Workpackages

Core Framework

a framework has to be set up that

- builds under Java 1.4 (Maven, ant,....)
- builds with gcj (make)
- reads xml files

•

- initially use compact format
- gradually extend/modify as needed
- · drivers need to read xml and implement gear
 - drivers should create geometry elements
 - for detailed view (low level)
 - ° for tracking view (mid level)
- initially export to GDML/LCDD should be suported

optionally some wrapping C++ code should be provided that allows to use LCGO in a C++ programm w/o having to rely on references to Java libraries, e. g. for the properties interface

- need to understand garbage collection for java and C++ objects allocated on the stack and on the heap and passing object between java and c++ java objects are objects inheriting from java::lang::Object
 - C++ are all other objects
 - Possibly interesting article http://gcc.gnu.org/ml/java/2004-03/msg00159.html

Material Database

a material data base is needed that

- · consists of the known materials
- · has all the material properties needed for dE/dx • A, Z (mean values), X0, Interaction length,....
- $^{\circ}~$ check geant4/geant3 which these are
- is extensible by the user in input xml files ٠
- provides material objects assigned to volumes (tracking) that give

 - material properties
 dEdx(PDG, p , flightlength)
 - optionally: dEdx(PDG, p , tracjectory) ?
- material properties/names should be consistent
- ° with geant4 • NIST database (the same ?)

Field map

LCGO needs to provide a mechanism for field maps

- should allow simple field definitions in xml file, e.g.
 - <BField x="0.0" y="0.0" z="4.0" unit="Tesla"/>
 - ° or better: <BField x="0.0" y="0.0" z="4.0*Tesla" /> (see System of Units)
- should allow arbitrary field map from file, e.g.
- o <BFIeld file="http://some.server.anywhere/ILCDetailedBField.map"/>
- need well defined file format and corresponding implementation class
- ٠ need well defined simple interface, e.g.
- interface Field { public Vector3D at(Vector3D x) ; }

Readout Properties

LCGO drivers need to provide an implementation of an interface to query cellIDs, positions and neighbors:

cellId <-> position

- cellid range (noise simulation)
- cell sizes

•

•

•

neighbors

Geometry Elements

we need a full set of geometry classes that allow

- · to construct the detailed low level view of the detector
- are used to create other representations

 geant4
 - GDML/LCDD
 - GDML/LCL
 HepRep
 - all classes currently available in geant4 should be represented ideally with the same interface
 - shapes (Box, Tube, Polycone,....)
 - logical volume (with material...)
 physical volumes with placement, i.e.
 - rotation
 - translation
 - parent volume
- · initially the volume heirarchy just needs to be static in order to allow creation of other representations
- issues/requiremnents
 - names
 - copy numbers
 - sensitive detectors
 - visualization attributes
 - ° user defined attributes

Tracking Geometry

for the tracking (pattern recognition and fitting) a somewhat simplified representation of the detector volumes is needed with

- averaged material (see material, dEdx)
- simple shapes only:
 - o boxes planes ?
 - ° tubes cylinders ?

this tracking geometry should allow to compute:

- intersection with given trajectory
 - ° line ° helix
 - ° ...

GEAR API

the mid and high level view of the geometry should be an extension of the current GEAR API

- · should allow to represent current detector concepts' subdetectors
- provide high level view as needed for
 - reconstruction
 - digitization
- should provide the geometrical functionality as needed for tracking
- should be the interface all client programs use to get information from LCGO (except low level view)

need to review what is currently in GEAR and define what is missing !

System of Units

it is probably desirable to have a well defined system of units a la Geant4, so that every quantity comes in the properly defined unit. So that one can write code, such as:

cout << " length of TPC [cm]: " << lcgo->getTPCParameters()->getLength() / cm << endl ;

Vectors and Matrices

need to define classes for vectors and matrices (symmetric) that are used throughout LCGO (and LCIO !?)

- 3d vectors
- ad vectors
 rotations 3d
 translations (3d vectors)
 4 vectors (in LCGO ?)
- matrix
- symmatrix
 ...