DD4hep

HEP detector description supporting the full experiment life cycle

M.Frank[1], F.Gaede[2], P.Mato[1]

[1]CERN, 1211 Geneva 23, Switzerland
[2]Desy, 22607 Hamburg, Germany

- **Motivation and Goals**
- Concepts and Design
- Implementation
- Simulation
- 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.

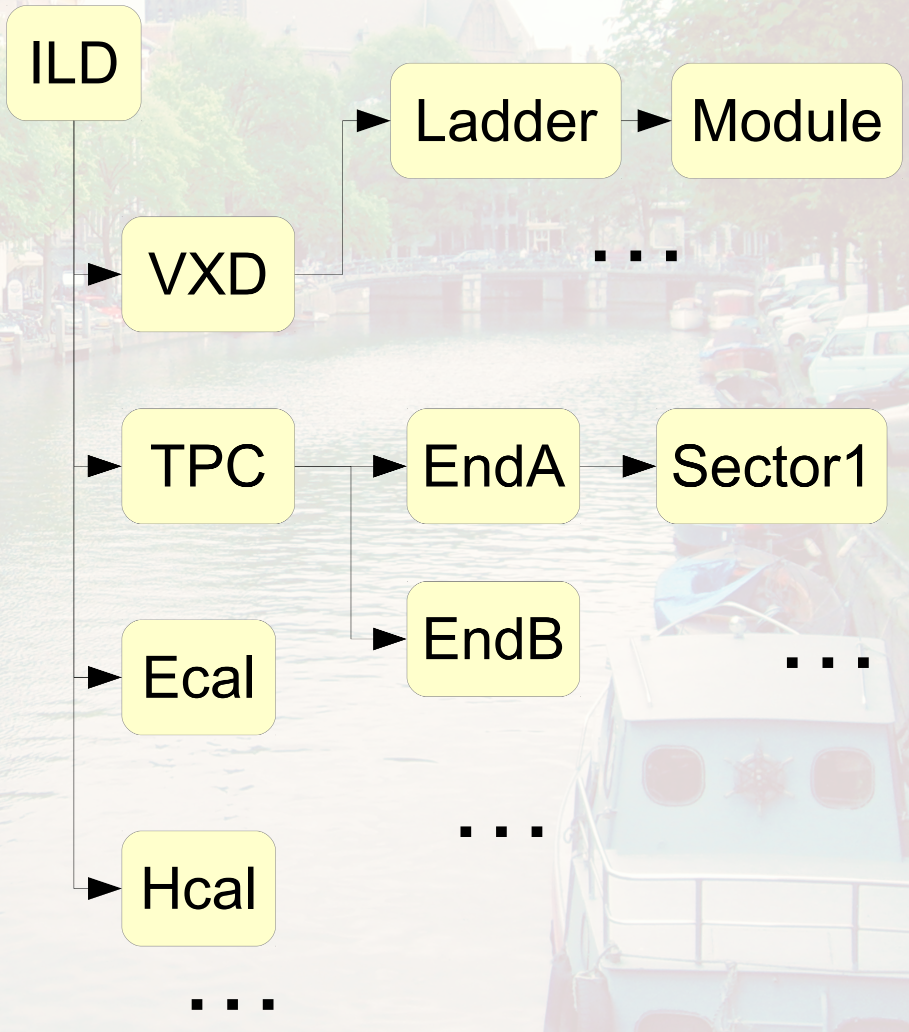
    + standard commercials apply: simple usage etc.
  - Part of AIDA project: WP2 Common Software Tools
    (EU FP7 Research Infrastructures, grant no 262025)

What is Detector Description ?

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

    Example (ILD):
    - Experiment
      - TPC
        - Endcap A/B
          - Sector

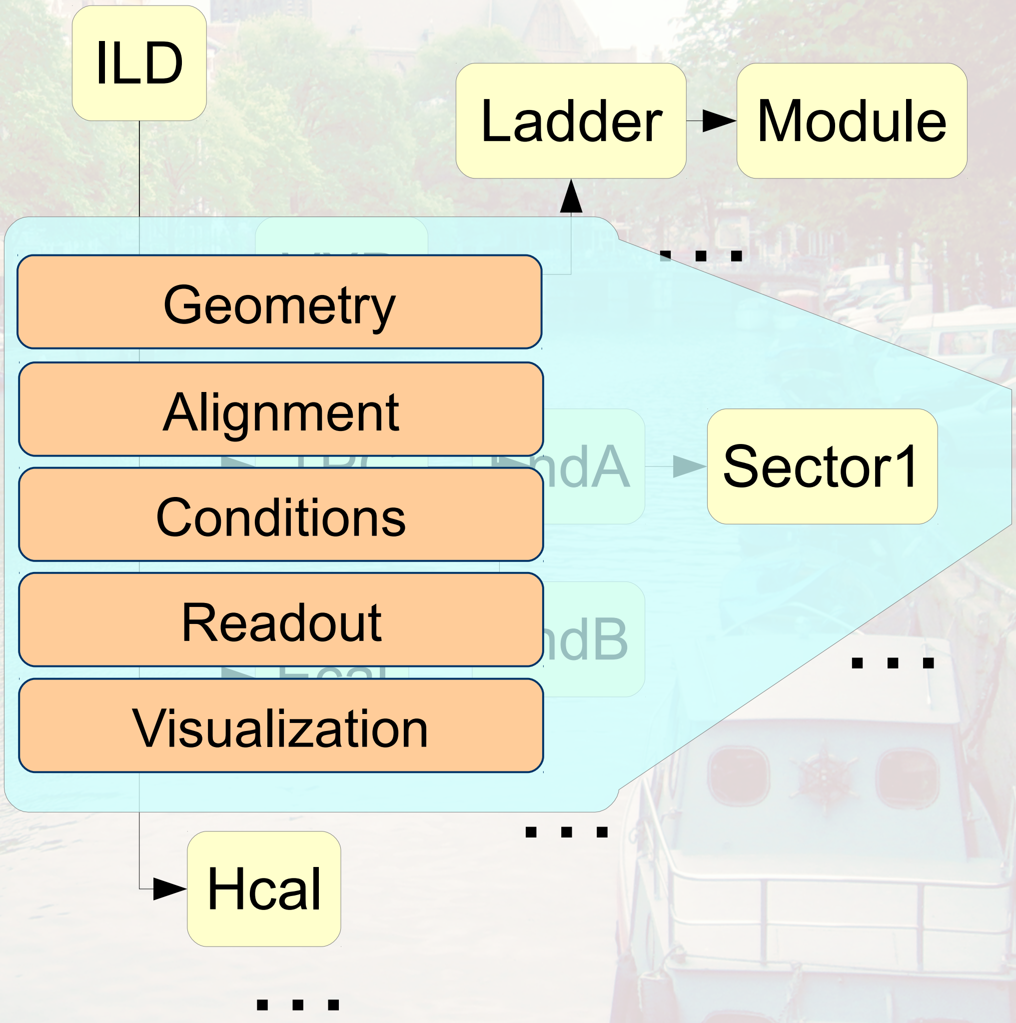
      ...



What is a Detector Element ?

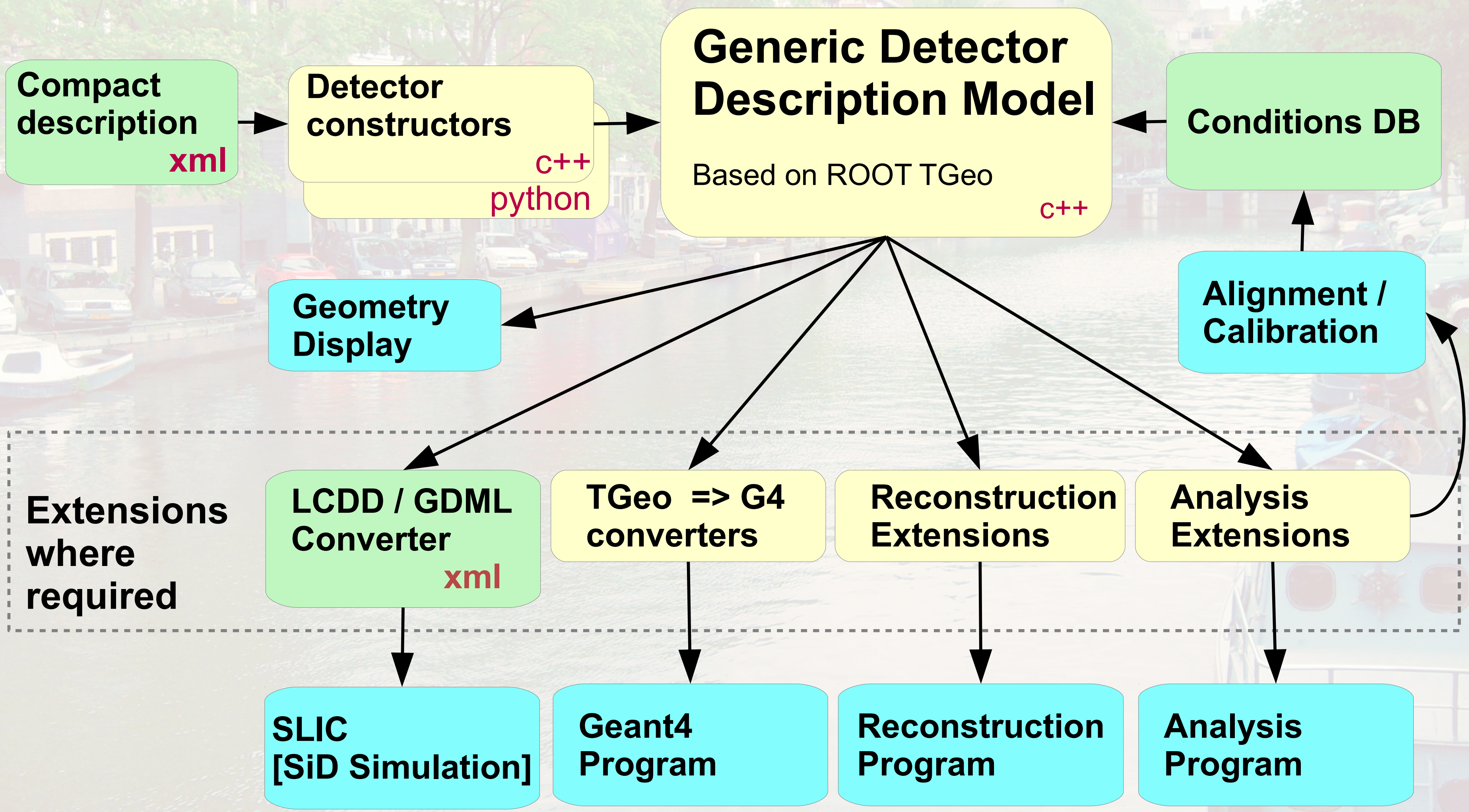
Describes a significant piece of a sub-detector including its state

- **Geometry**
- **Environmental conditions**
- **All properties required to fully interpret data from particle collisions**



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

DD4Hep - The Big Picture



Compact Description – XML

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

```xml
<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'

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

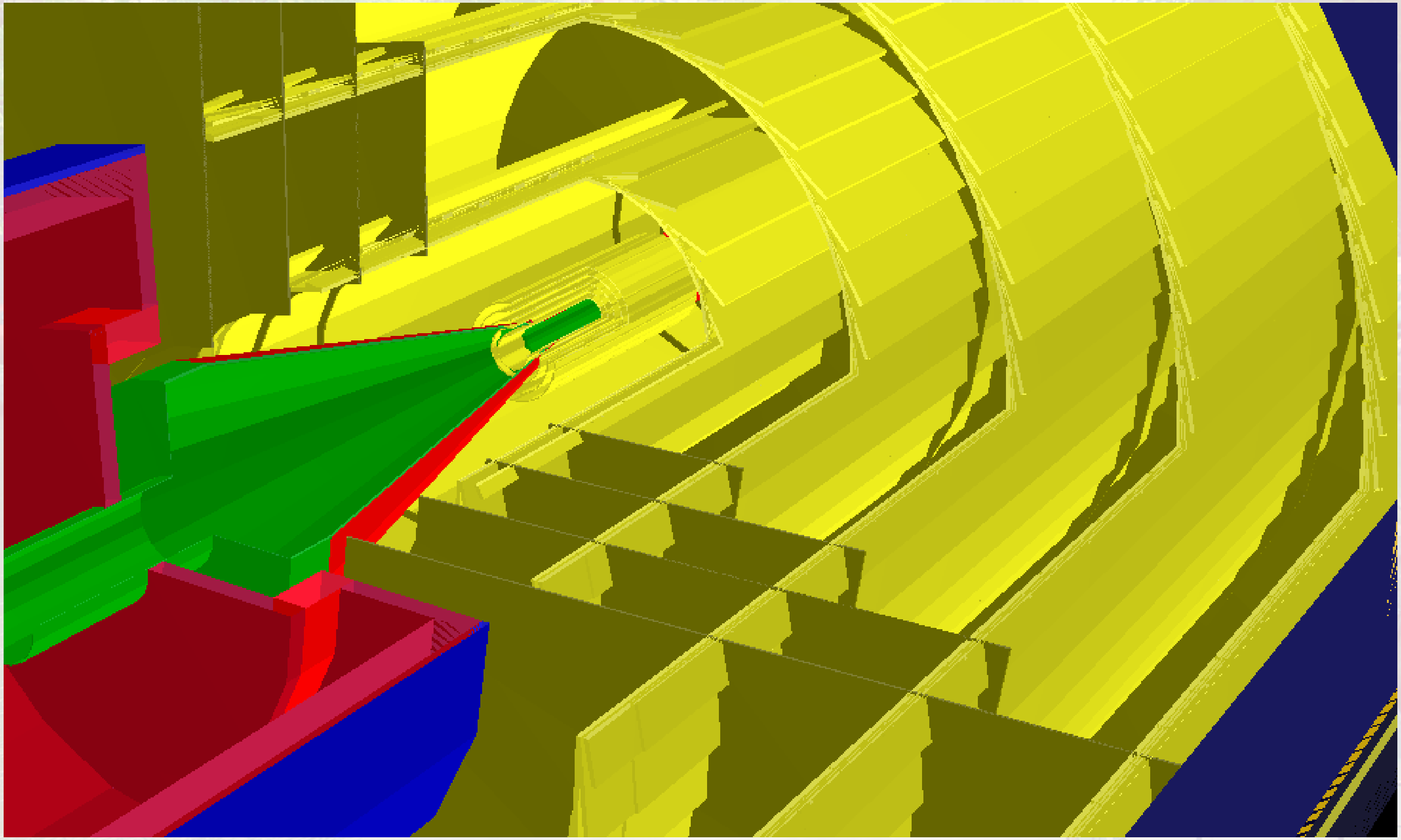
6) Publish constructor

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

Display options

Display using native ROOT

OpenGL, Eve, etc.

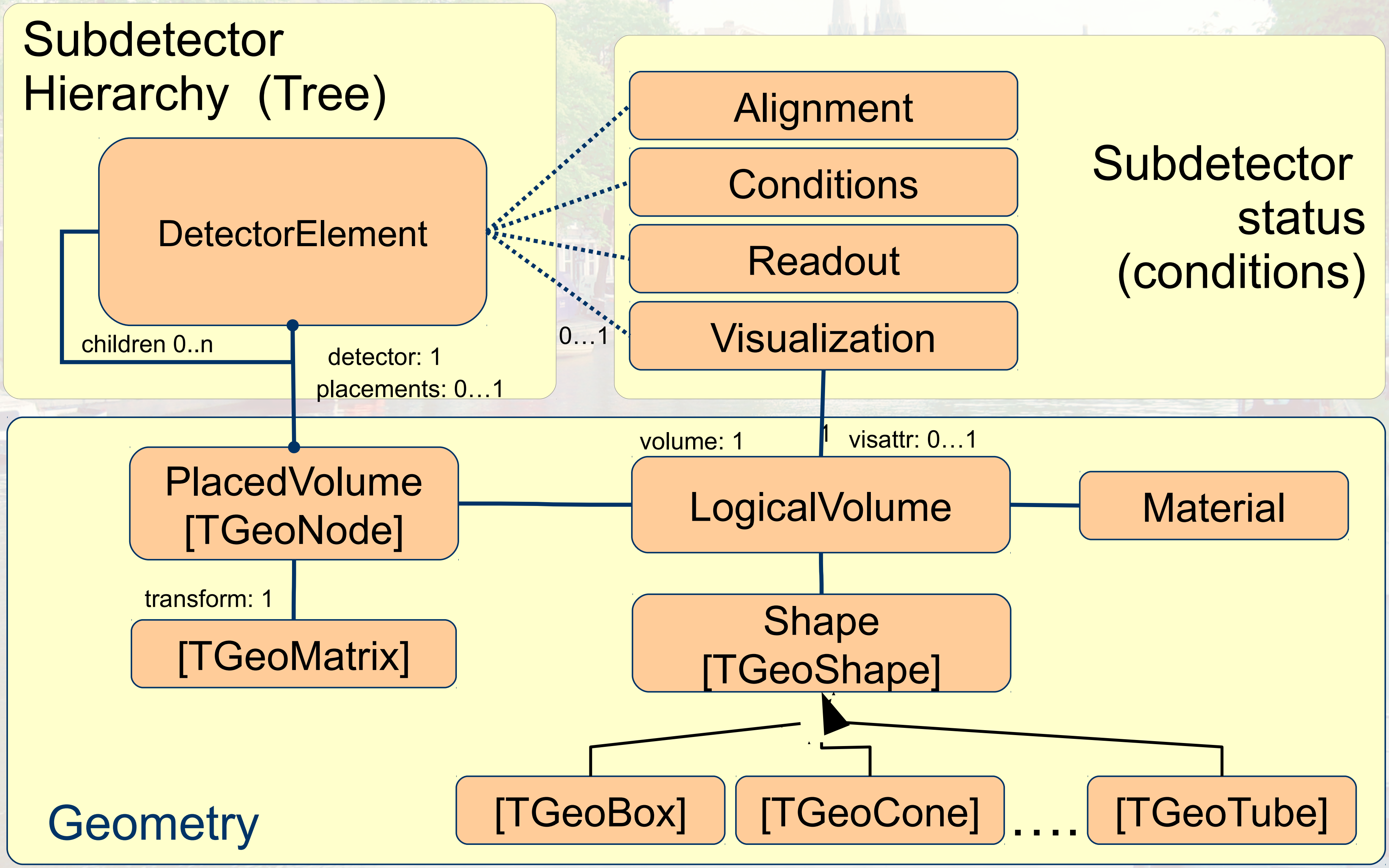


- Motivation and Goals
- Concepts and Design
- **Implementation**
- Simulation
- Summary

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

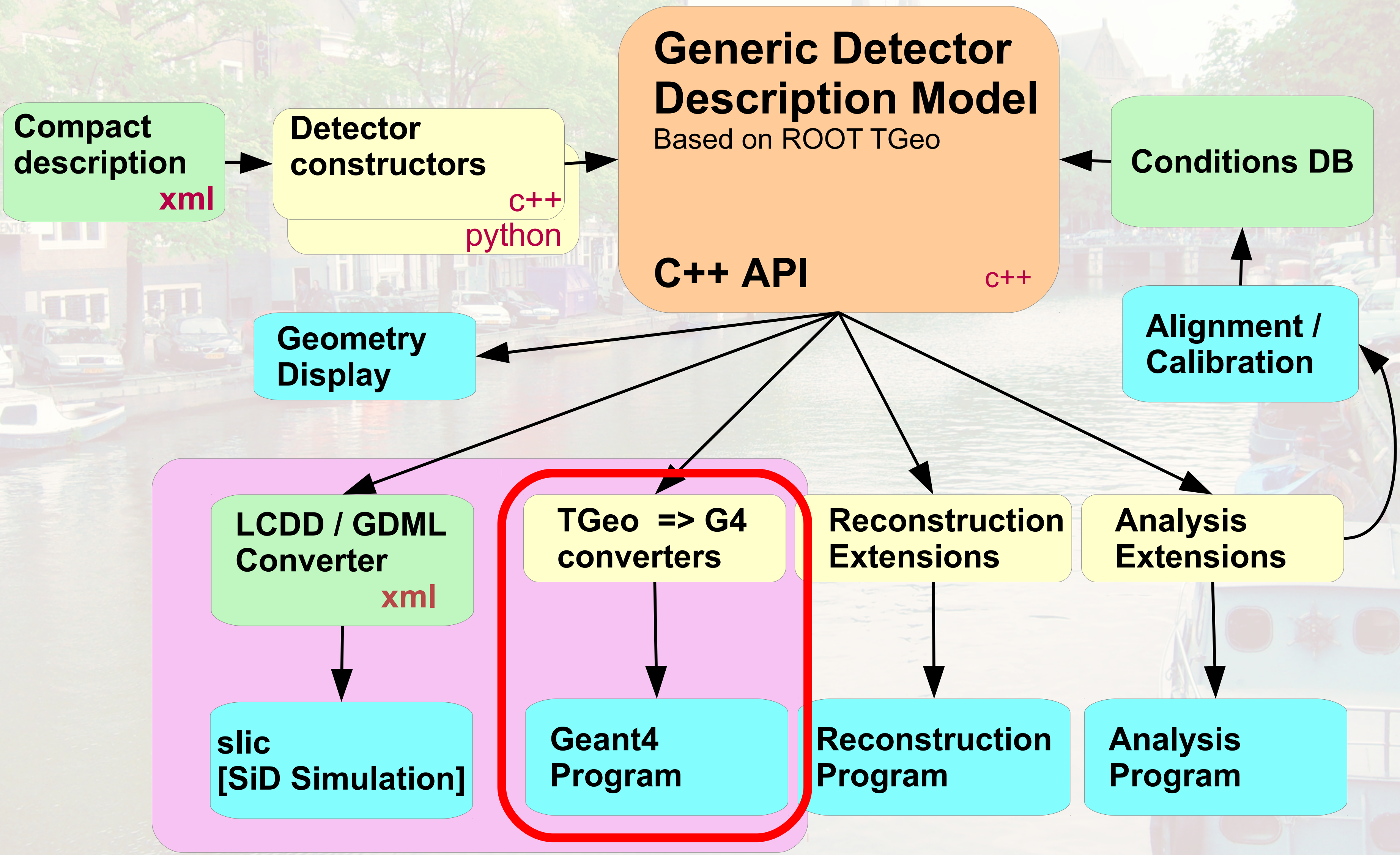


Deal with the Unforeseeable

- Use case: The use of the geometry is different in track reconstruction and alignment
  => specialized 'behavior' required
- Object functionality is achieved by 'views' of public data describing a detector element
  - Consequence of separation of 'data' and 'behavior'
  - Many different views share the same data
    - Support specialized behavior
    - User objects may be attached to data (extensions)
    - Views are 'handles': Creation is efficient and fast
  - View for Convenience, Compatibility and Optimization

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

Simulation: Ongoing Work for LC



Geant 4 Gateway: In Memory

- Basic Idea:
  - walk through the geometry starting from “world”
  - convert the geometry from ROOT to Geant4
  - all runs by magic
- Convert full geometry while walking from TGeo automatically to Geant4
  - Materials, Solids, Limit sets, Regions
  - Logical volumes, placed volumes / physical volumes
  - Fields
  - Sensitive detectors

In Memory Translation to Geant 4

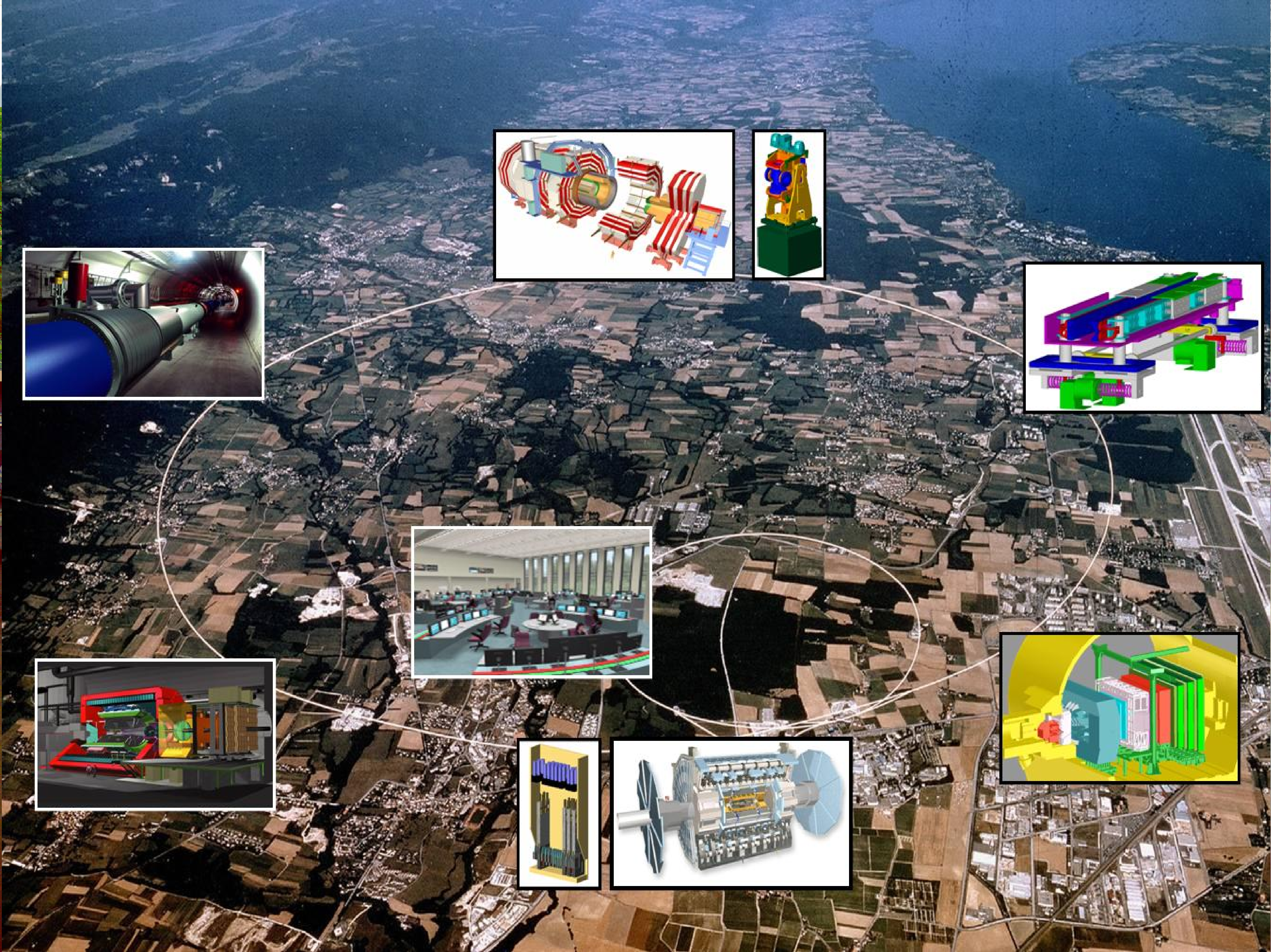
- This processing chain was implemented
- Geant4 Sensitive Detectors are not provided
  - Couples detector 'construction' to reconstruction, MC truth and Hit production
  - Highly dependent on technology of the sensitive elements
  - Palette of most common sensitive components will be provided – but cannot serve specific needs
- Ongoing work
  - Development for 'generic' framework
  - Investigations to 'reuse' existing SD (LC community)

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

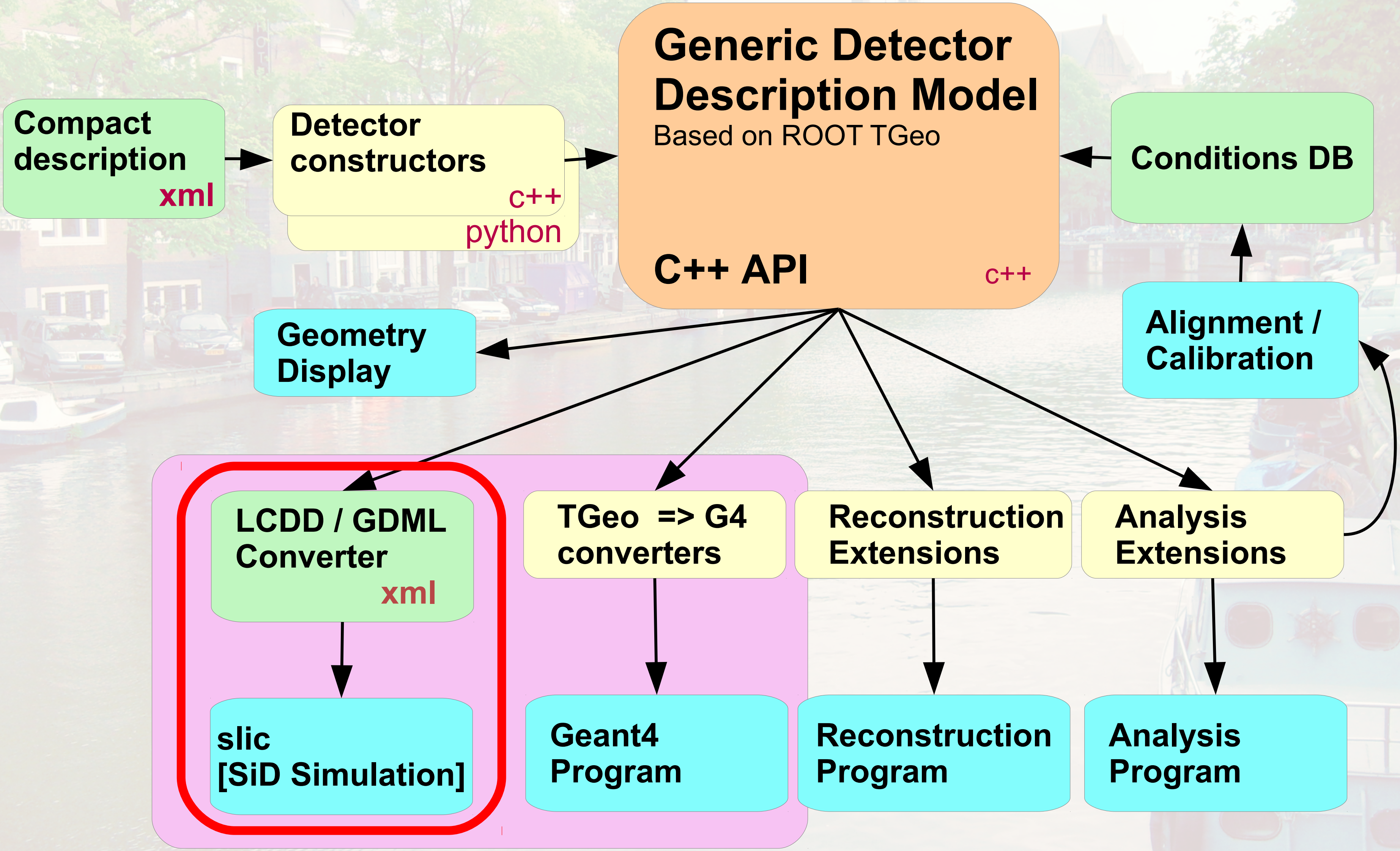
Summary

- 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
- Missing functionality to be addressed
  - needed for mature experiments
  - Alignment
  - Connection to conditions

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



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)
- Model is working