices (e.g. from B-mesons). Instead those decays have to be done by Geant4 and the vertex information has to be taken from there.

3.2. Data model for simulation output

Some small modifications have been made to the data model for simulation output. These involve mainly naming and the redefinition of the flag-bits for the CalorimeterHit block. At the time of writing SIO doesn’t allow to have several blocks with the same name in one record. Thus it was agreed to have a list of block names and types in the RunHeader.

The French group would like to add an additional geometry word to CalorimeterHit, that is needed for their geometry interface CGA. It has to be discussed whether this is going to be in the standard for all users of LCIO or whether we need to introduce another bit-flag to tag the existence of this additional word.

Another issue of discussion was the definition of particles that enter the MCParticle list. The idea is to have all particles in the list that leave a track in the detector, e.g. decays in flight (K_s) or delta electrons. The technical definition is not so straight forward, as it is unclear how to treat decays that happen between calorimeters or shower particles that back scatter from the calorimeter into the tracker. It was agreed, that people define what they consider an optimal solution/definition in order to get some implementation that is practical. Some proposals that involve energy cut offs have been made since (e.g. see Ties memo on LCIO webpage).

3.3. Data model for reconstruction output

Quite some time of discussion was spent on the reconstruction output data format:
For the clusters information on the intrinsic direction (tracking calorimeter) and its errors has to be added. Three parameters for the shower profile are probably not enough (what is the number needed?). Reconstructed objects/particles need some parameters for PID hypotheses (how many and which?). Vertex information has to be stored for reconstructed particles – do we need several vertex hypotheses in the event ?
Is there a 'complete and exclusive list' of particles in the event, i.e. particles at the IP?
Or will a list of reconstructed particles contain mothers and daughters with corresponding flags for a loop over particles at the IP (see Ties considerations on the web page)?
It was agreed to further discuss these issues and compare with other experiments' (LEP) data models. It was also considered useful to have examples in writing that explain how v_zeros (K_s), gamma conversions, etc. would be stored in the persistency scheme.

3.4. Implementation

Tony agreed to implement the Java version of LCIO (adopt the LCIO interfaces for hep.lcd) and to create a CVS repository for code management and documentation.
I will implement the C++ version of LCIO and probably the f77 interface as well, as this will be a wrapper interface for the C++ version.
Paulo is going implement LCIO into the Mokka framework, once the definition of the API is stable.
Ties will adopt LCIO (f77) for the Brahms reconstruction program.
It was agreed that it would be desirable to have at least one simulation and reconstruction program using LCIO by the Amsterdam ECFA/DESY workshop.

3.5. Future development

It would be desirable to have a reconstruction program in the near future that is flexible in terms of the geometry used for simulation. One option would be to use the CGA (common geometry interface) of the Mokka framework for the geometry definition in BrahmsReco, albeit this is feasible for the track reconstruction, it seems to be not so trivial for the calorimeter part. Another option would be use existing Java reconstruction code from the American group together with CGA and Mokka. The French group offered to provide a JAVA interface to CGA (using JNI) if people declare the necessity for such an interface.

Paulo agreed to open Mokka to a larger group of developers if this is desired. A scheme of how responsibilities (sub packages, e.g. for different sub detectors) are shared would have to be defined. Also Mokka could serve as a basis for a more open 'open source like' simulation framework that uses a more general way of defining the geometry so it could be used by different groups at the same time. Right now, in Mokka a separate set of geometry drivers has to be provided for every detector design.