DD4hep Status

The attempt towards a HEP detector description supporting the full experiment life cycle
• **Motivation and Goals**

  => Introduction

• **Concepts and Design**

• **Going to the 'real world'**

• **Summary**
Motivation and Goal

- Develop a detector description
  - For the full experiment life cycle
    - detector concept development, optimization
    - detector construction and operation
    - Easy transition from one phase to the next
    - “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.
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
    ...

Diagram:
- ILD
  - Ladder
  - Module
  - VXD
    - TPC
      - EndA
        - Sector1
        ...
    - Ecal
        ...
    - Hcal
        ...
  ...

What is a Detector Element?

- Subdetector or the part of a subdetector including the description of its state
  - Geometry
  - Environmental conditions
  - Properties required to process event data
• Motivation and Goals

• Concepts and Design

  => Focus on recent developments

• Going to the 'real world'

• Summary
Reminder: DD4Hep - The Big Picture

- **Compact description** in XML
- **Detector constructors** in Python
- **Generic Detector Description Model** based on ROOT TGeo
- **Conditions DB**

**Extensions where required**

- **LCDD / GDML Converter** in XML
- **TGeo => G4 converters**
- **Reconstruction Extensions**
- **Analysis Extensions**

**Geometry Display**

- **Alignment / Calibration**

**Reconstruction**

- **Program**
- **Analysis Program**

**Analysis**

- **Program**
- **Reconstruction Program**

**slic**

- **[SiD Simulation]**

- **Geant4 Program**
Implementation: Design Choices

- **Detectors are described by a compact notation**
  - Inspired by SiD compact description [Jeremy McCormick]
  - Flexible and extensible

- **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
Follow-up: Geometry Implementation

- Based on ROOT TGeo
- No insufficiencies found: model seems correct
Shapes and Solids: Enhanced Palette

- TGeo shapes used internally. Palette ~complete
- Commitment of TGeo to use USolids
Dealing with the 'Unforeseeable'

- **Compact description** (xml)
- **Detector constructors** (c++)
- **Geometry Display**
- **LCDD / GDML Converter** (xml)
- **TGeo => G4 converters**
- **Reconstruction Extensions**
- **Analysis Extensions**

**Extensions where required**

- **slic [SiD Simulation]**
- **Geant4 Program**
- **Reconstruction Program**
- **Analysis Program**

**Generic Detector Description Model**
- Based on ROOT TGeo
- **C++ API**

**Conditions DB**

**Alignment / Calibration**
Client Extensions

- Provide flexible functionality to solve reconstruction and analysis problems
- Approach to deal with the “unforeseeable”
- Motivated by the fact that Different use cases require different functionality
  - Example: Optimization of coordinate transformations local TPC hit to experiment coordinates => specialized data required (cache of precomputed results)
  - Need to extend the detector element's data
Implementation: Client Extensions

- **Functionality achieved by 'views'**
  - Corollary of the design choice to separate ‘data’ from ‘behavior’
  - Possibility of many views based on the same data
    - All views share the same data __OR__
    - Same ‘data’ can be associated to different ‘behaviors’
    - All views are consistent
  - Public data describing a detector
    - User objects may be attached to data
  - Views are 'handles' to the data
    - Creating views is efficient and fast
    - Typically only a pointer needs to be copied
Client Extensions

- **Default DetElement properties and data**

- **Added subdetector specific data**
Example: TPC (A. Muennich)

- Customize a DetElement object to support TPC specific questions and cached data
  - Data cache: save CPU using precomputed results
  - Facilitate TPC specific interface to clients
- Which is the mechanism behind?
- How to implement such extensions?
TPC – Detector Constructor

```
DetElement module(part_det,m_nam,mdcount);
PadLayout* pl = new RectangularPadRowLayout(module);
module.addExtension<PadLayout>(pl);
```

- **Detector element to extend**
- **Extension object**
- **Public type of the extension object**
  (May be ABC or interface like here)

- Any number of extensions
  - Must differ by public type
- Adding an extension is possible anywhere
  - Extensions are not confined to detector constructor
  - Could also be somewhere in the reconstruction code
TPC Module View

```cpp
TPCModule(const Geometry::DetElement & e) : Geometry::DetElement(e), padLayout(0)
{
    getExtension();
}

void TPCModule::getExtension()
{
    padLayout = isValid() ? extension<PadLayout>() : 0;
}
```

- The PadLayout is retrieved from the detector element if present
  - Lookup relatively cheap, but not for free
  - Hence: extension pointer is cached
  - Map lookup by type_info
• Motivation and Goals

• Concepts and Design

• Going to the 'real world'

  => Out of 'lab conditions' towards clients

• Summary
End of Playing: Getting Mature

- Identified first 'massive' client
  - Linear Collider Detector community (ILD)
- Simulation framework (Mokka) at end of life
  - Replacement required for future detector studies
- Small study group established to verify feasibility
  - M.Frank, C.Graefe, A.Sailer, J.Strubbe (CERN/LCD)
- Additional complication: 2 frameworks
  - ILD: Mokka + Marlin
  - SiD: slic + java based reconstruction/analysis
DD4Hep - The Big Picture

Generic Detector Description Model
Based on ROOT TGeo

Compact description
xml

Detector constructors
python

C++ API

Geometry Display

Conditions DB

Alignment / Calibration

LCDD / GDML Converter
xml

TGeo => G4 converters

Reconstruction Extensions

Analysis Extensions

slic [SiD Simulation]

Geant4 Program

Reconstruction Program

Analysis Program

Reconstruction Program

Analysis Extensions
Geant 4 Gateway

• **Idea:**
  - walk through the geometry starting from “world”
  - convert the geometry from ROOT to Geant4
  - all runs by magic

• **Geometry is automatically converted 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
- Unfortunately the approach was a little bit naïve
  - Requires additional development of sensitive detectors
  - Couples detector 'construction' to reconstruction, MC truth and Hit production

- For ILD it was decided to benefit from 'slic' developments as simulation framework
  - Convert DD4hep geometry to LCDD notation (xml)
Next Step: Translate one Mokka model and feed simulation

- Compact description: `xml`
- Detector constructors: `c++`, `python`
- Geometry Display
- Extensions where required
  - LCDD / GDML Converter: `xml`
  - slic [SiD Simulation]
  - ROOT TGeo: Generic Detector Description Model: `c++`
- Conditions DB
- Alignment / Calibration
- Reconstruction Extensions
- Analysis Extensions
- Geant4 Program
- Reconstruction Program
- Analysis Program
- `slic [SiD Simulation]`
Using the SiD Simulation with DD4hep

• The SiD simulation application 'slic' solves these issues with bravura
  – Collaboration with slic developers started at LCD software workshop in February (N.Graf et al.)

• Goal: Allow to flexibly attach Geant4 sensitive detectors to slic (plugin like mechanism)

• Requires a detector description in 'lcdd' format (XML) with some 'gdml' section
Slic and DD4hep: Status

- Sensitive detector work is ongoing
- Geometry converter was build and is part of the repository
  - Technique to generate such a file is similar to 'in-memory' conversion
  - Slic 'understands' the generated lcdd file (proof of concept)
- Formally the slic engine and the Geant4 event simulation is functional
  - If only existing sensitive detectors are required
Summary

- The DD4hep core was consolidated
- A extensible way to support flexibility is in place
- A functional path to event simulation is present
- Start to face 'real-world-conditions'
  - LCD as major client
  - Start to establish a common toolkit for simulation and reconstruction for linear collider detector studies
Questions and Answers