DD4hep Status

HEP detector description supporting the full experiment life cycle

- **Motivation and Goals**
- **Concepts and Design**
- **Implementation**
- **Ongoing work**
- **Summary**

Motivation and Goal

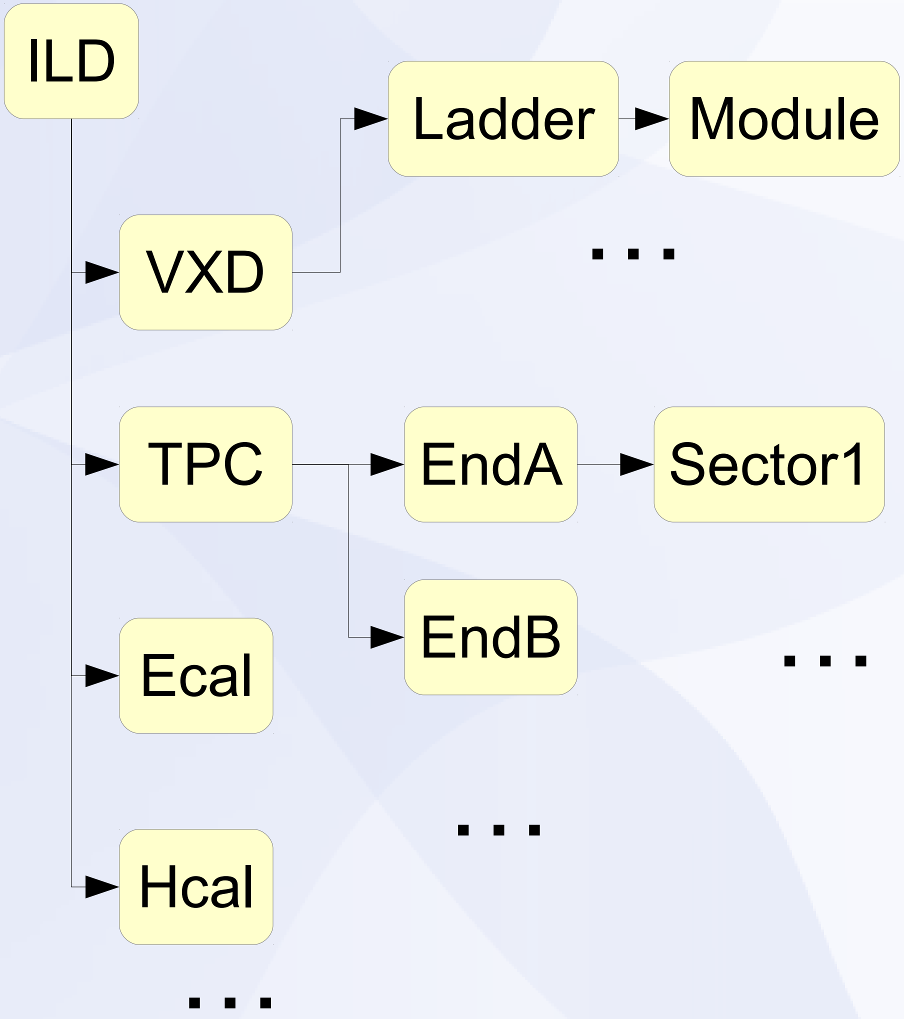
- Develop a detector description
  - For the full experiment life cycle
    - detector concept development, optimization
    - detector construction and operation
    - “Anticipate the unforeseen”
  - Consistent description, with single data source
    - Support for simulation, reconstruction, analysis
  - Full description, including
    - Geometry, readout, alignment, calibration etc.

  + standard commercials apply: simple usage etc.

What is Detector Description ?

- **Description of a tree-like hierarchy of 'detector elements'**
  - **A subdetector or parts of thereof**

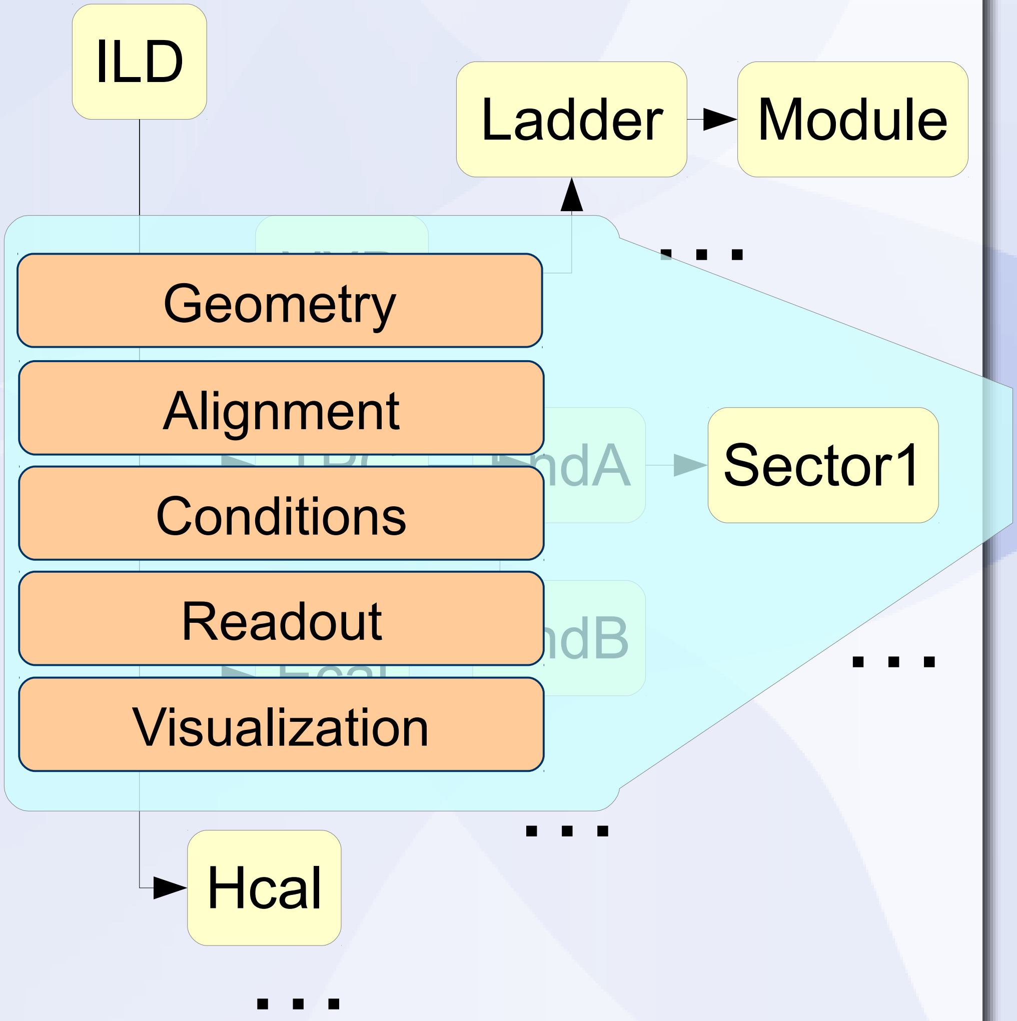
    Example:
    - Experiment
    - TPC
    - Endcap A/B
    - Sector
    ...



What is a Detector Element ?

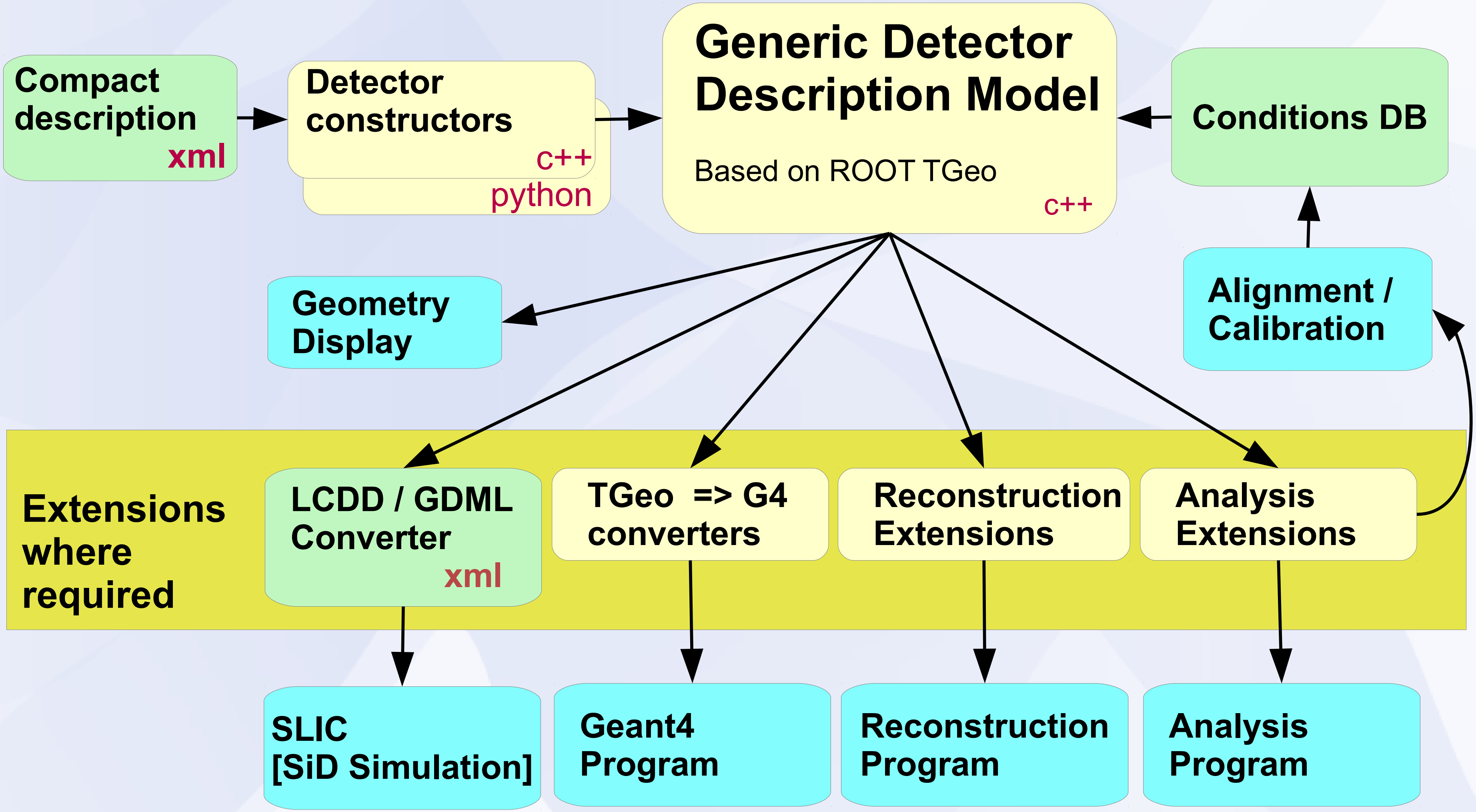
Significant piece of a subdetector including the description of its state

- **Geometry**
- **Environmental conditions**
- **All properties required to process event data (alignment etc.)**



- Motivation and Goals
- Concepts and Design
- Implementation
- Ongoing work
- Summary

DD4Hep - The Big Picture



Compact description xml

Detector constructors c++ python

Generic Detector Description Model

Based on ROOT TGeo

c++

Conditions DB

Alignment / Calibration

Geometry Display

Extensions where required

LCDD / GDML Converter xml

TGeo => G4 converters

Reconstruction Extensions

Analysis Extensions

SLIC [SiD Simulation]

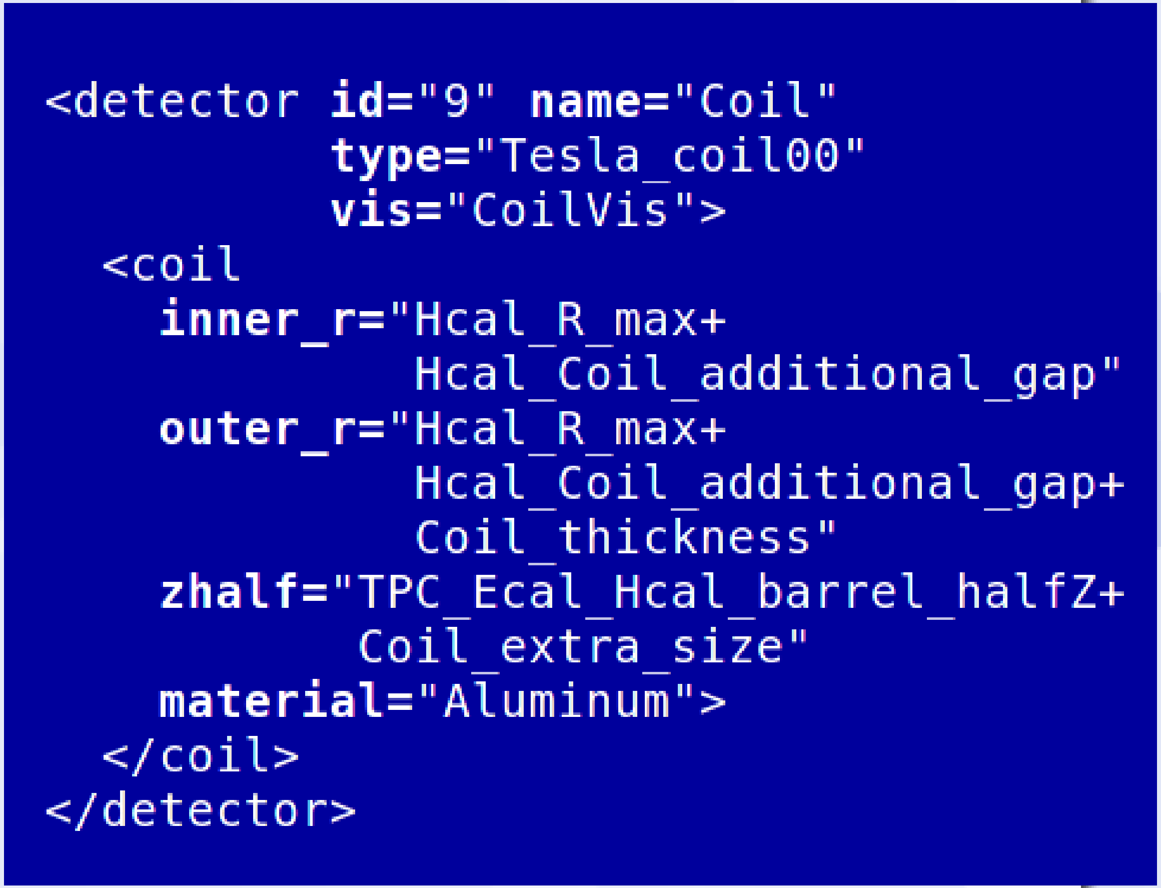
Geant4 Program

Reconstruction Program

Analysis Program

Compact Description – XML

- Human readable
- Extensible
- Interpreter supports units and formulas
- Parsed by DD4hep core



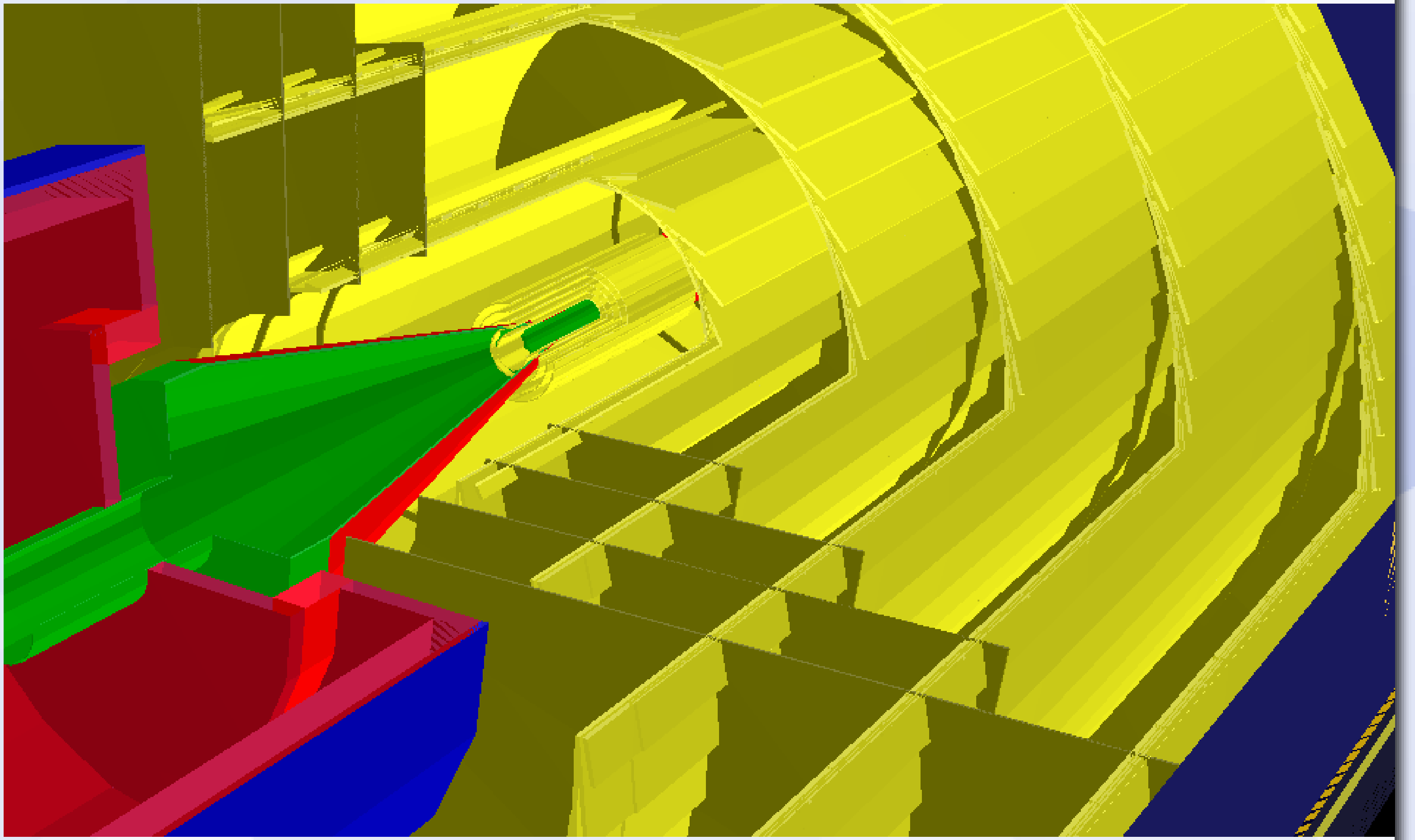
```
<detector id="9" name="Coil"
          type="Tesla_coil00"
          vis="CoilVis">
  <coil
    inner_r="Hcal_R_max+
             Hcal_Coil_additional_gap"
    outer_r="Hcal_R_max+
             Hcal_Coil_additional_gap+
             Coil_thickness"
    zhalf="TPC_Ecal_Hcal_barrel_halfZ+
           Coil_extra_size"
    material="Aluminum">
  </coil>
</detector>
```

Requires interpreting code to create 'detectors'

Display options

Display using native ROOT

OpenGL, Eve, etc.

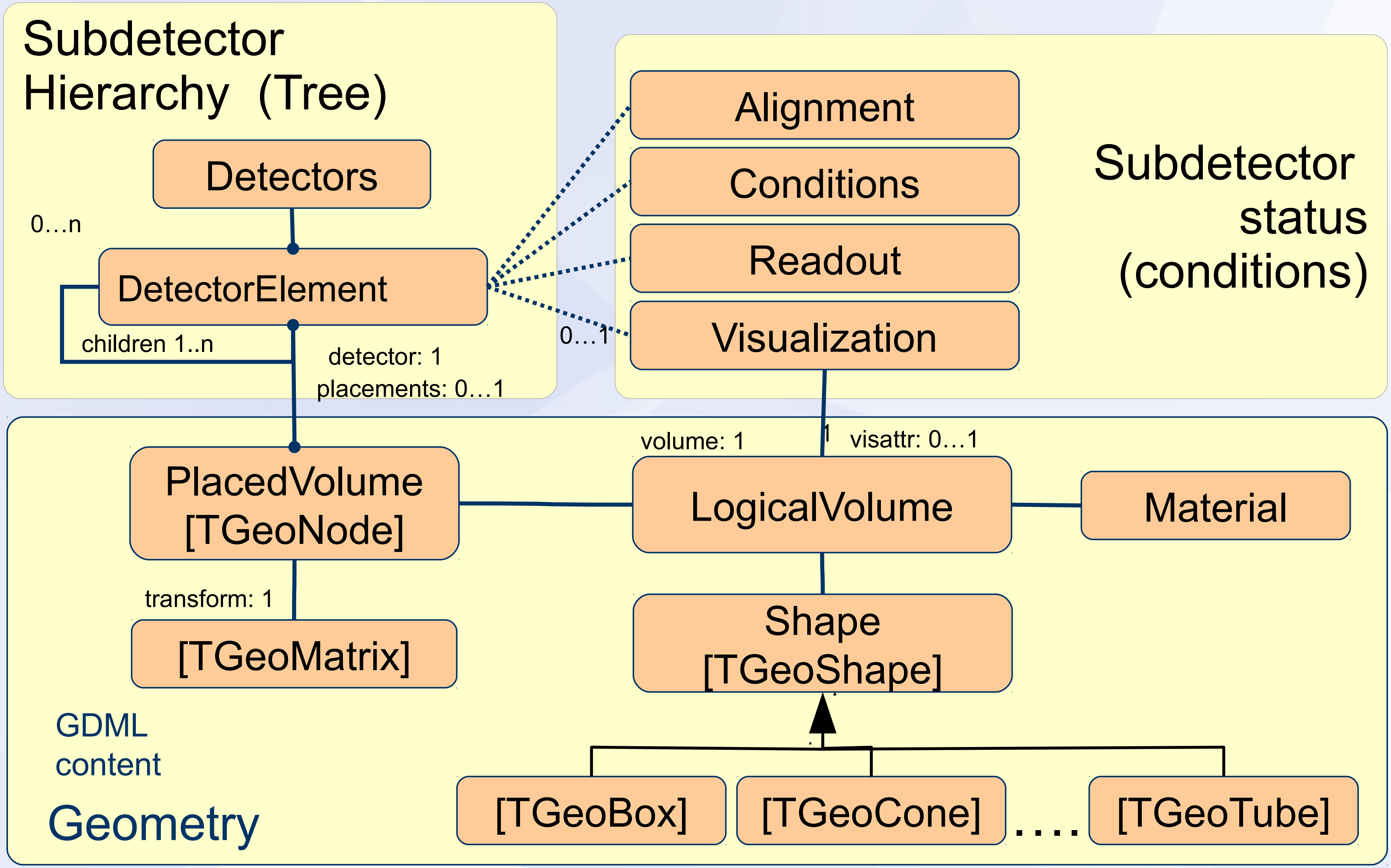


- Motivation and Goals
- Concepts and Design
- Implementation
- Ongoing work
- Summary

Implementation: Design Choices

- Detectors are described by a compact notation
  - Inspired by SiD compact description [Jeremy McCormick]
  - Flexible and extensible
- C++ model separation of ‘data’ and ‘behavior’
  - Classes consist of a single ‘reference’ to the data object
  - Same ‘data’ can be associated to different ‘behaviors’
- Implementation based on TGeo (ROOT)
  - TGeo classes directly accessible (no hiding)
  - TGeo has support for alignment

Implementation: Geometry



Subdetector Hierarchy (Tree)

Detectors

0…n

DetectorElement

children 1..n

detector: 1

placements: 0…1

0…1

Alignment

Conditions

Readout

Visualization

Subdetector status (conditions)

volume: 1

1

visattr: 0…1

PlacedVolume [TGeoNode]

LogicalVolume

Material

transform: 1

[TGeoMatrix]

Shape [TGeoShape]

GDML content

Geometry

[TGeoBox]

[TGeoCone]

….

[TGeoTube]

Deal with the Unforeseeable

- Different use cases require different functionality
  - Example: The use of the geometry is different in
    track reconstruction and alignment
    => specialized 'behavior' required
  - Example: Optimization of coordinate transformations
    local hit to experiment coordinates
    => specialized data required
    (cache of precomputed results)
- Object functionality is achieved by 'views' of public data describing a detector element
  - Same data are shared by all views (no copy of data)
  - User objects may be attached to data (extensions)

Implementation: Views & Extensions

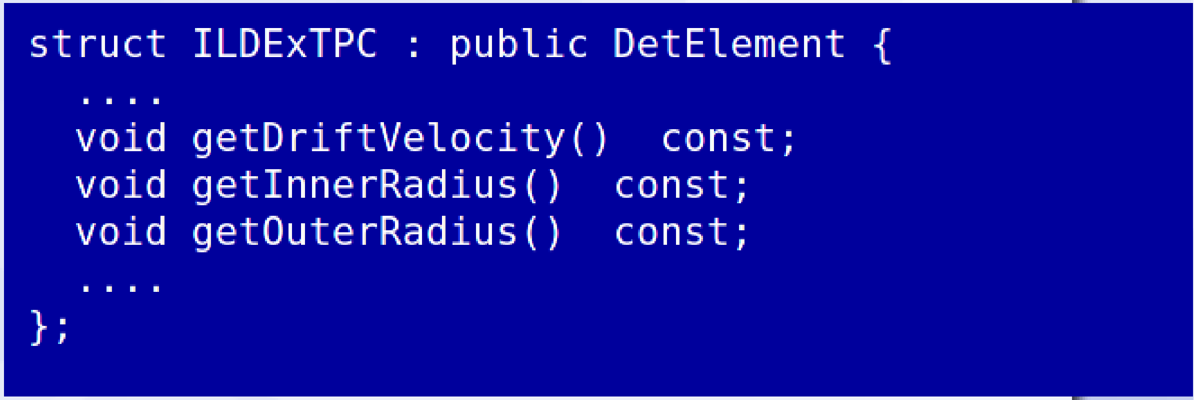
- Functionality achieved by 'views'
  - Corollary of the design choice to separate ‘data’ from ‘behavior’
  - Possibility of many views based on the same data
    - Same ‘data’ can be associated to different ‘behaviors’
    - All views are consistent
  - Views are 'handles' to the data
    - Creating views is efficient and fast
    - Typically only a pointer needs to be copied

Simple Client Views

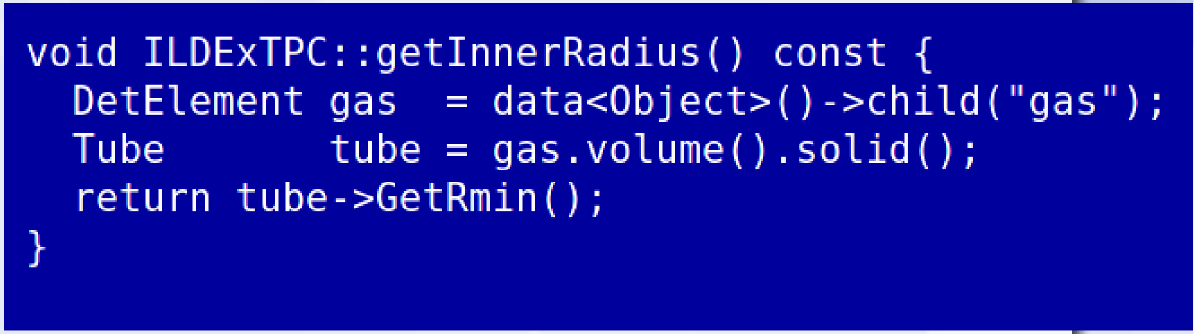
Views ensure

- **Convenience**
  - high level abstractions
- **Compatibility**
  - Details may change,
    but not the code
- **Optimizations**
  - Using data 'attachments'
- **Flexibility**
  - **Multiple views**
    depending on problem

Example: Convenience View



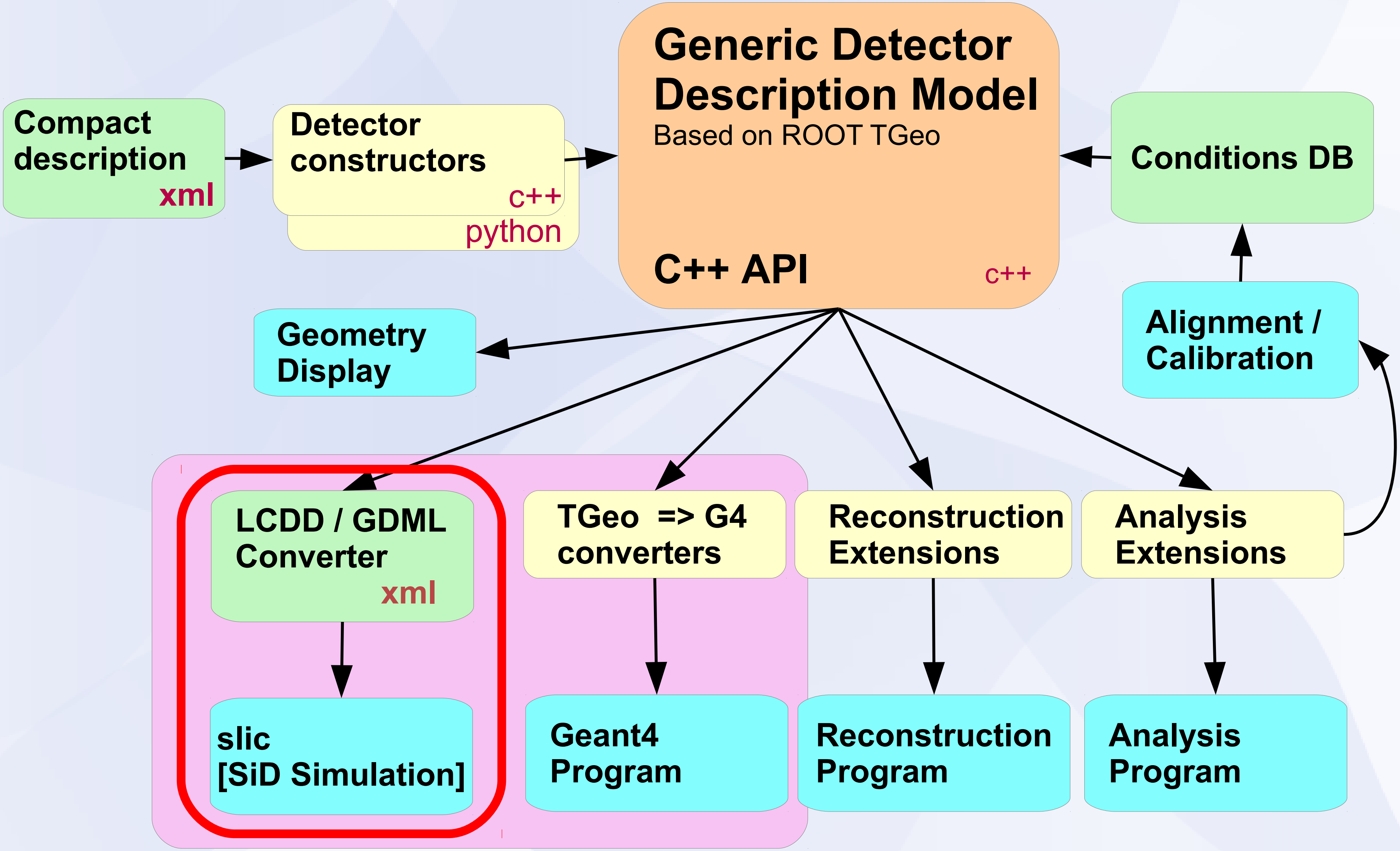
```
struct ILDExTPC : public DetElement {
  ....
  void getDriftVelocity()  const;
  void getInnerRadius()  const;
  void getOuterRadius()  const;
  ....
};
```



```
void ILDExTPC::getInnerRadius() const {
  DetElement gas  = data<Object>()->child("gas");
  Tube        tube = gas.volume().solid();
  return tube->GetRmin();
}
```

- Motivation and Goals
- Concepts and Design
- Implementation
- **Ongoing work**
- Summary

Simulation: Ongoing Work for LC

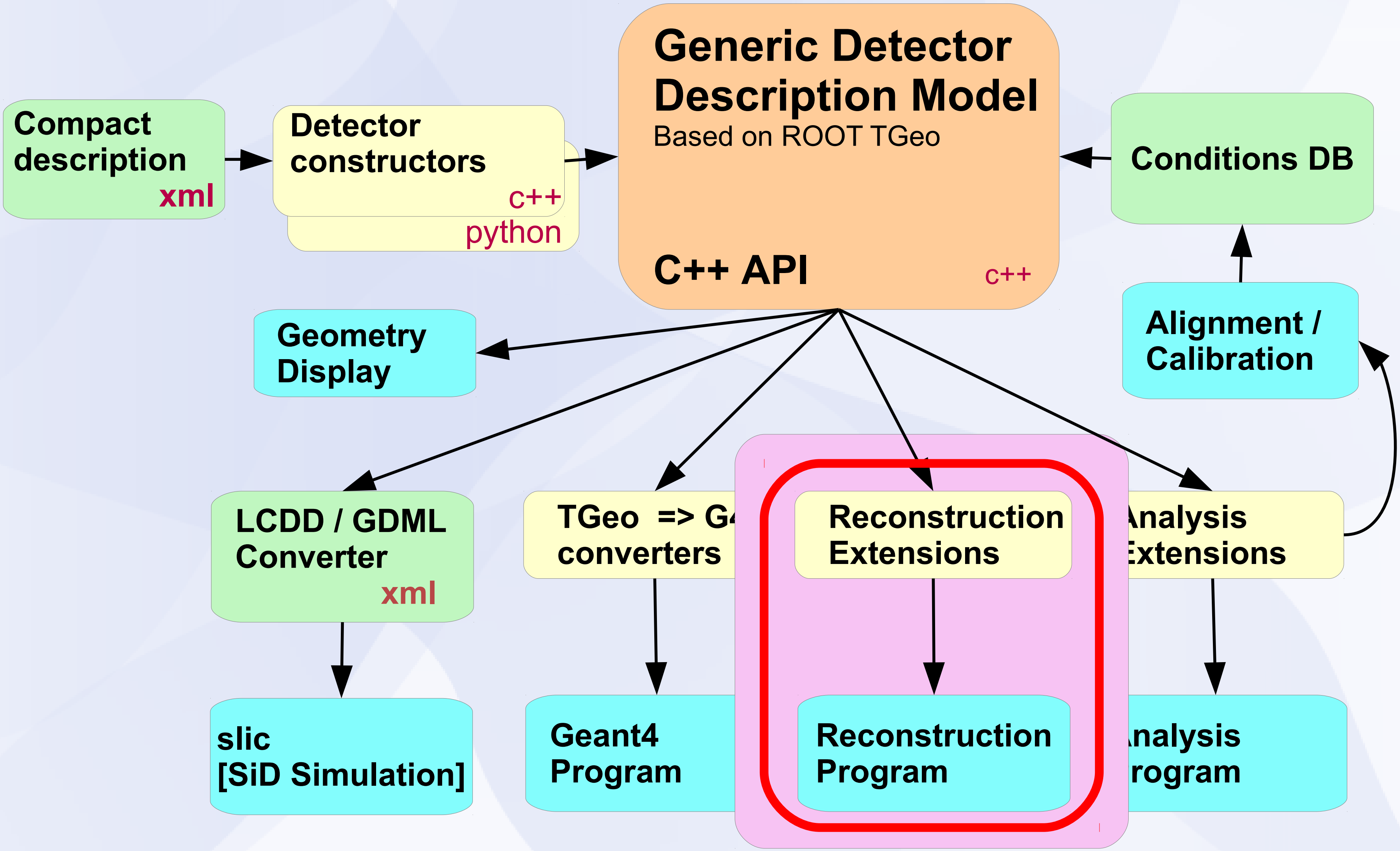


Geant 4 Gateway

- CERN/LCD follow suggestion to benefit from 'slic' (SiD) as simulation framework
  - Convert DD4hep geometry to LCDD notation (xml)
  - Materials, Solids, Limit sets, Regions
  - Logical volumes, Placed volumes / physical volumes
  - Fields
  - Sensitive detector information
- Collaboration with SiD/SLAC (N.Graf, J.McCormick)

F.G. successfully simulated ILD example detector

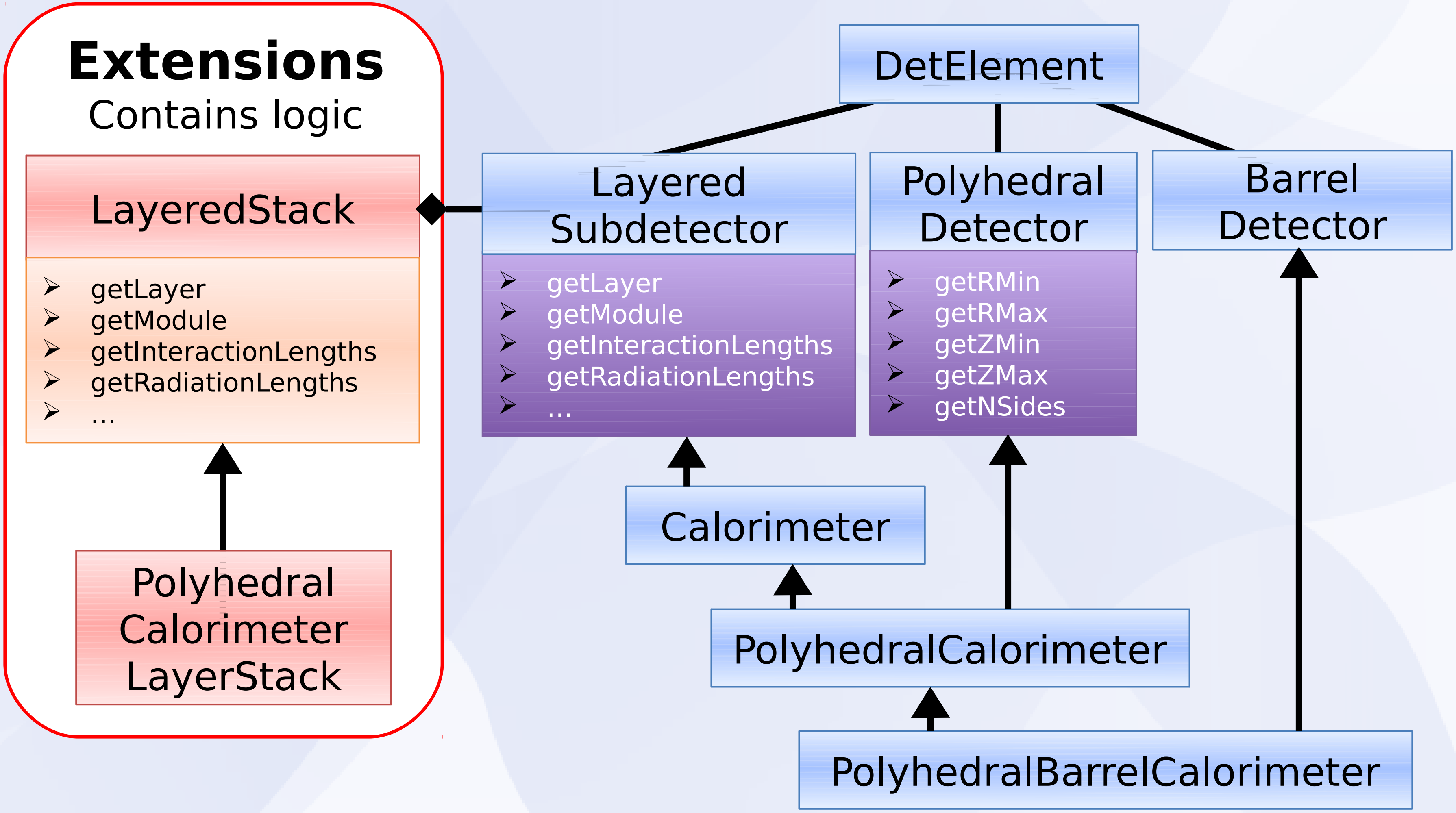
Simulation: Ongoing Work for LC



Calorimeter Reconstruction Interface (C.Grefe)

- **Reconstruction classes extend DetElement**
- **Define high level interface** but contain no logic
- **Specific information stored in extension**
- Extensions have to fulfill specific interfaces, i.e. LayeredSubdetector uses LayerStack
- Add concrete extension in det constructor, i.e. PolyhedralCalorimeterLayerStack
- **Easily extendable** to all subdetectors, should be able to **re-use interfaces**

Calorimeter Reconstruction Interface



Extensions

Contains logic

LayeredStack

- getLayer
- getModule
- getInteractionLengths
- getRadiationLengths
- ...

Polyhedral Calorimeter LayerStack

DetElement

Layered Subdetector

- getLayer
- getModule
- getInteractionLengths
- getRadiationLengths
- ...

Polyhedral Detector

- getRMin
- getRMax
- getZMin
- getZMax
- getNSides

Barrel Detector

Calorimeter

PolyhedralCalorimeter

PolyhedralBarrelCalorimeter

- Motivation and Goals
- Concepts and Design
- Implementation
- **Summary**

Summary

- DD4Hep is a generic tool able support any HEP experiment - not (yet) perfect though
- Supports functionality for the detector design phase
- Work to support simulation and reconstruction for linear collider detectors ongoing
- Functionality which will need to be addressed
  - Functionality arising once experiments get mature
  - Alignment
  - Connection to conditions
  - LHCb showed interest. Are these their topic(s) ?

http://aidasoft.web.cern.ch/DD4hep