DD4hep

Saga in Five Episodes

DD4hep work plan for AIDA 2020
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, single source of information, which supports**
    • simulation, reconstruction, analysis
  – **Full description, including**
    • Geometry, readout, alignment, calibration etc.
What is Detector Description?

- Description of a tree-like hierarchy of 'detector elements'
  - Subdetectors or parts of subdetectors

- Detector Element describes
  - Geometry
  - Environmental conditions
  - Properties required to process event data
  - Optionally: experiment, sub-detector or activity specific data

Diagram:

- ILD
- Ladder → Module
- VXD
- TPC
- EndA → Sector1
- Ecal
- Hcal
- geometry
- alignment
- conditions
- readout
- visualization
- subdet. data
- reconstr. data
DD4Hep - The Big Picture

Note:
One way to populate DD4hep (plugin based)
Not the exclusive way.

Generic Detector Description Model
Based on ROOT TGeo

Extensions where required
GDML Converter
TGeo => G4 converters
Reconstruction Extensions
Analysis Extensions

Geometry Display

Alignment / Calibration

Conditions DB

Compact description
Detector constructors

Geometry Display

Extensions where required
GDML Converter
TGeo => G4 converters
Reconstruction Extensions
Analysis Extensions

Geant4 Program
Reconstruction Program
Analysis Program
Saga in 5 Episodes: Sub-packages

- Detector description usage
- DD4hep – basics/core
  - Stable, bug-fixes, enhancements
- DDG4 – Simulation using Geant4
  - Validation and enhancement
- DDRec – Reconstruction supp.
  - Not covered here
  - Driven by LC community
- DDAlign – Alignment support
  - In work, needs DDCond
- DDCond – Detector conditions
  - Basically non-existant
DD4hep Core and Multi-.Threadinging

- Multi-threading during initialization
  - Limited pressure. Speed issues may be solved by loading initialized persistent images
- While processing events
  - Normally detector description data are read-only
  - Multi-threading by construction
- Problematic are updates: Conditions, Alignment
  - Multi-threading often uses event parallelism
  - On update the event pipeline MUST be drained to avoid inconsistent detector views
  - This can only be steered by the processing framework
    [lesson learned from Gaudi-Hive]
Simulation: Generic Geant4 Gateway

- **Simulation** = Geometry + Detector response + Physics
- **Attempt for formalization of Geant4**
  - Ideally: configuration without extra (C++) user code
- **DDG4**
  - Bootstrap Geant4 from DD4hep in memory geometry
  - Configure using XML, python or Cint (ROOT 5)
  - Configure Geant4 actions, physics-list, processes, particle constructors, sensitive actions, I/O etc using module palette
Simulation: DDG4

- **Concept**
  - Walk through the geometry and convert on the fly from ROOT to Geant4
  - Instantiate sensitive detectors from palette
  - Instantiate physics list, -constructors and -processes
  - Start simulating

- **Palette of sensitive detectors**
  - Basic concepts implemented

- **Processing chain is implemented**
  - Validation in progress – time consuming process
DDG4 Work List: Keep The Boys Off the Street

- **Support for fast and parametrized simulation**
  - Speed-up by avoiding full Geant4 machinery
  - Workshop @ CERN this autumn

- **Heterogenous simulation**
  - Full, fast and parametrized simulation depending on sensitive region

- **Multi-threading support**
  - According to Geant4 rules
  - Multiple instances of elements handling actions during energy deposits while tracking
DDAlign: Alignment and Detector Conditions

- Fundamental functionality to interpret event data in the real world
  - Selling argument for existing experiments
  - Necessity to handle imperfections
    - Geometry => (Mis)Alignment
    - Anomalous conditions
      - Pressures, temperatures => Gains, refractive indices
      - Contractions, expansions
DDAlign: Status and Plans

- **Clarification:**
  DDAlign does not provide *algorithms* to determine alignment constants and never will. DDAlign supports hosting the results of such algorithms and applies the resulting imperfections.

- **Basic functionality under development**

- **Alignment data are typically time-stamped**
  - String connections to conditions database
  - Persistency needs to be integrated with DDCond
  - Cannot complete without DDCond
DDCond: Conditions Data

- Time dependent data necessary to process the detector response [of particle collisions]
- Conditions data support means to Provide access to a consistent set of values according to a given time
  - Fuzzy definition of a “consistent set” typically referred to as “interval of validity”
  - May be time interval, run number, named period, ...
  - Configurable and extensible
- Data typically stored in a database
Conditions Data: Consistent Dataset

Production version:
- VDET: v3 for T < 3, v2 for 3 < T < 5, v3 for 5 < T < 9, v1 for T > 9
- HCAL: v1 for 1 < T < 2, v2 for 2 < T < 8, v1 for T > 8
- RICH: v1 everywhere
- ECAL: v1 everywhere

[Time = T]

[Image: Diagram showing version and time relationship for Data Items: VDET alignment, HCAL calibration, RICH pressure, ECAL temperature]

[Pere Mato / 2000]
DDCond: Workplan

- **The transient implementation**
  - Flexible definition and handling of intervals of validity
    => Key point

- **Persistent implementation**
  - Define interface/ABC
  - Proof of concept using one XML, SQLite, Oracle, ...
Toolkit Users

Users are mandatory for feedback to avoid developments in thin air (i.e. purely academic)

- **ILD:** F. Gaede et al., ported complete Mokka model ILD_o1_v05
- **CLICdp:** starting new design after CDR
- **FCC-eh:** P. Kostka et al.
- **FCC-hh:** starting, A. Salzburger et al.

<table>
<thead>
<tr>
<th>DD4hep</th>
<th>DDG4</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>X</td>
<td></td>
</tr>
</tbody>
</table>
Summary and Outlook

- The DD4hep toolkit (+extensions) start to become accepted: Client validation has started
- Basic DD4hep API essentially stable
- Simulation kit DDG4 being validated
- Alignment support implemented
  - Requires conditions support for full functionality
    - => DDCond: extension to be developed
- Validate, verify, enhance and document
Work Plan

- **The DD4hep toolkit (+extensions)**
  - API basically stable
  - Client validation has started

- **Simulation kit DDG4 being validated**
  - Geant4 multi-threading
  - Fast simulation

- **Support alignment and conditions handling**
  - Main work items

- **Validate, verify, enhance and document**
Backup
Standard Detector Palette: DDDetectors

- **Mostly arose from the SiD model**
  - Layer based detectors
  - Tracker barrel & endcap
  - Several calorimeter constructs
- **Partially with measurement surfaces**
  (see also talk by F. Gaede)
- **Plugin mechanism to enhance detector elements**
  - Neat mechanism to attach user defined optional data
    => Proof that 'anticipate the unforeseen' works
  - NOT intrusive to detector constructors
  - Flexible definition of the measurement surface