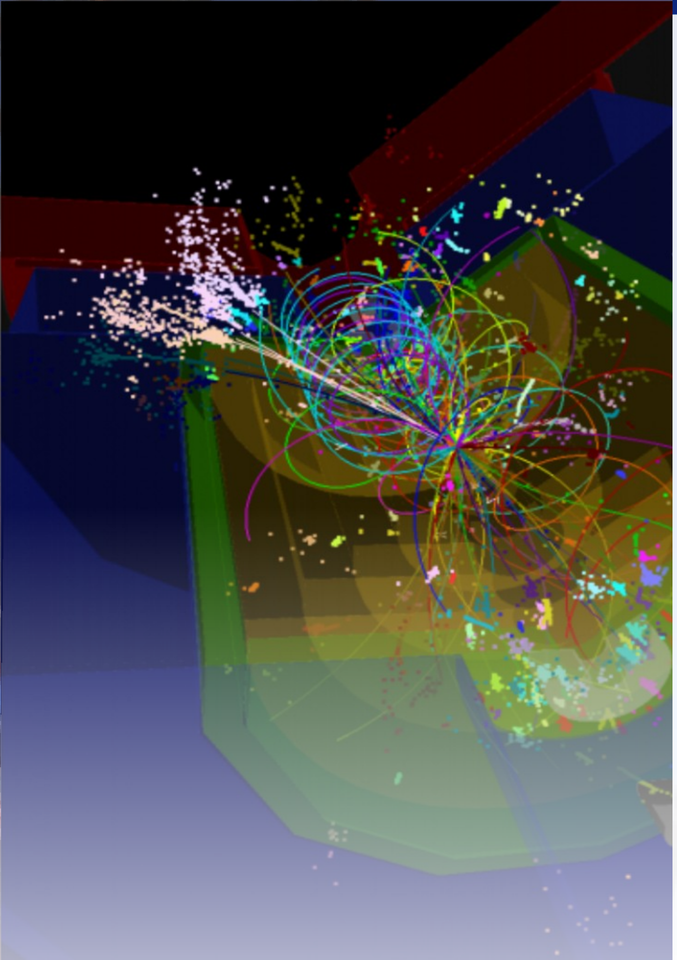


DD4hep

Detector Description Status

DD4hep motivation, goals, components and usage

M. Frank[1], F. Gaede[2], N. Nikiforou[1], M. Petric[1], A. Sailer[1]

[1] CERN
[2] DESY

February 24th., 2016

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.

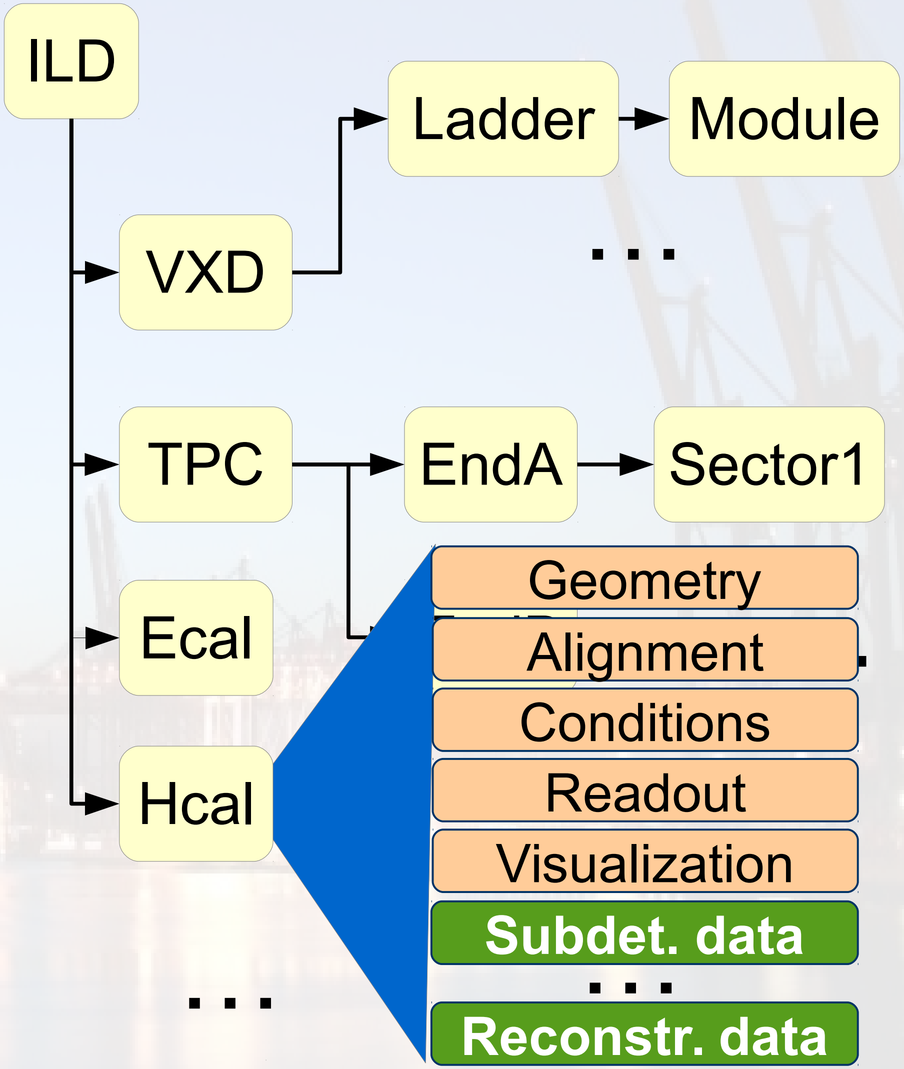
[*] DD4hep is a sub-package of AIDA2020 WP3: http://aidasoft.web.cern.ch

Motivation and Goal

- Minimize and avoid new developments
- Rather: attempt to coherently combine in a user friendly manner what belongs together:
  - Detector Geometry
  - Detector Alignment and Conditions
  - Simulation using Geant4
- Let's be driven by laziness ...
  - Get most out of it with minimal efforts
  - In particular when developing new detector concepts

What is Detector Description ?

- Description of a detector as a tree-like hierarchy of 'detector elements'
  - Sub-detectors or parts of subdetectors
- Detector Element describes
  - Geometry
  - Environmental conditions
  - Properties required to process event data
  - Optionally: experiment, sub-detector or activity specific data



ILD

VXD → Ladder → Module

. . .

TPC → EndA → Sector1

Ecal

Hcal

. . .

Geometry

Alignment

Conditions

Readout

Visualization

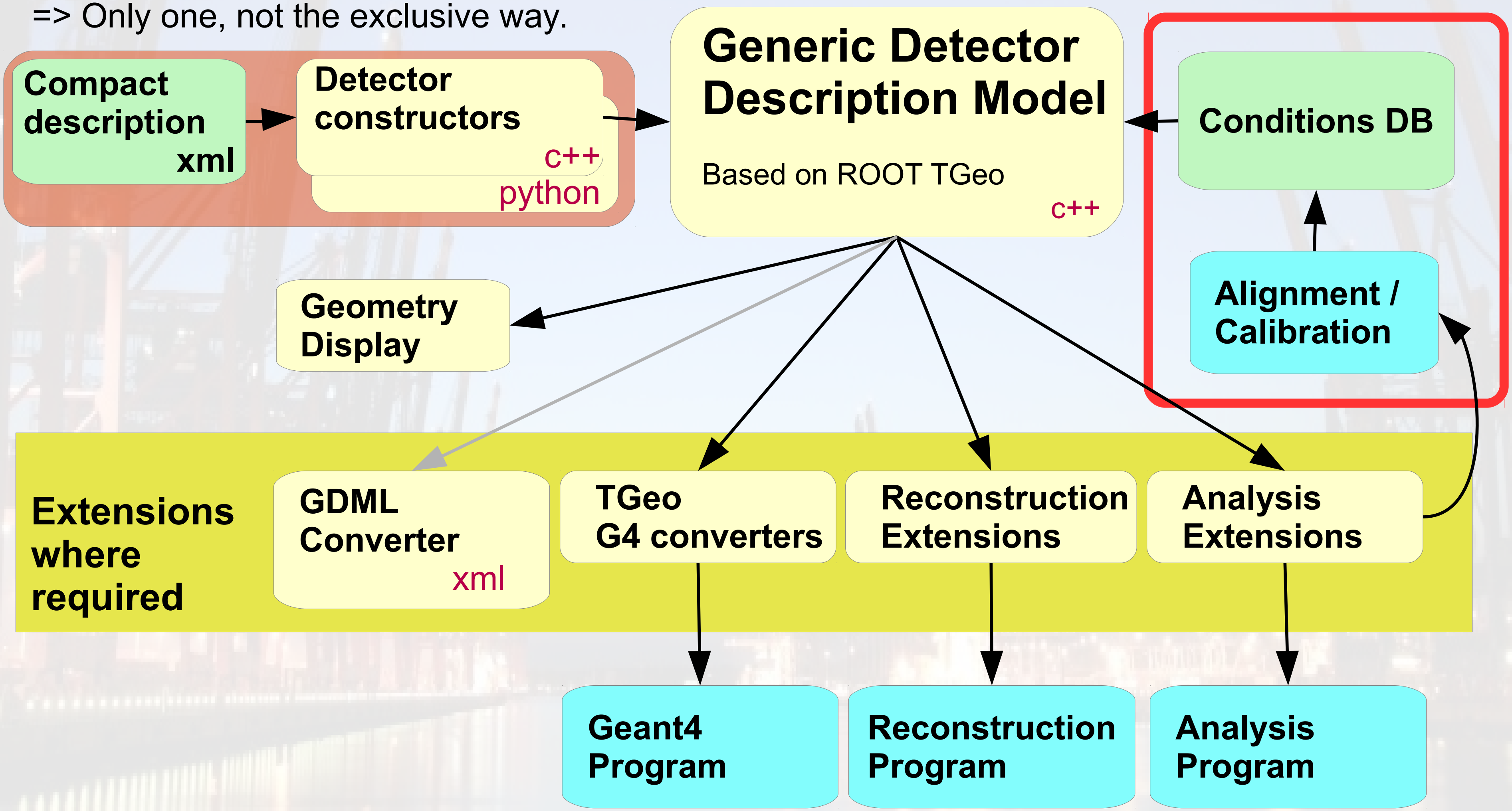
Subdet. data

. . .

Reconstr. data

DD4Hep - The Big Picture

Note:
DD4hep population is plugin based
=> Only one, not the exclusive way.



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

GDML Converter xml

TGeo G4 converters

Reconstruction Extensions

Analysis Extensions

Geant4 Program

Reconstruction Program

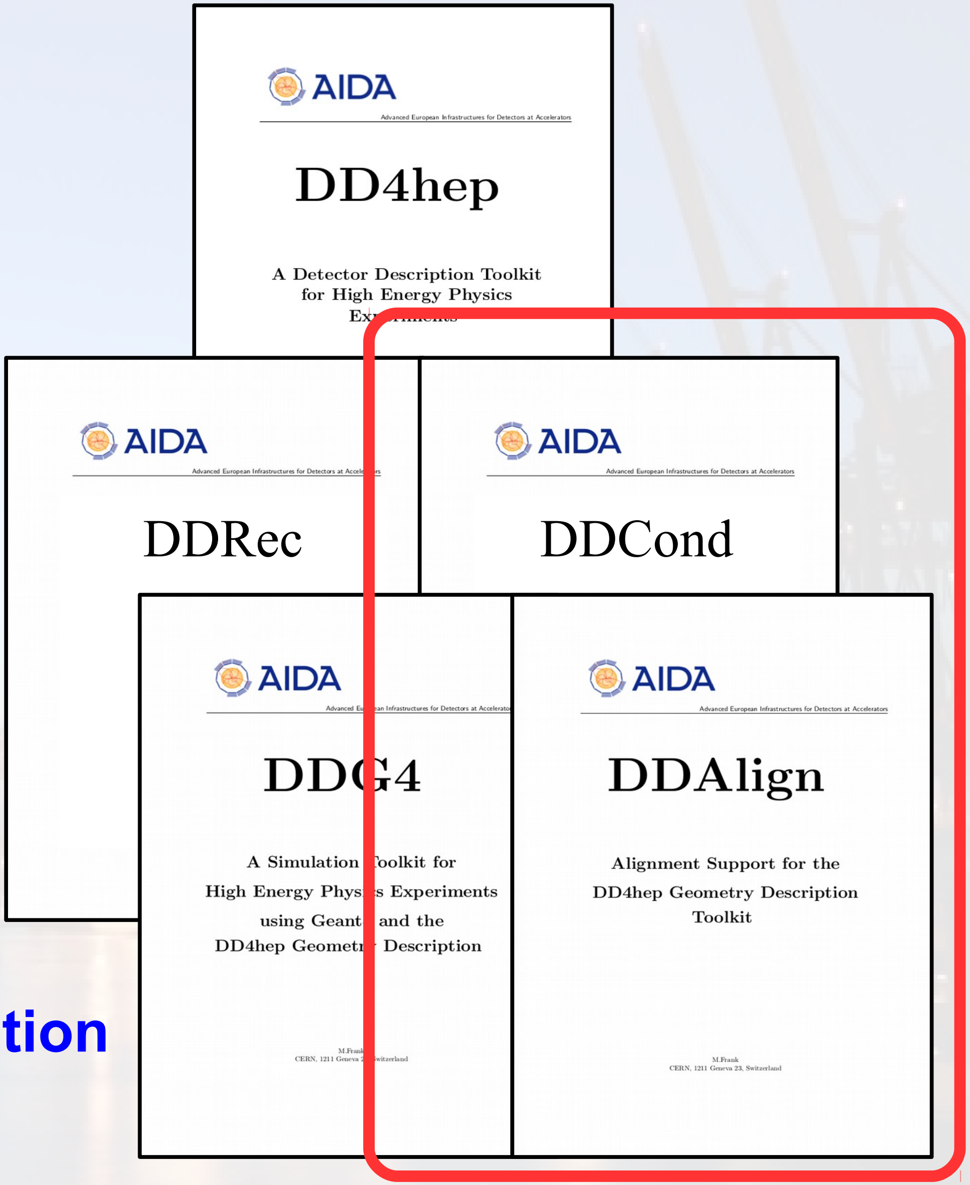
Analysis Program

Saga in 5 Episodes: Sub-packages

- **DD4hep – basics/core**
  - Basically stable
- **DDG4 – Simulation using Geant4**
  - Towards end of validation
- **DDRec – Reconstruction supp.**
  - Driven by FG, NN, AS, …
- **DDAlign – Alignment support (*)**
- **DDCond – Detector conditions (*)**

(*) In work:

- **Hope to get LHCb in the boat**
- **Need running experiment to achieve a realistic implementation**



AIDA
Advanced European Infrastructures for Detectors at Accelerators

DD4hep

A Detector Description Toolkit
for High Energy Physics
Experiments

AIDA
Advanced European Infrastructures for Detectors at Accelerators

DDRec

AIDA
Advanced European Infrastructures for Detectors at Accelerators

DDCond

AIDA
Advanced European Infrastructures for Detectors at Accelerators

DDG4

A Simulation Toolkit for
High Energy Physics Experiments
using Geant4 and the
DD4hep Geometry Description

M.Frank
CERN, 1211 Geneva 23, Switzerland

AIDA
Advanced European Infrastructures for Detectors at Accelerators

DDAlign

Alignment Support for the
DD4hep Geometry Description
Toolkit

M.Frank
CERN, 1211 Geneva 23, Switzerland

DD4hep Core

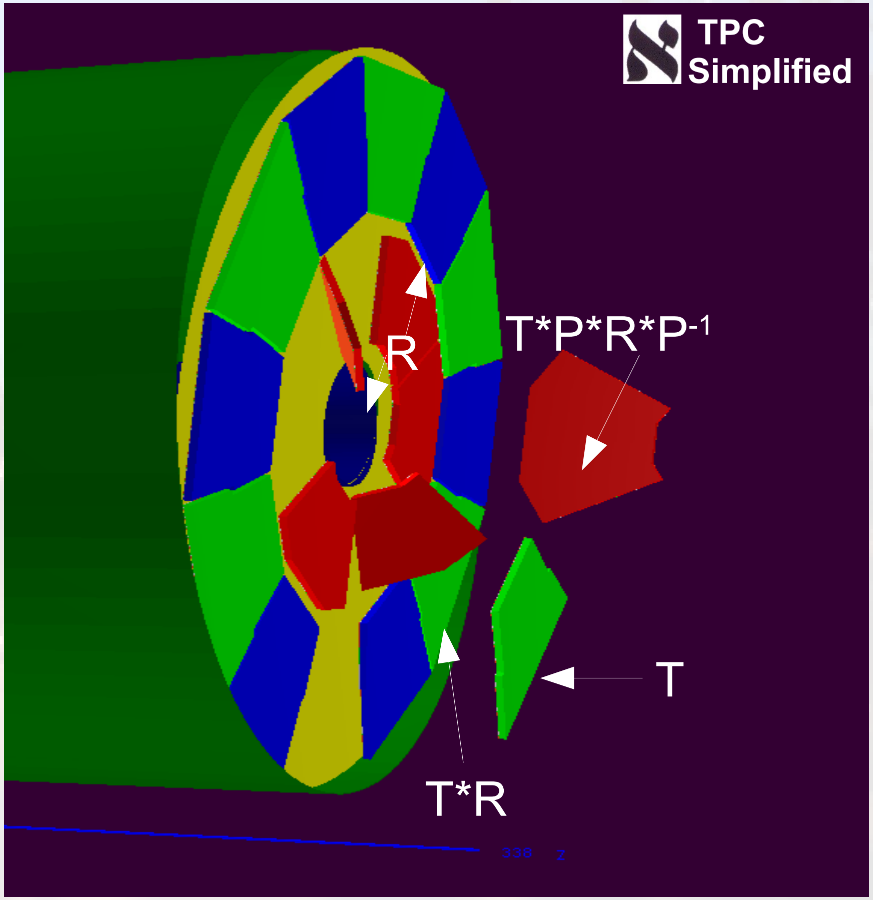
- Handles the functionality of detector elements
- Basically stable
  - Bug fixes, enhancements
- OpenGL displays come for free (ROOT)
  - Good graphics is indispensable to debug a geometry
- Objects are fully reflective
  - C++ dictionaries defined
  - Cross-language development & interactivity: Cint, Python

Simulation: DDG4

- Simulation = Geometry + Detector response + Physics
- Concept: Formalization of Geant4
  - Automatic conversion from ROOT to Geant4
  - Instantiate objects palette:
    Physics list, -constructors, sens. detectors
  - Start simulating
- No extra (C++) user code necessary
  - But not inhibited e.g. sophisticated sensitive detectors
  - Uses heavily plugin mechanisms
- Flexible configuration with python or Cint [, XML]
- Support for Geant4 multi-threading
  (event - parallelism)

DDAlign: Detector Alignment

- Fundamental functionality to interpret event data in the real and imperfect world
  - Must handle imperfections
    - Geometry => (Mis)Alignment
  - Anomalous conditions
    - Pressures, temperatures
    - => Gains, refractive indexes
    - => Contractions, expansions
  - Basic functionality present
    - But no connection to persistency present (yet)
    - Needs conditions to apply alignments as a timed sequence



DDAlign: Detector Alignment

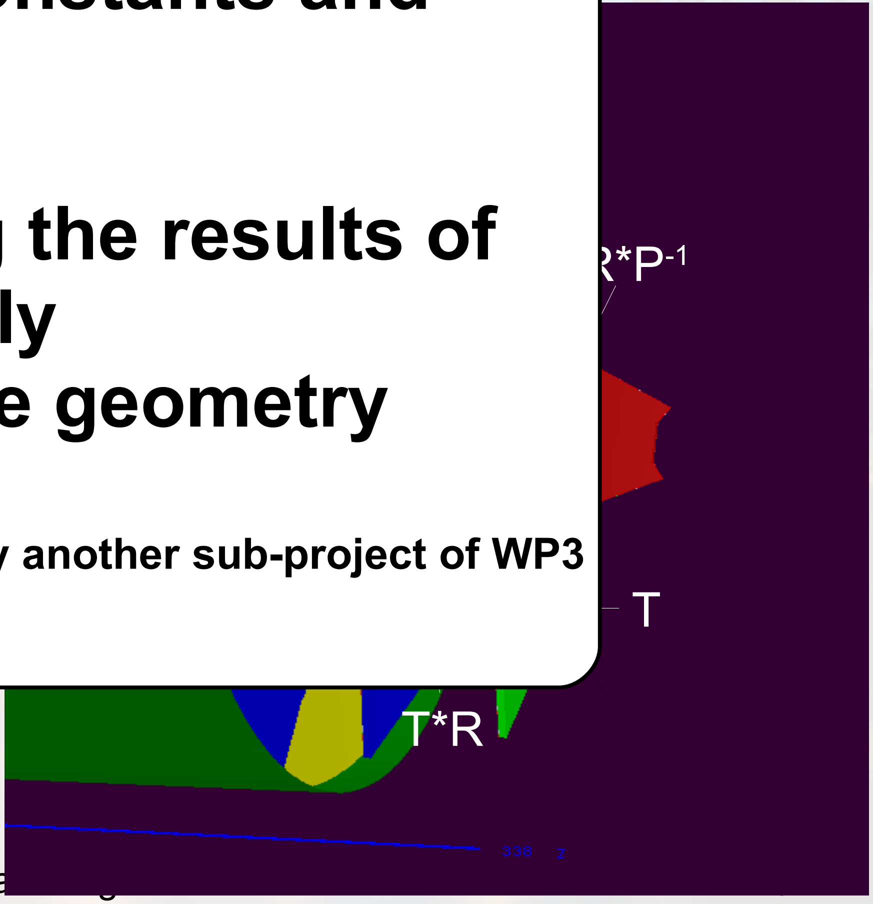
- Fu in
  th

Please Note:

DDAlign does not provide *algorithms* to determine alignment constants and never will [(*)]

DDAlign supports hosting the results of the algorithms and to apply alignment constants to the geometry

(*) Alignment procedures investigated by another sub-project of WP3 (Chris Parkes et al.)



DDCond: Conditions Data

- Time dependent data necessary to process the detector response [of a particle collisions]
- Conditions data support means to "Provide access to a consistent set of values according to a given event"
  - Fuzzy definition of a "consistent set" typically referred to as "interval of validity"
  - May be time interval, run number, named period, ...
- Data typically stored in a database
- Currently under investigation

Multi-Threading in DDAlign & DDCond

- Mandatory: Nowadays can't sell anything without
- Has consequences in the usage
  - More sociological than technical problem
    Need to agree on use cases and API
- Example:
  - TGeo applies alignment to real geometry
  - Nice: can simulate misaligned geometries
  - But: How do you deal with reconstruction in the presence of run-changes and multiple concurrent events, with multiple versions of conditions data ?
    - Either: geometry is constant (and drain event loop) bef ore application of new constants
    - Or: apply misalignments on the spot during hit reconstruction
      => Conditions are accessed using event's IOV

Multi-Threading in DDAlign & DDCond

- LC community has no strong use case yet
  - Early experiment phase
- Need running experiment
  - LHCb plans to investigate the use of DD4hep for the upgrade (LHC LS2)
  - Manpower got allocated this year
  - Hope to implement missing elements according to viable usage patterns

Other Upcoming Issues

- C++11 and ROOT 6 (and dropping ROOT 5)
  - No question IF, only WHEN
- DDG4: Support fast and parametrized simulation
  - Speed-up by avoiding full Geant4 machinery
- DDG4: Revisit integration in experiment frameworks
  - Depends on future requirements of LHCb / FCC (Gaudi) and ILC (Marlin)

Toolkit Users

Users are mandatory for feedback to avoid purely academic developments in thin air...

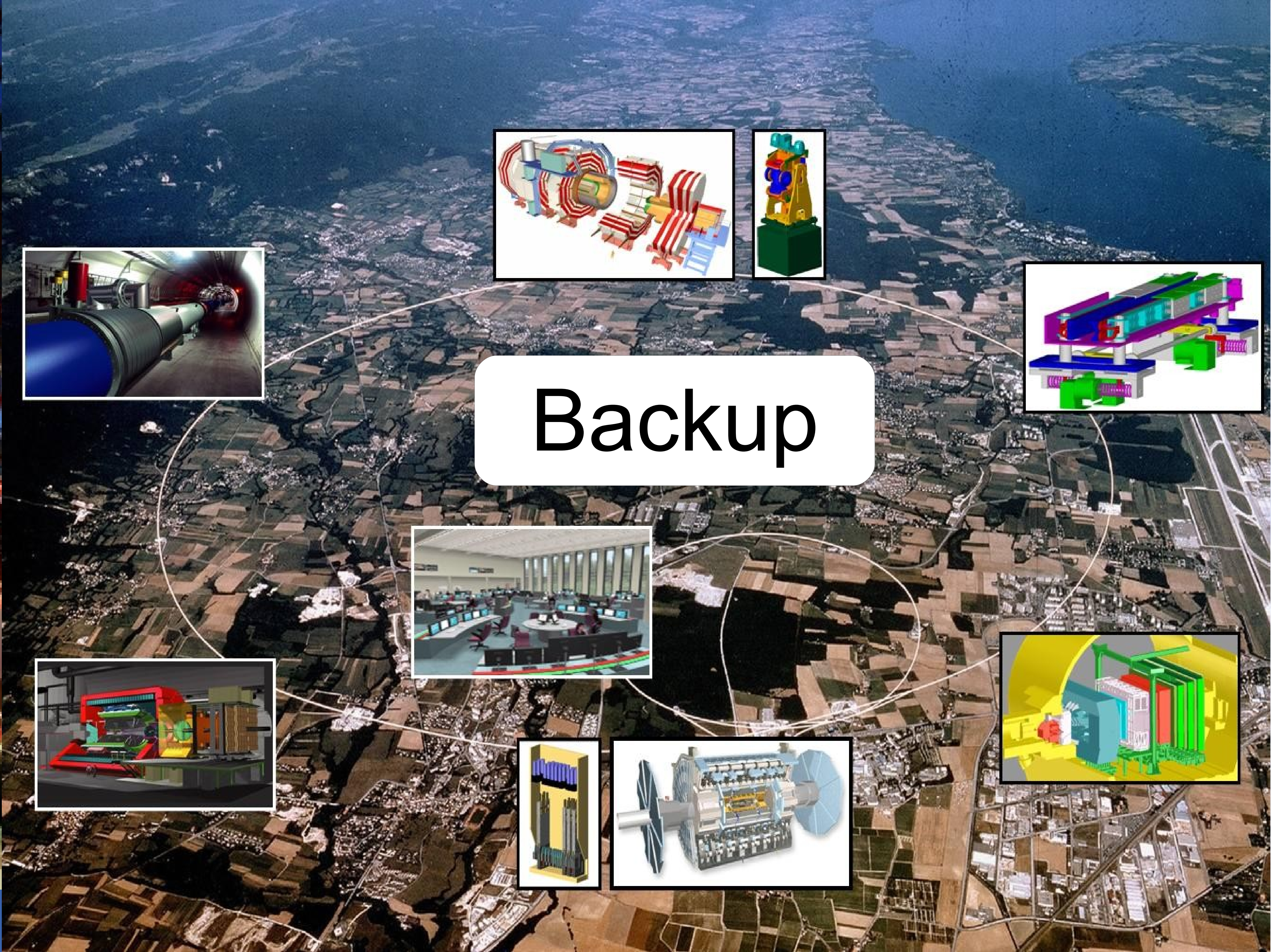
DD4hep would not where it is without its users

• ILC	F. Gaede et al.
• CLICdp	A. Sailer et al.
• FCC-eh	P. Kostka et al.
• FCC-hh	A. Salzburger et al.
• FCC-ee	Interest was expressed
• SiD	Decision to use DD4hep taken at LCWS 2015
• CALICE	Started
• LHCb	Manpower allocated this year for upgrade

DD4hep	DDG4
X	X
X	x
X	X
X	

Summary and Outlook

- The DD4hep toolkit (+extensions) accepted: Client validation ongoing
- Simulation kit DDG4 is being validated/deployed
- Alignment / Conditions support to be reassessed
  - Will be developed in collaboration with LHCb
    => Multi-threading issues to be sorted out
- Validate, verify, enhance and document

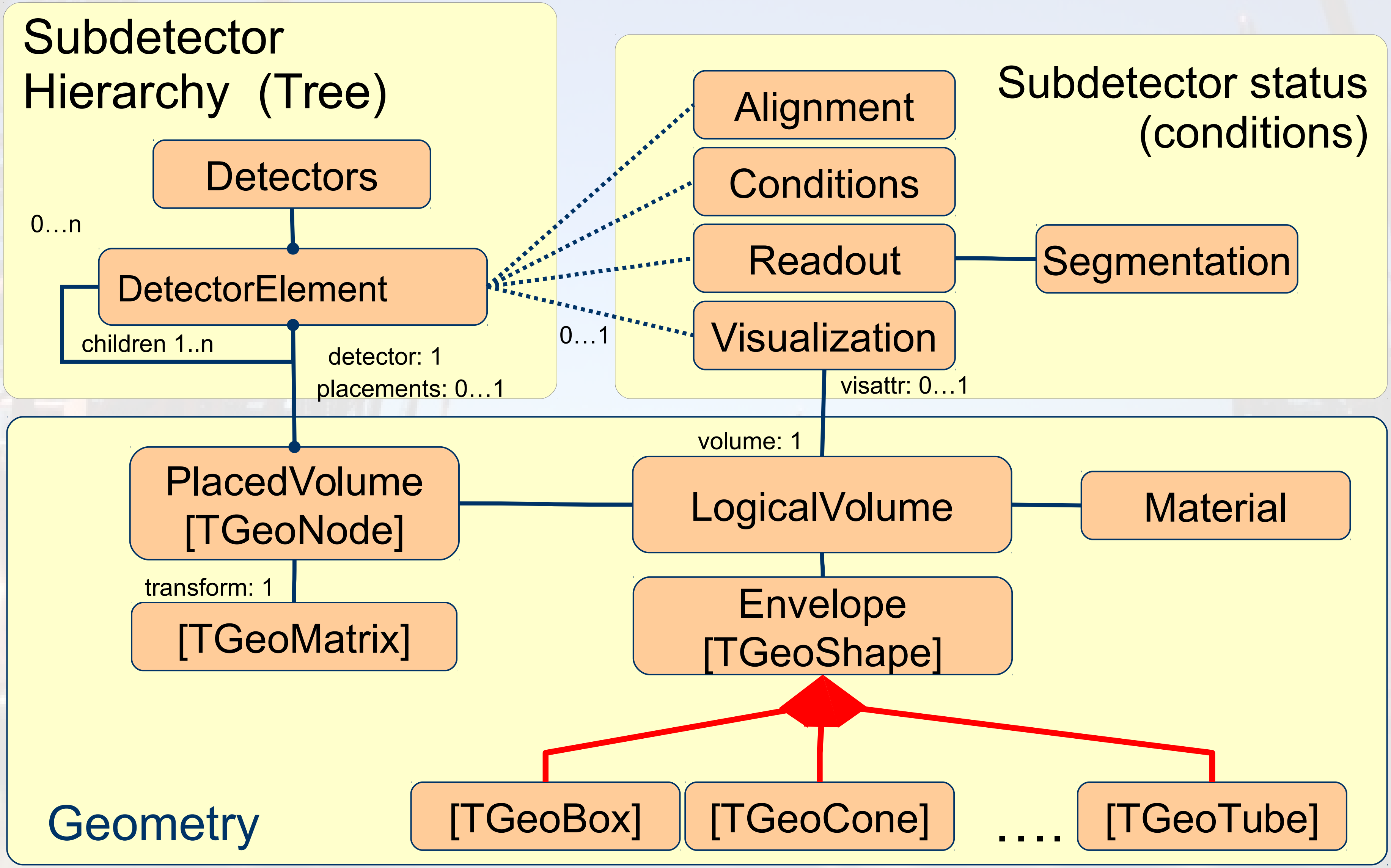


Backup

Design Principles

- Separation of data and behavior
  - Data are fully accessible (no encapsulation!)
  - Behavioral classes are wrappers around objects containing data only
  - There may be many behavioral wrapper implementations using the same data objects
    - User chooses “most suitable” behavior
  - One “data-object” may be shared among many behavioral wrapper instances

Class Diagram: Detector Element



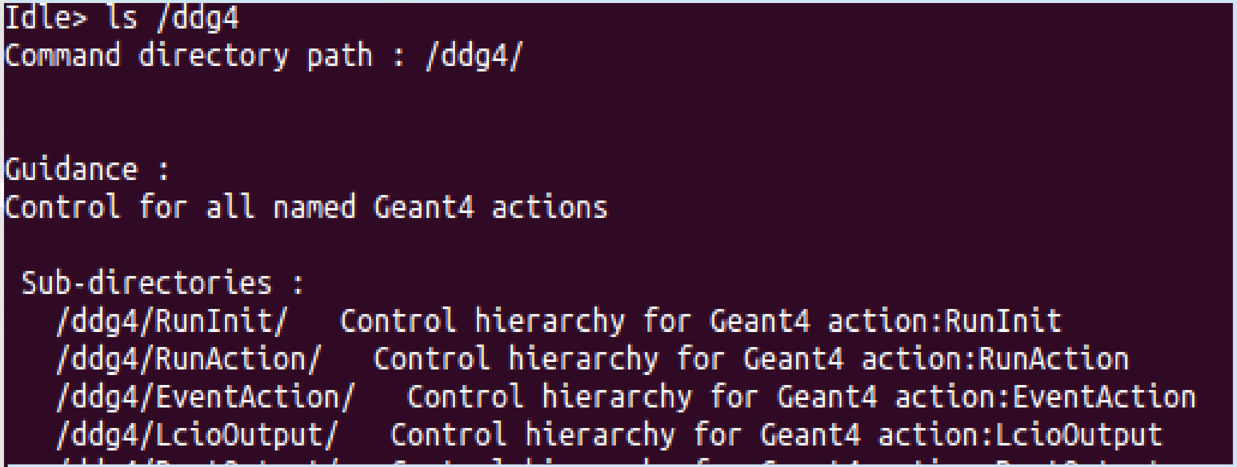
Standard Detector Palette: DDDetectors

- 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

Geant4 Interactivity



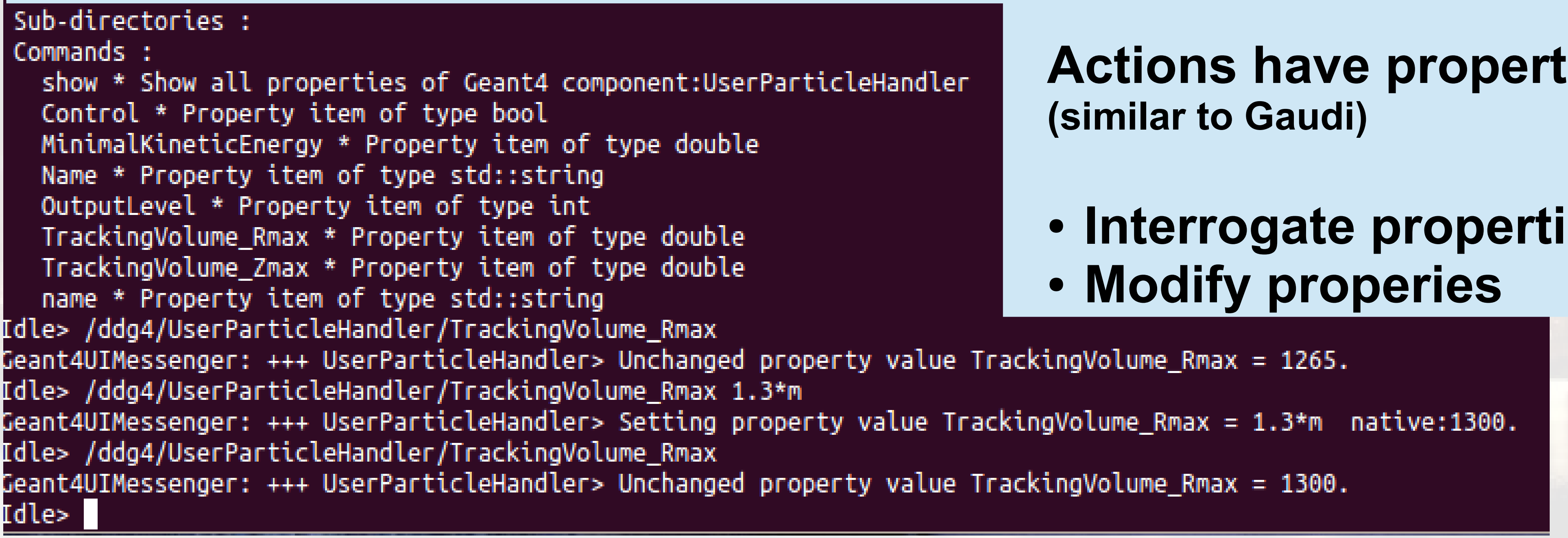
```
Idle> ls /ddg4
Command directory path : /ddg4/


Guidance :
Control for all named Geant4 actions

 Sub-directories :
   /ddg4/RunInit/   Control hierarchy for Geant4 action:RunInit
   /ddg4/RunAction/   Control hierarchy for Geant4 action:RunAction
   /ddg4/EventAction/   Control hierarchy for Geant4 action:EventAction
   /ddg4/LcioOutput/   Control hierarchy for Geant4 action:LcioOutput
```

Geant4 interactivity interfaced to every action object

- **Enabled on request**

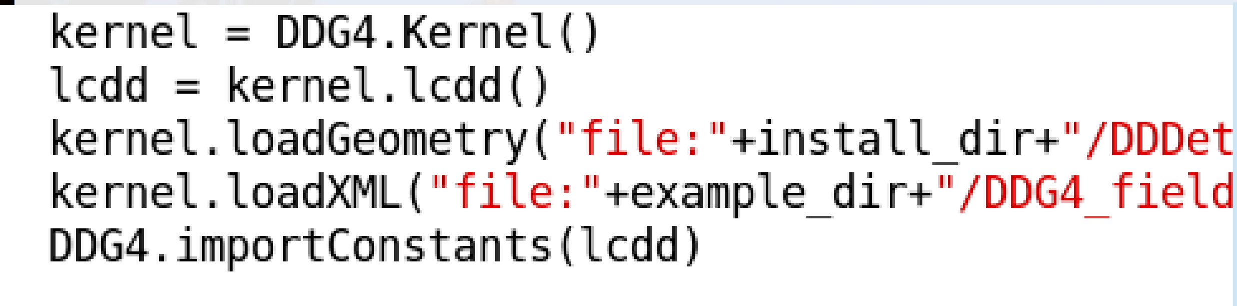


```
 Sub-directories :
 Commands :
   show * Show all properties of Geant4 component:UserParticleHandler
   Control * Property item of type bool
   MinimalKineticEnergy * Property item of type double
   Name * Property item of type std::string
   OutputLevel * Property item of type int
   TrackingVolume_Rmax * Property item of type double
   TrackingVolume_Zmax * Property item of type double
   name * Property item of type std::string
Idle> /ddg4/UserParticleHandler/TrackingVolume_Rmax
Geant4UIMessenger: +++ UserParticleHandler> Unchanged property value TrackingVolume_Rmax = 1265.
Idle> /ddg4/UserParticleHandler/TrackingVolume_Rmax 1.3*m
Geant4UIMessenger: +++ UserParticleHandler> Setting property value TrackingVolume_Rmax = 1.3*m  native:1300.
Idle> /ddg4/UserParticleHandler/TrackingVolume_Rmax
Geant4UIMessenger: +++ UserParticleHandler> Unchanged property value TrackingVolume_Rmax = 1300.
Idle> 
```

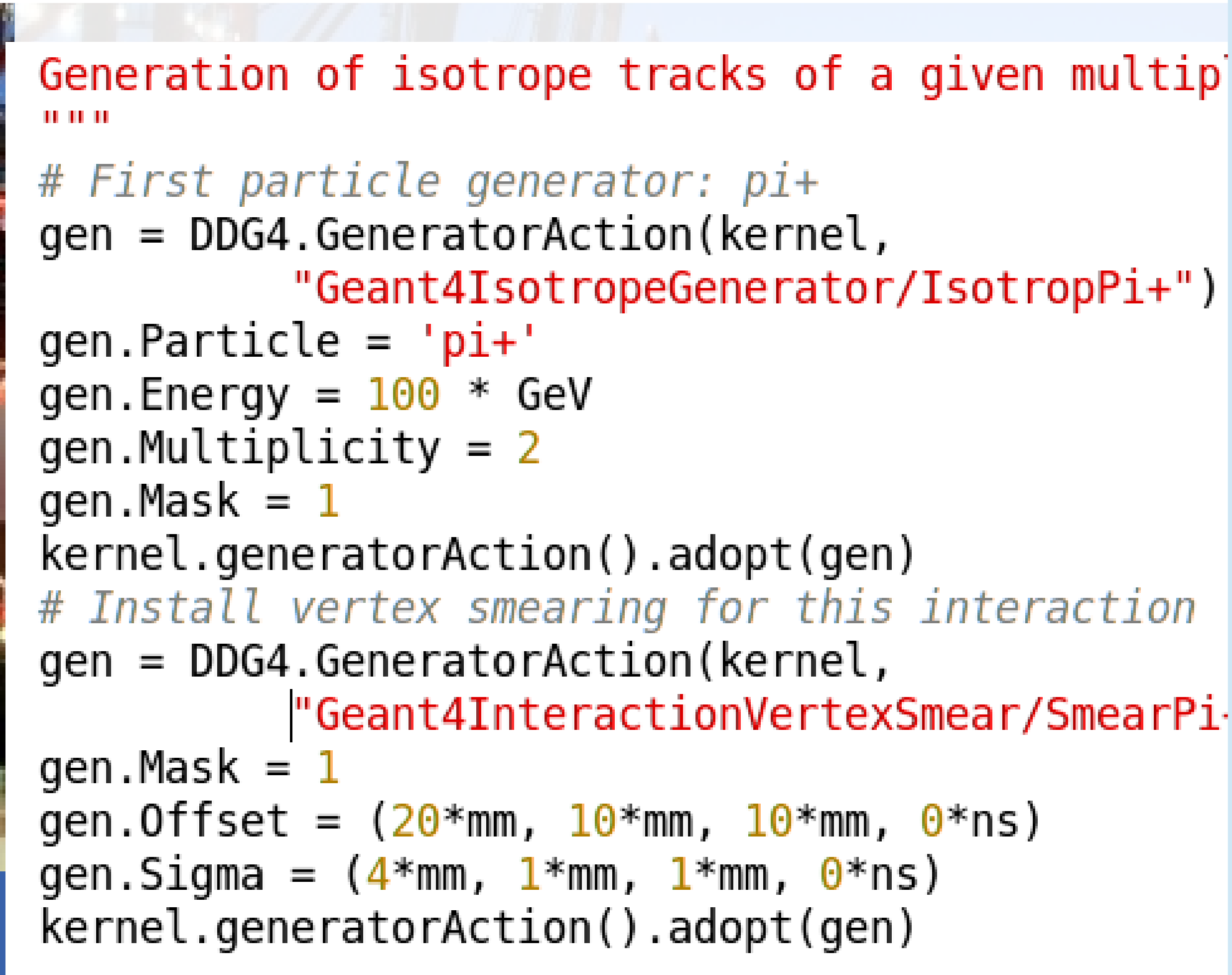
Actions have properties
(similar to Gaudi)

- **Interrogate properties**
- **Modify properies**

Configure DDG4 Application with python



```
kernel = DDG4.Kernel()
lcdd = kernel.lcdd()
kernel.loadGeometry("file:"+install_dir+"/DDDet
kernel.loadXML("file:"+example_dir+"/DDG4_field
DDG4.importConstants(lcdd)
```



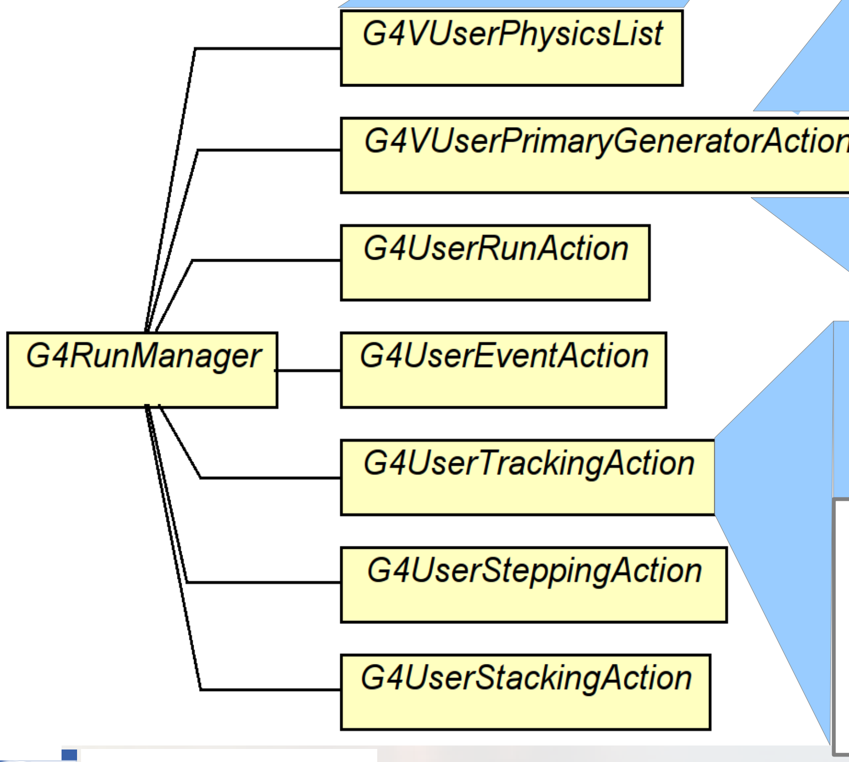
```
Generation of isotrope tracks of a given multip
"""
# First particle generator: pi+
gen = DDG4.GeneratorAction(kernel,
          "Geant4IsotropeGenerator/IsotropPi+")
gen.Particle = 'pi+'
gen.Energy = 100 * GeV
gen.Multiplicity = 2
gen.Mask = 1
kernel.generatorAction().adopt(gen)
# Install vertex smearing for this interaction
gen = DDG4.GeneratorAction(kernel,
          "Geant4InteractionVertexSmear/SmearPi
gen.Mask = 1
gen.Offset = (20*mm, 10*mm, 10*mm, 0*ns)
gen.Sigma = (4*mm, 1*mm, 1*mm, 0*ns)
kernel.generatorAction().adopt(gen)
```

- **Python configuration snippets**
  - **Loading geometry**
  - **Configuring actions**
  - **Steer Geant4 until it's prompt/batch**
- **C++ config ~ same**
- **Alternative: xml Load xml with lcdd**

Geant4 Provided Hooks

[and what we want to do inside]

Main issue: flexible configuration



Flexible definition of the physics list

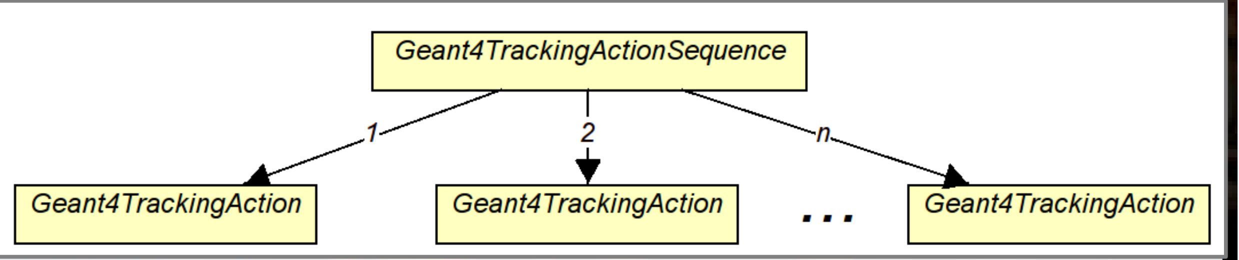
- Define particles, processes, physics constructors or use/extend predefined physics lists

Flexible data input

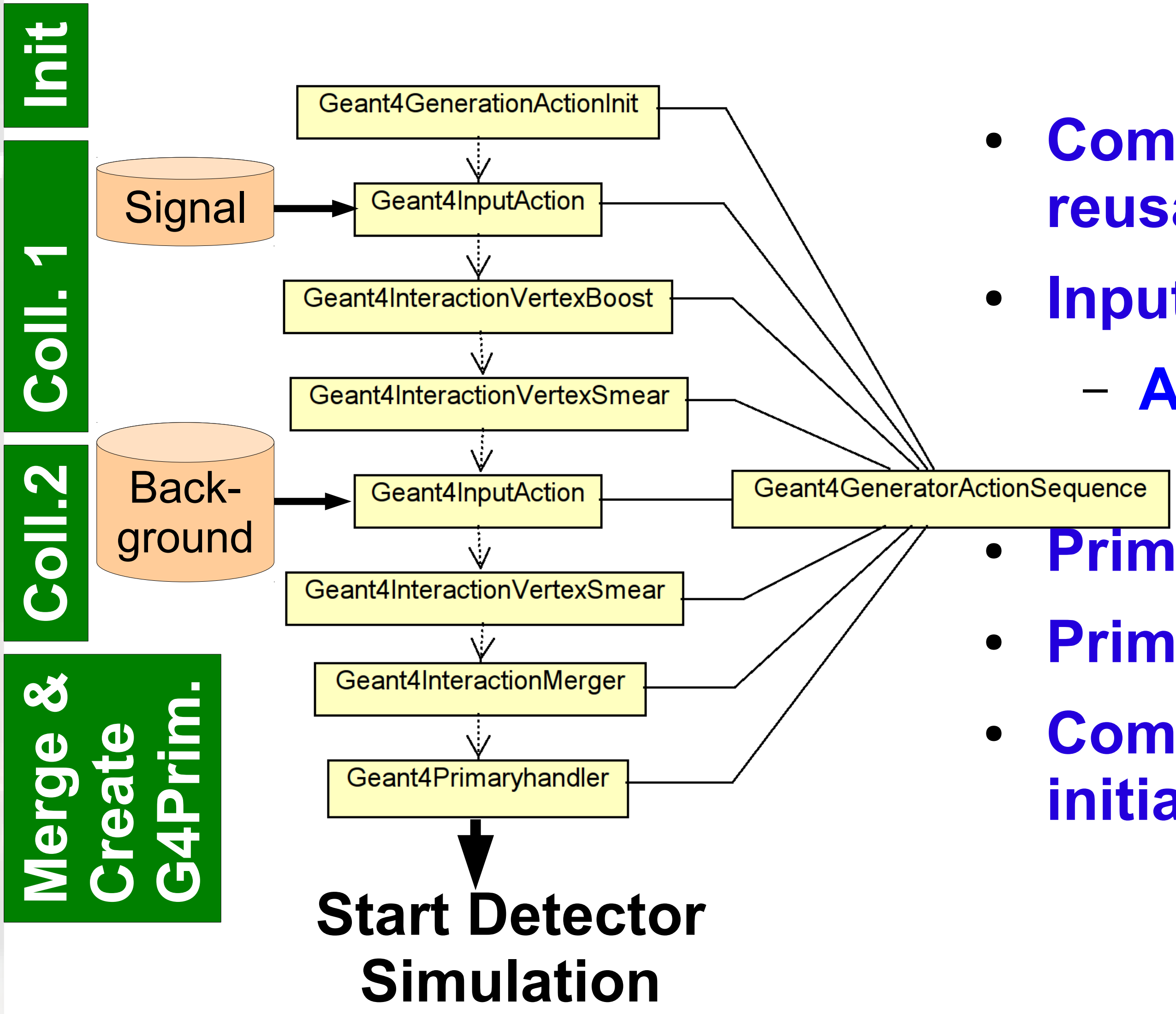
- Programmable sequence. Input from particle gun, lcio, stdhep or HepMC (text) – easily extensible
- Modules to smear and boost primary vertices
- Modules to construct interaction overlays
- Further extensions may independently added

Provide user programmable sequences

- Either as explicit object type using ABC
- Or registering a member function as callback



Example of an Action Sequence: Event Overlay with Features

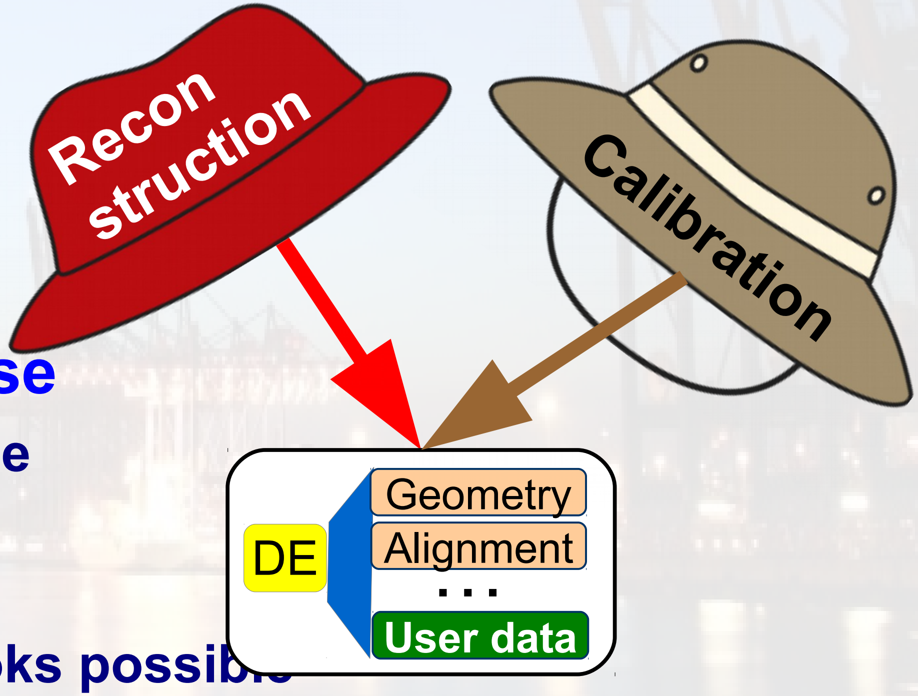

Init
Coll. 1
Coll.2
Merge & Create G4Prim.
Signal
Back-ground
Geant4GenerationActionInit
Geant4InputAction
Geant4InteractionVertexBoost
Geant4InteractionVertexSmear
Geant4InputAction
Geant4InteractionVertexSmear
Geant4InteractionMerger
Geant4Primaryhandler
Geant4GeneratorActionSequence
Start Detector Simulation

- Combine simple and reusable modules
- Input module
  - Any data format
- Primary vertex smearing
- Primary vertex boost
- Common: initialization, final merge

Views & Extensions: Users Customize Functionality

DD4hep is based on handles to data

- Clients only use the handles
- Possibility of many views based on the same DE data
  - Associate different behavior to the same data
  - Views consistent by construction
  - User data according to needs
- Be prudent: blessing or curse
  - User data: common knowledge
  - No one fits it all solution
  - Freedom is also to not do everything what somehow looks possible


Recon struction
Calibration
DE
Geometry
Alignment
. . .
User data