DD4hep

An attempt towards a HEP detector description supporting the full experiment life cycle

- **Motivation and Goals**
- Concepts and Design
- Implementation
- Future work – next steps
- 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 source, which supports**
    - **simulation, reconstruction, analysis**
  - **Full description, including**
    - **Geometry, readout, alignment, calibration etc.**

  + standard commercials apply: simple usage etc.

AIDA Project – WP2

- **AIDA: Advanced Infrastructure for Detector development for future Accelerators**
  - **Four year project of the EU Framework 7**
- **WP2: Common Software Tools:**
  - **develop core software tools that are useful for the HEP community at large and in particular for the next big planned projects: sLHC and Linear Collider (ILC/CLIC)**
  - **DD4hep is a subproject of WP2**

AIDA Project - WP2

http://aida.web.cern.ch/aida/activities/networks/software/WP2

Task 2.2: Geometry toolkit for HEP

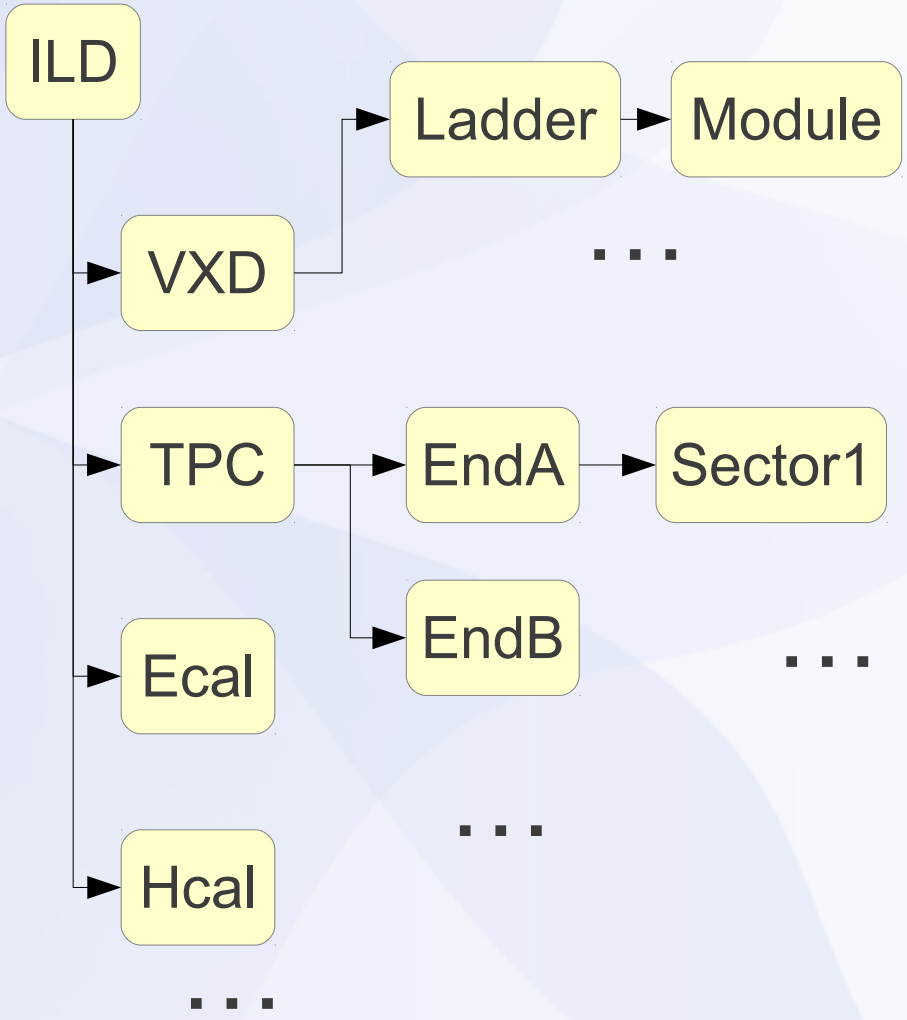
- Allow the description of complex geometrical shapes, materials an sensitive detectors
- Provide interfaces to full simulation programs (Geant4), fast simulations, visualization tools and reconstruction algorithms
- Allow for the misalignment of detector components
- Provide an interface to calibration constants and conditions data

Task 2.3: Reconstruction toolkit for HEP

- Tracking toolkit based on best practice tracking and pattern recognition algorithms
- Provide alignment tools
- Allow for pile up of hadronic events
- Calorimeter reconstruction toolkit for highly granular calorimeters based on Particle Flow algorithms

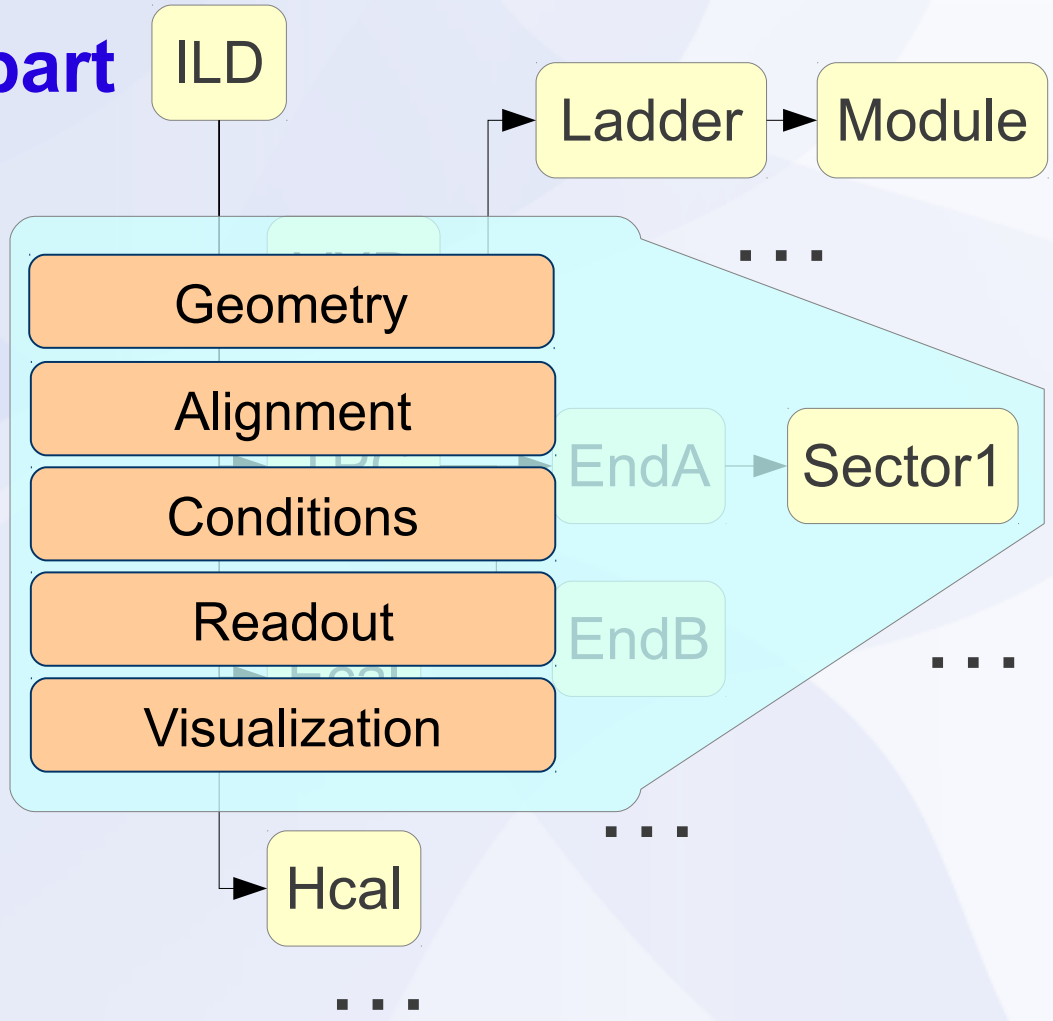
What is Detector Description ?

- **Description of a tree-like hierarchy of "detector elements"**
  - **Subdetectors or parts of subdetectors**
  - **Example:**
    - Experiment
    - TPC
    - Endcap A/B
    - Sector
    ...



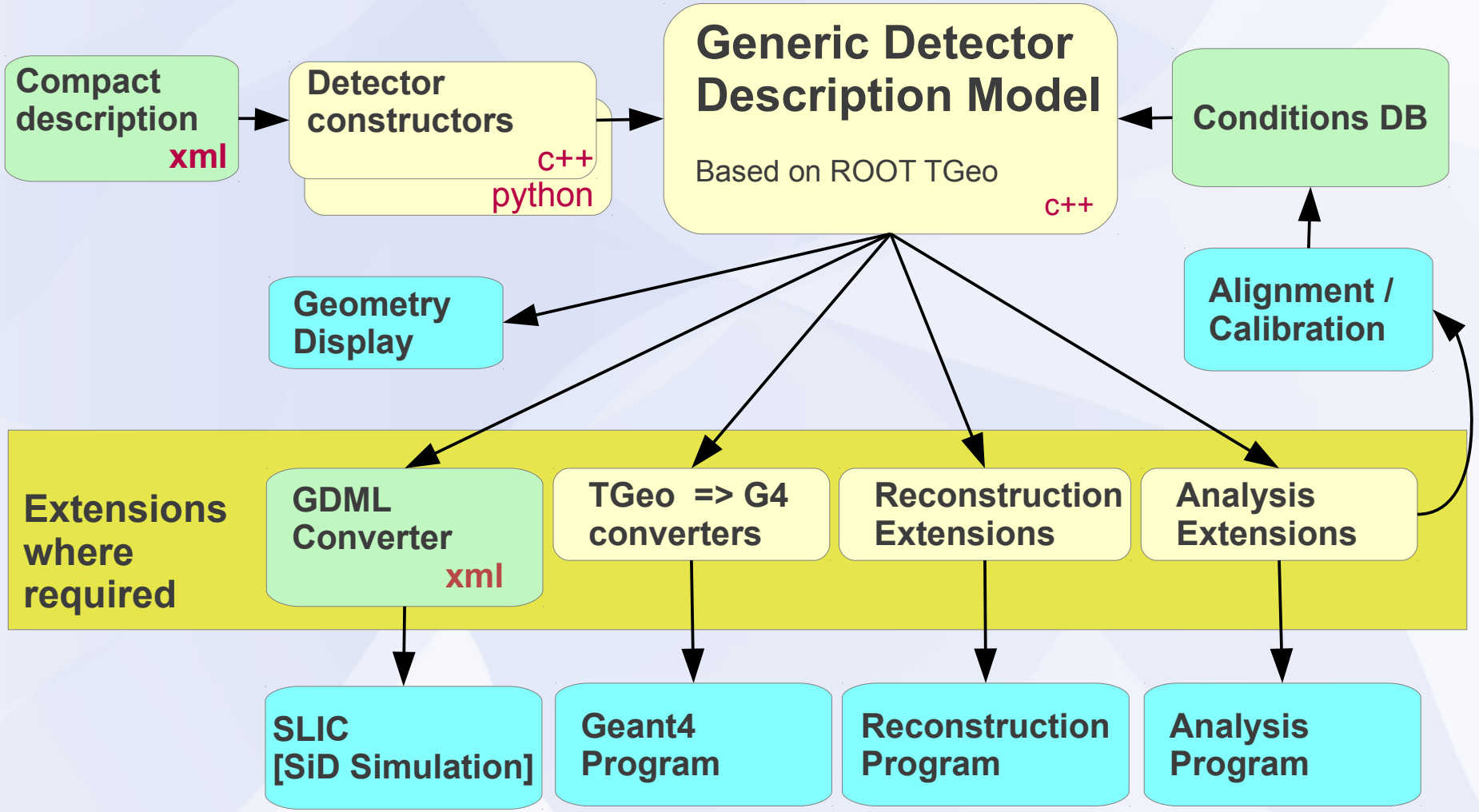
What is a Detector Element ?

- **Subdetector or the part of a subdetector including the description of its state**
  - **Geometry**
  - **Environmental conditons**
  - **Properties required to process event data**



- Motivation and Goals
- **Concepts and Design**
- Implementation
- Future work – next steps
- Summary

DD4Hep - The Big Picture



XML – Compact Description

- **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>
```

DD4Hep – Detector Constructors (C++)

```
static Ref_t create_element(LCDD& lcdd, const xml_h& e, SensitiveDetector& sens)  {
 xml_det_t   x_det  = e;
 string      name   = x_det.nameStr();
 DetElement  sdet(name,x_det.id());
 Assembly    assembly(name);
 xml_comp_t  x_coil = x_det.child(Unicode("coil"));

 Tube   coilTub(x_coil.inner_r(),x_coil.outer_r(),x_coil.zhalf());
 Volume coilVol("coil",coilTub,lcdd.material(x_coil.materialStr()));
 coilVol.setVisAttributes(lcdd.visAttributes(x_det.visStr()));
 assembly.placeVolume(coilVol);

 PlacedVolume pv=lcdd.pickMotherVolume(sdet).placeVolume(assembly);
 sdet.setPlacement(pv);
 return sdet;
}

DECLARE_DETELEMENT(Tesla_coil00,create_element);
```

1) Create Detector Element

2) Create envelope

3) Create volume: *Shape of given Material*

4) Place volume in envelope

5) Place envelope

6) Publish constructor

- **More complex DetectorElements are tree-like**
- **recurse steps 3,4 to describe inner structures**

DD4Hep – Detector Constructors (Python)

(Pere Mato CERN/SFT)

```
from ROOT import DD4hep

def detector_Tesla_coil00(lcdd, det):
 tube     = det.find('tubs')
 material = det.find('material')
 det_elt  = DetElement(det.name, det.id)
 assembly = Assembly(det.name)
 x_coil   = det.find('coil')

 coilTub = Tube(x_coil.inner_r,x_coil.outer_r,x_coil.zha
 coilVol = Volume(det.name+'_coil',coilTub,material)
 coilVol.setVisAttributes(lcdd.visAttributes(det.vis))
 assembkly.placeVolume(coilVol)

 phv = lcdd.worldVolume().placeVolume(assembly
 det_elt.setPlacement(phv)
 return det_elt
```

1) Create Detector Element

2) Create envelope

3) Create volume: *Shape of given Material*

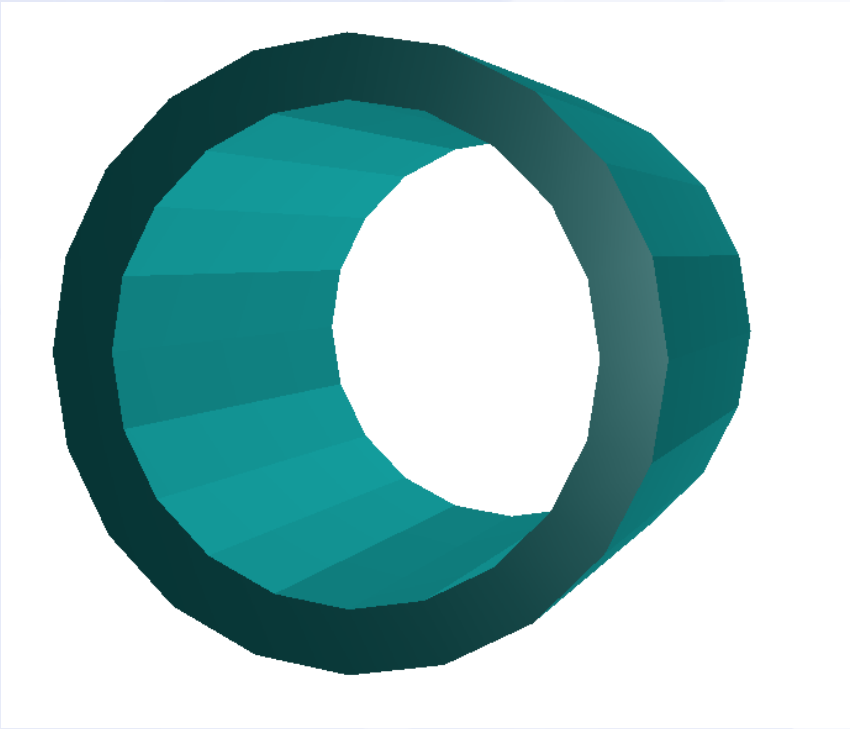
4) Place volume in envelope

5) Place envelope

- **Roughly identical code**

And what you obtain:

- **Develop Display application using native ROOT (OpenGL, Eve, etc.)**

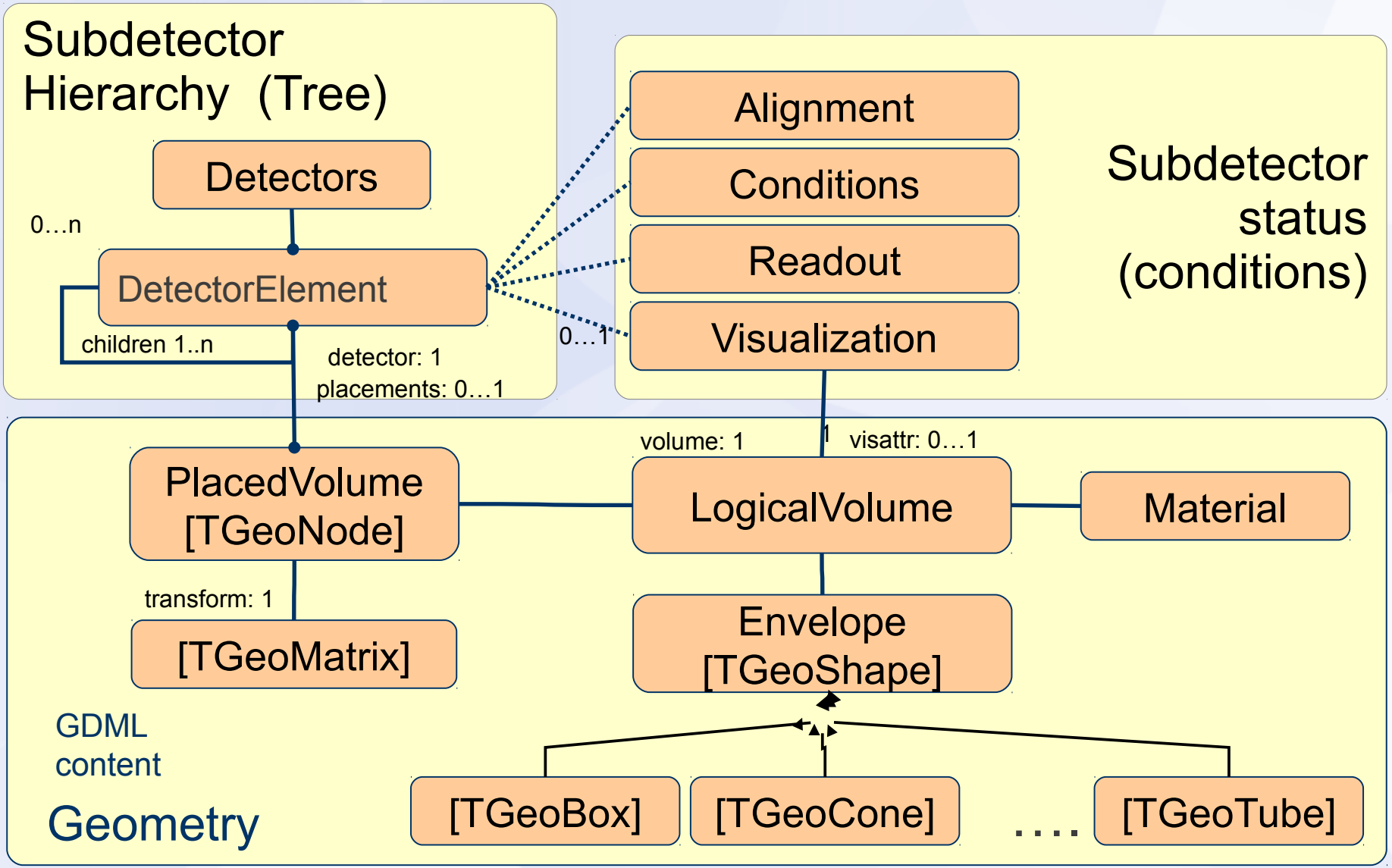


- Motivation and Goals
- Concepts and Design
- **Implementation**
- Future work – next steps
- Summary

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



Implementation: Client Extensions

- **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 TPC hit to experiment coordinates => specialized data required (cache of precomputed results)**

Implementation: Client Extensions

- Functionality achieved by creating 'views'
  - Data describing a detector element are public
    - User objects may be attached to data
  - Possibility of many views based on the same data
    - All views share the same data
    - All views have a consistent state
  - Views are 'handles' to the data of the detector element
    - Creation of a view is efficient in memory and CPU

Implementation: 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

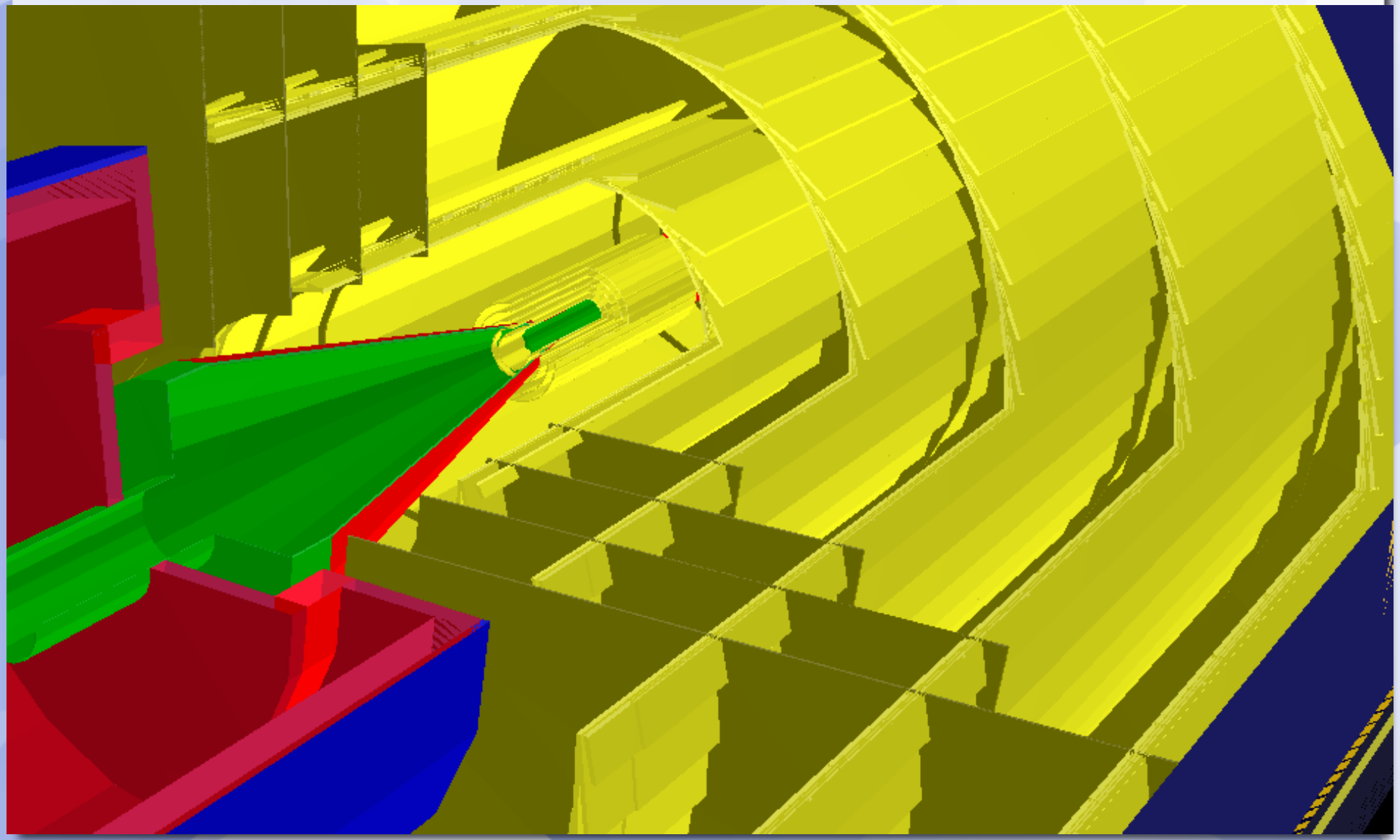
```
struct ILDExTPC : public DetElement {
  ....
  void getDriftVelocity()  const;
  void getInnerRadius()  const;
  void getOuterRadius()  const;
  ....
};
```

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

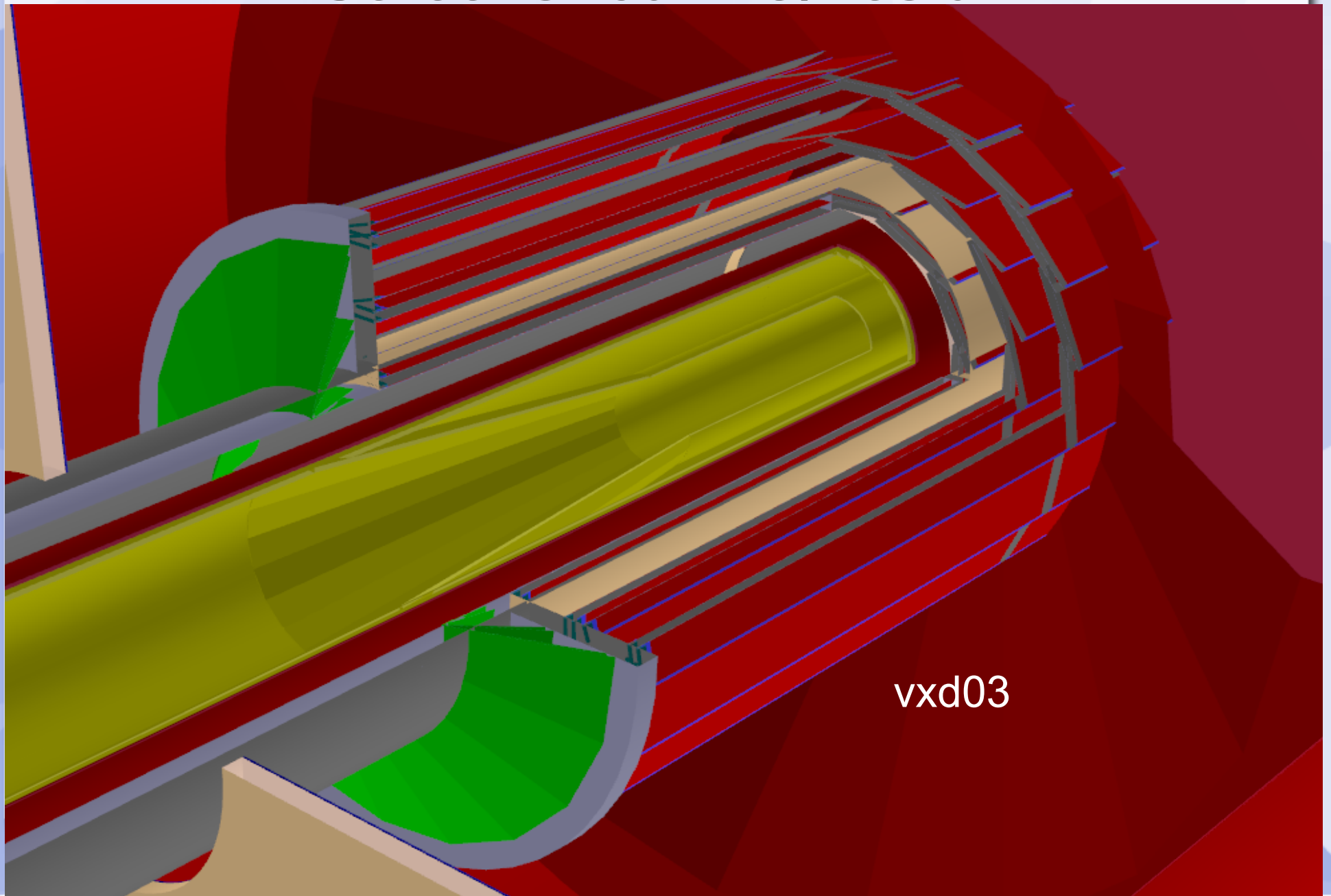
Some Screen Shots

...Please lean back...

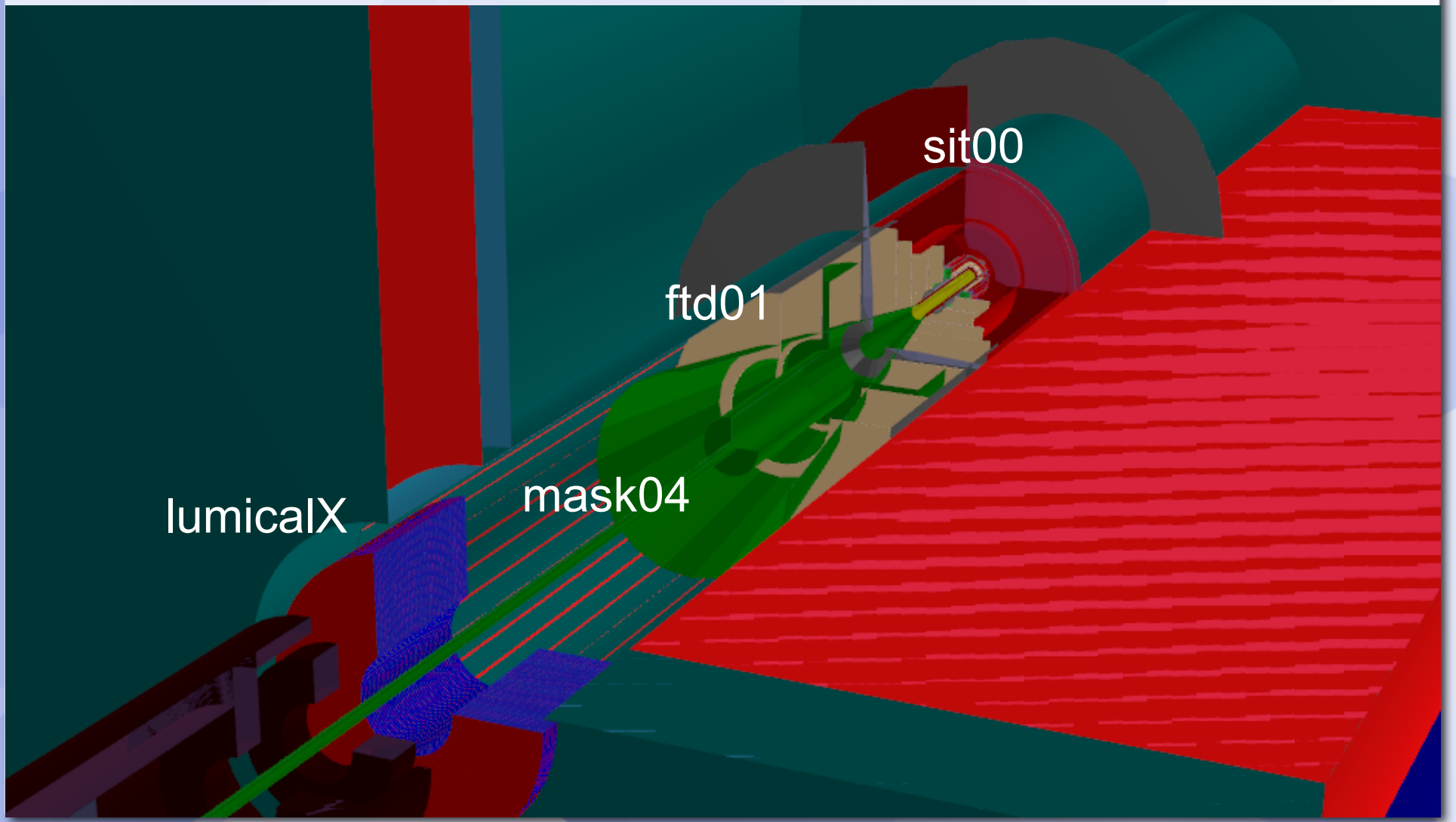
Screenshot: SiD



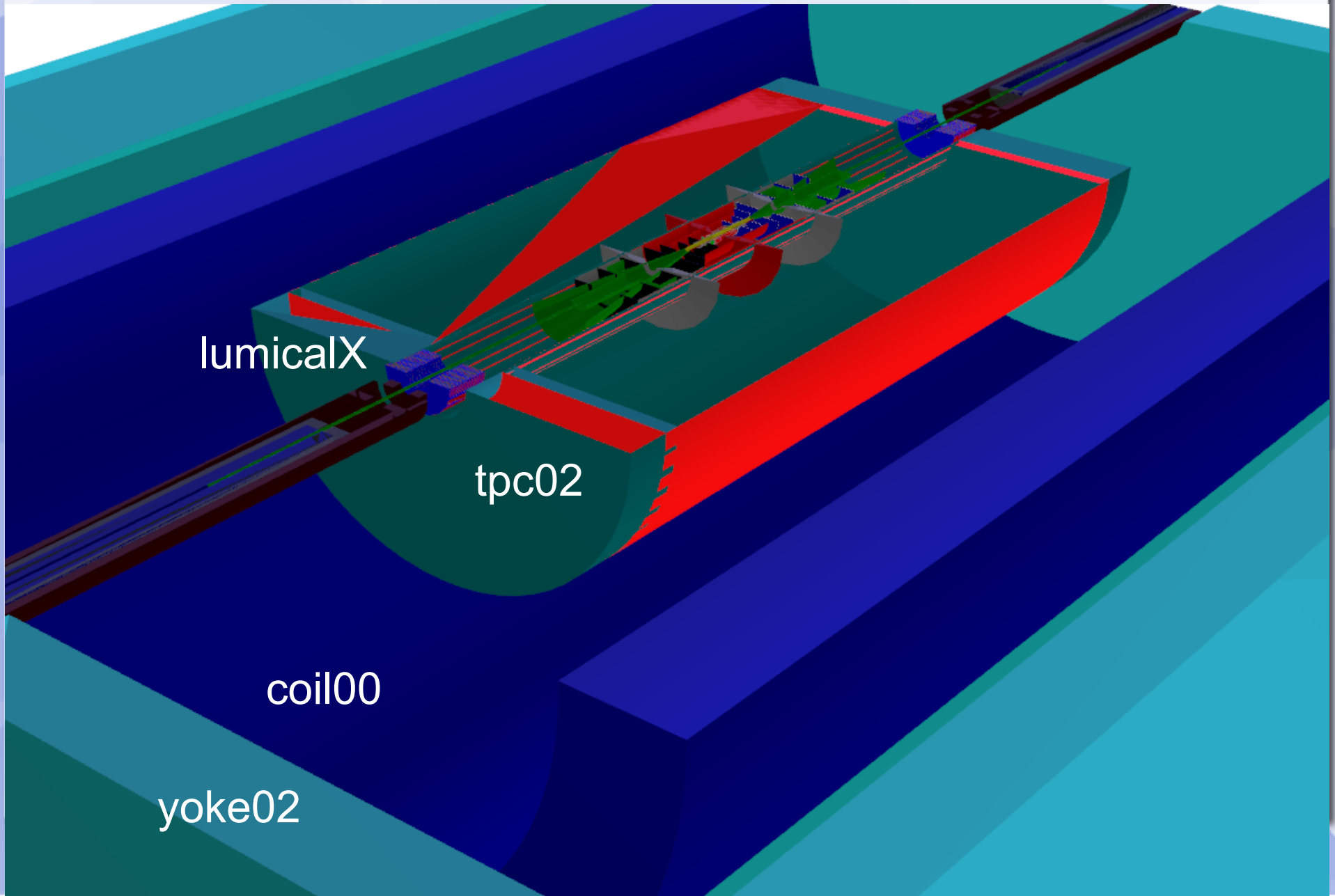
Screenshot: ILC/Tesla



Screenshot: ILC/Tesla

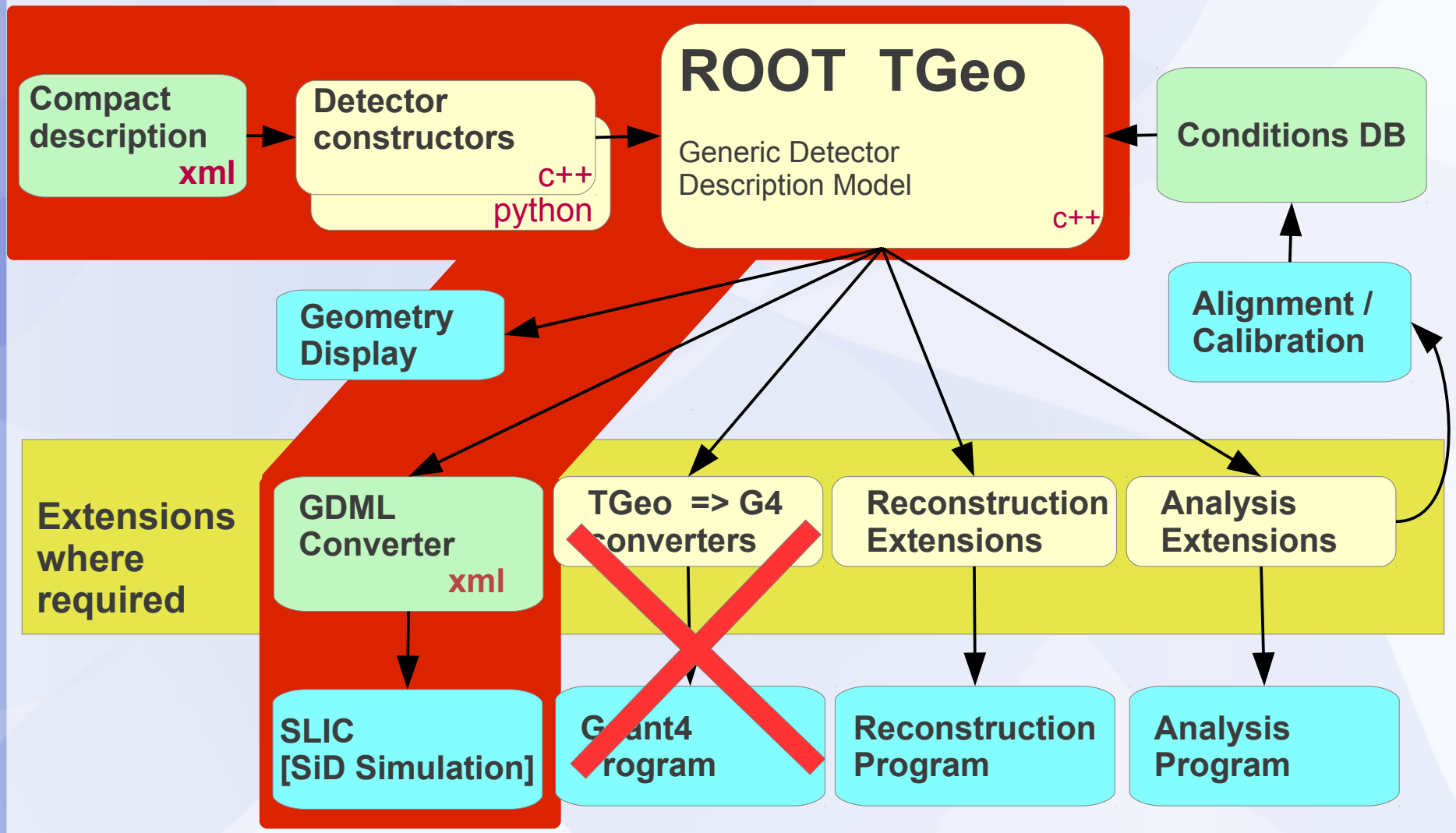


Screenshot: ILC/Tesla



- Motivation and Goals
- Concepts and Design
- Implementation
- **Future work – next steps**
- Summary

Next Step: Translate one Mokka model and feed simulation



Work Involved

- Define model dependent compact description for the subdetectors
  - Possibly generate from Mokka database [??]
- Develop the corresponding detector constructors
- Save detector model to intermediate GDML file
- Hope: slic (Generic [SiD] simulation) is sufficiently flexible to cope with the detector model
  - Provided the detector constructor create a detector description conform to slic expectations/requirements

- Motivation and Goals
- Concepts and Design
- Implementation
- Future work – next steps
- **Summary**

Summary

- **Common detector geometry description tool is one of the keys towards a common software base for a future LC**
- **DD4Hep developed in the context of the AIDA project is such generic tool**
- **Required functionality for the detector design phase (geometry) is present**
- **Aiming for first release soon**
- **Apply geometry model to existing LC detector design for simulation**

Web:	http://aidasoft.web.cern.ch/DD4hep
Code view:	http://svnsrv.desy.de/viewvc/aidasoft/DD4hep
Code access:	svn co https://svnsrv.desy.de/desy/aidasoft/DD4hep/trunk