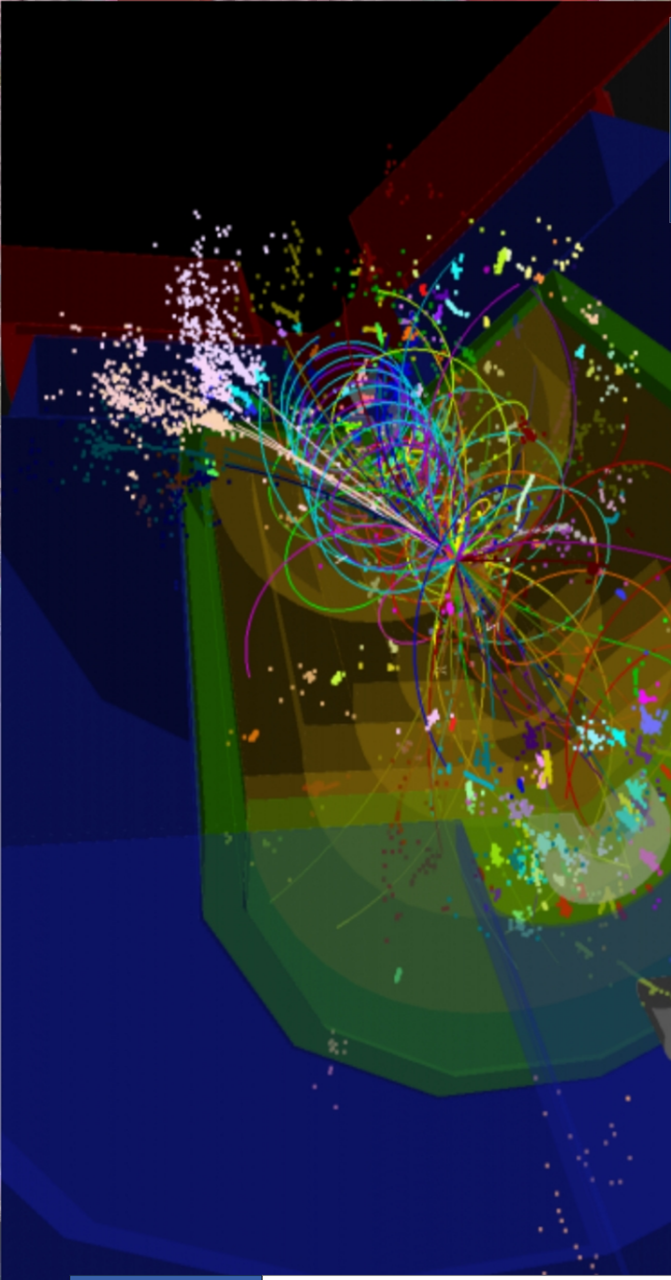


DD4hep

HEP detector description supporting the full experiment life cycle

M.Frank, F.Gaede, M.Petric, A.Sailer

CERN

This project has received funding from the European Union's Horizon 2020 Research and Innovation programme under Grant Agreement no. 654168.

AIDA 2020

- **Motivation and Goals**
- DD4hep core
- Detector palette for design studies
- Simulation
- Conditions and Alignment
- Miscellaneous
- Summary

Motivation and Goal

DD4hep

- **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, supporting**
    - **simulation, reconstruction, analysis**
  - **Full description, including**
    - **Geometry, readout, alignment, calibration etc.**

This project has received funding from the European Union’s Horizon 2020 Research and Innovation programme under Grant Agreement no. 654168.

AIDA 2020

About DD4hep & Co

DD4hep

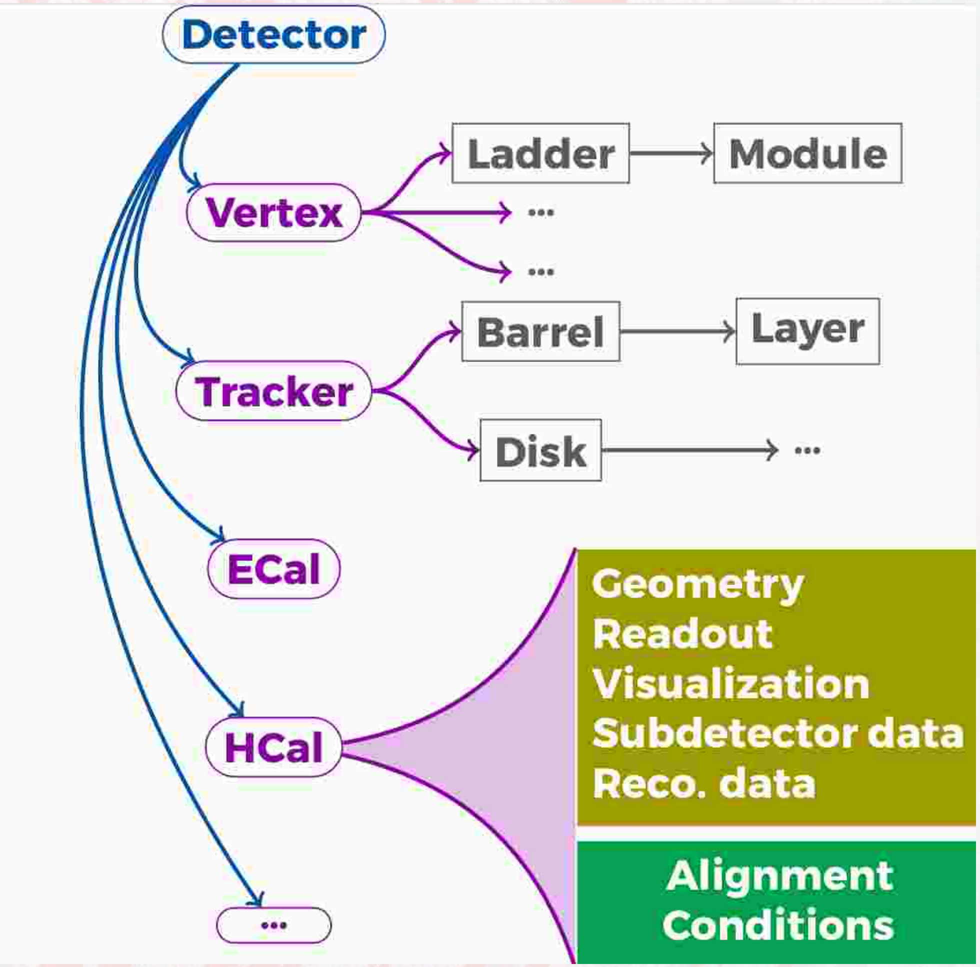
- **Effort of very few people with a simple, humble and comprehensive vision**

Detector description for the lazy
Minimal effort, pragmatic, no technical restrictions,
No obstacles induced by religious wars

- **We welcome new collaborators / users and provide support**
  - **Suggestions are welcome but not under pressure**
  - **Contributions are even more welcome**
  - **Responsible users in design and in trouble**
    => Don’t stretch & tweak it to the bitter end
    => Feed back proper analysis to fix problem
    => “It doesn't like me and answers SEGV”

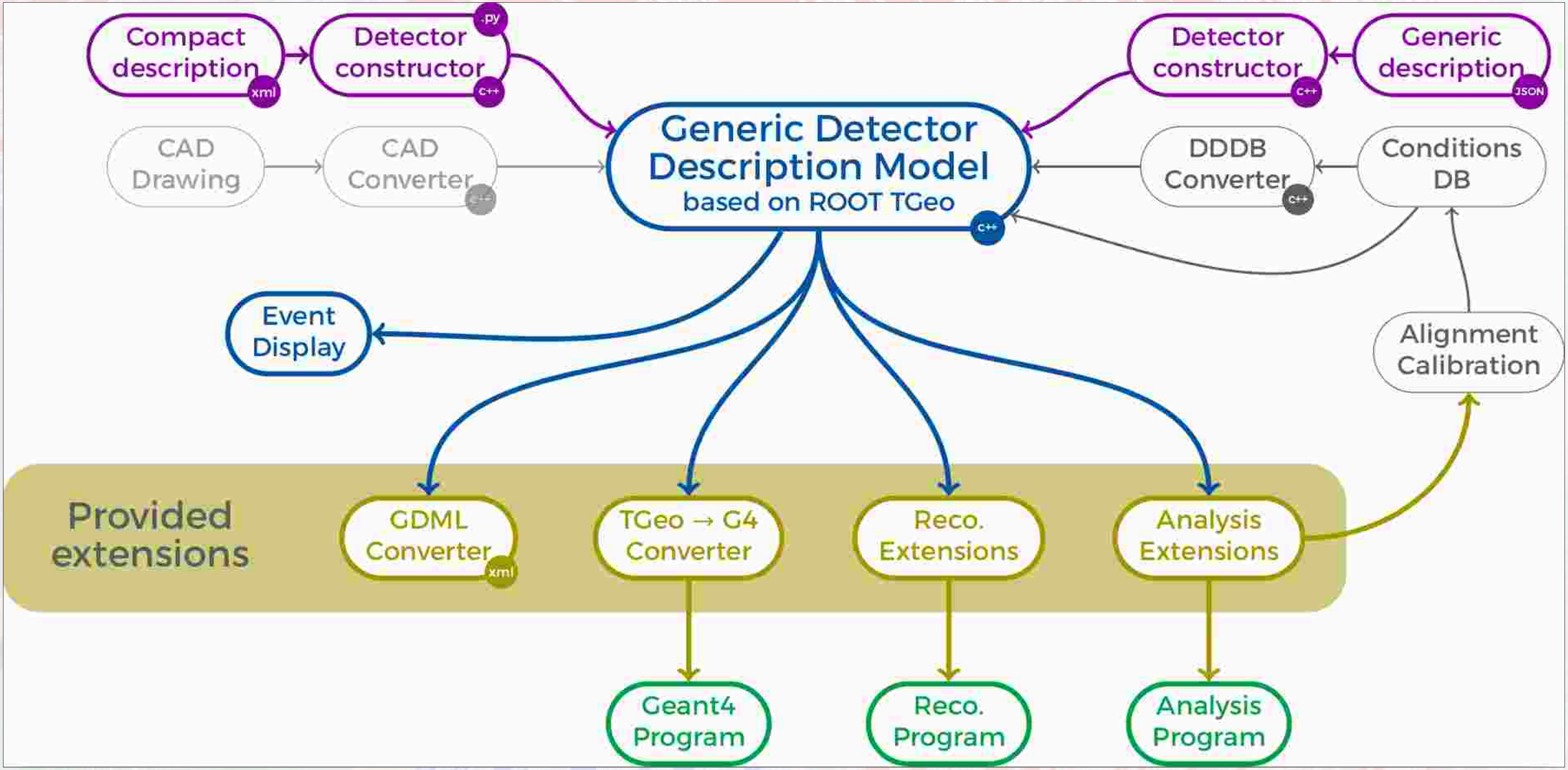
What is Detector Description ?

- Tree-like hierarchy of "detector elements"
  - Macroscopic (ie. not a strip)
  - Subdetectors or parts of subdetectors
- Detector Element
  - Geometry
  - Properties to process events
    - Environmental data
    - Alignments
    - Derivatives of these
  - Optionally experiment, sub-detector or activity specific data



DD4Hep - The Big Picture

DD4hep



Saga in 5 Episodes

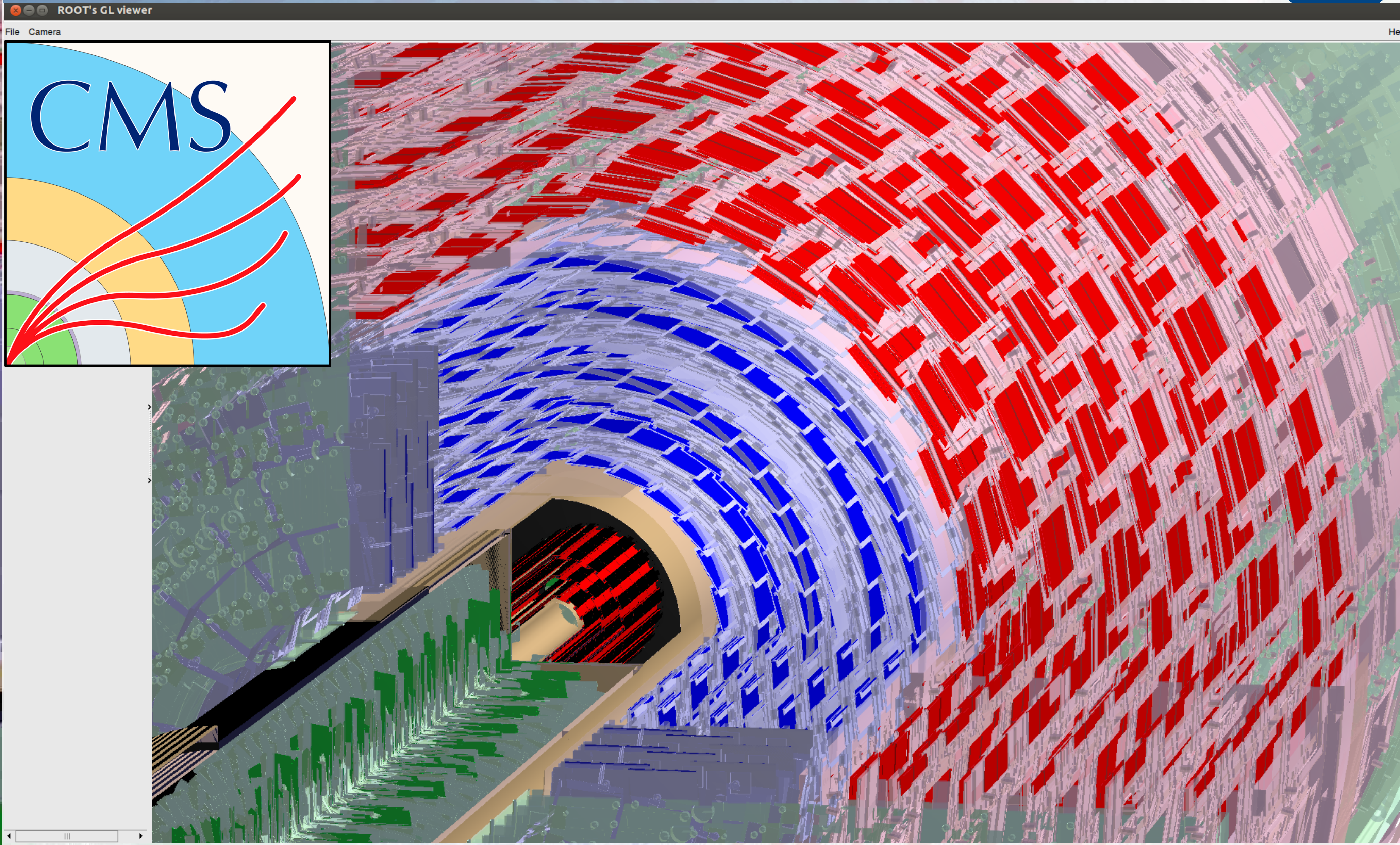
DD4hep

- **DD4hep – basics/core [1]**
- **DDG4 – Simulation using Geant4 [1]**
- **DDRec – Reconstruction supp.[2]**
- **DDCond – Detector conditions [3]**
- **DDAlign – Alignment support [3]**

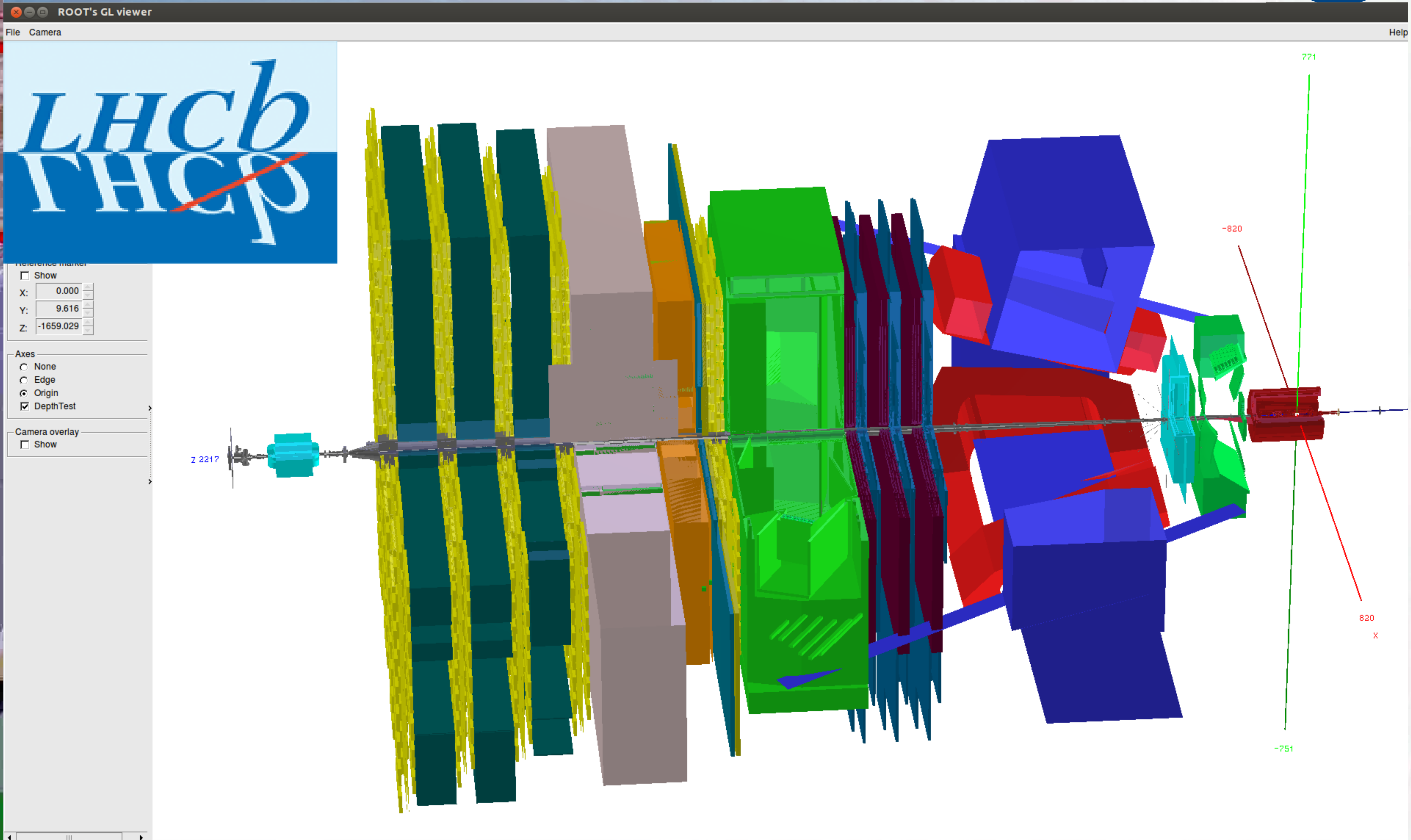
(1) Mature state: bug-fixes and maintenance
(2) F. Gaede (WP3, Task 3.6)
(3) Work since start of AIDA^{2020}

PR: CMS Trackers

DD4hep

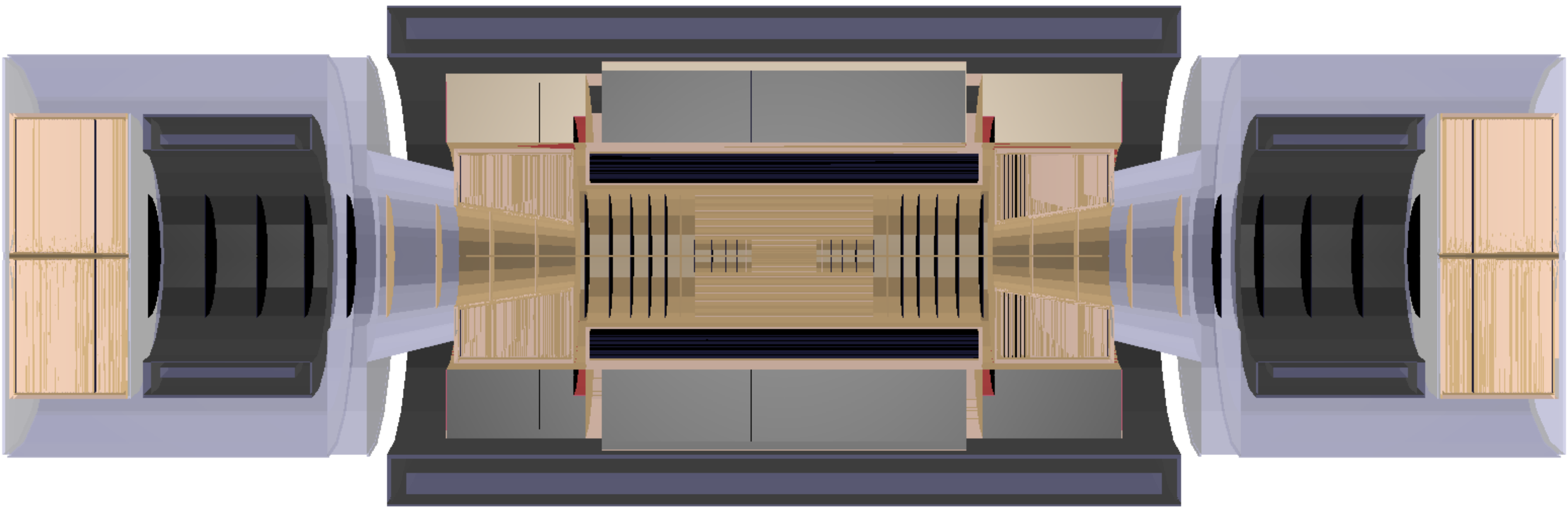


PR: LHCb Detector of Run I / II



PR: FCC Design Study

DD4hep



DD4hep

- Motivation and Goals
- **DD4hep core**
- Detector palette for design studies
- Simulation
- Conditions and Alignment
- Miscellaneous
- Summary

DD4hep Core

DD4hep

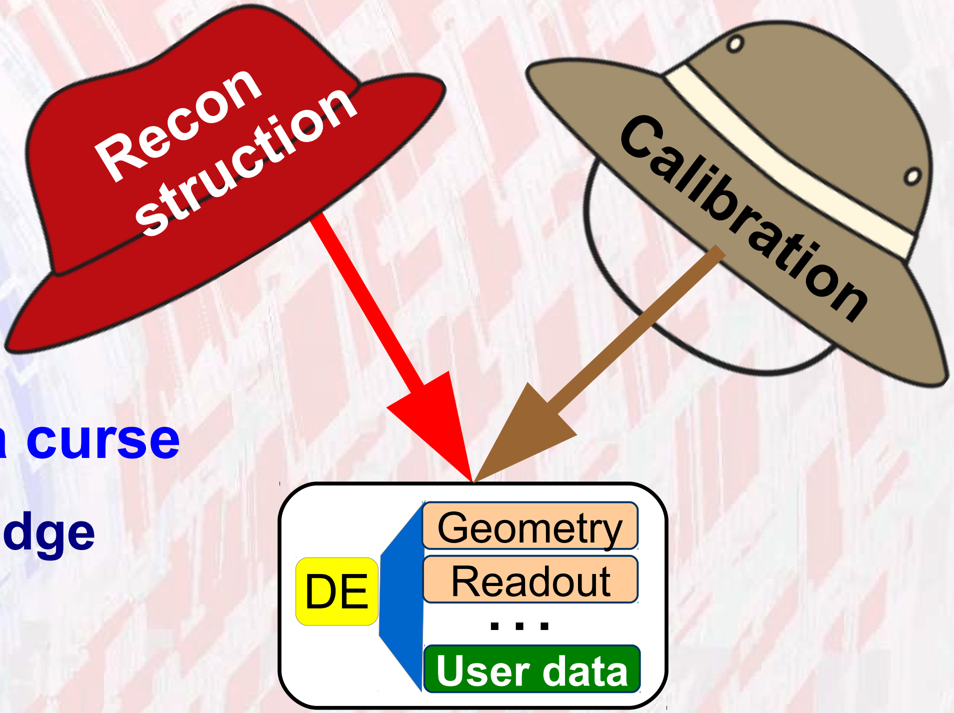
- Handles the detector element functionality
- Basically stable
  - Bug fixes, enhancements
- Objects are fully reflective
  - C++ dictionary defined
  - Intrinsic support for cross-language development
- Reflection supports interactivity
  - Cint (Cling) and python (cppyy)

Views & Extensions: Users Customize Functionality

DD4hep

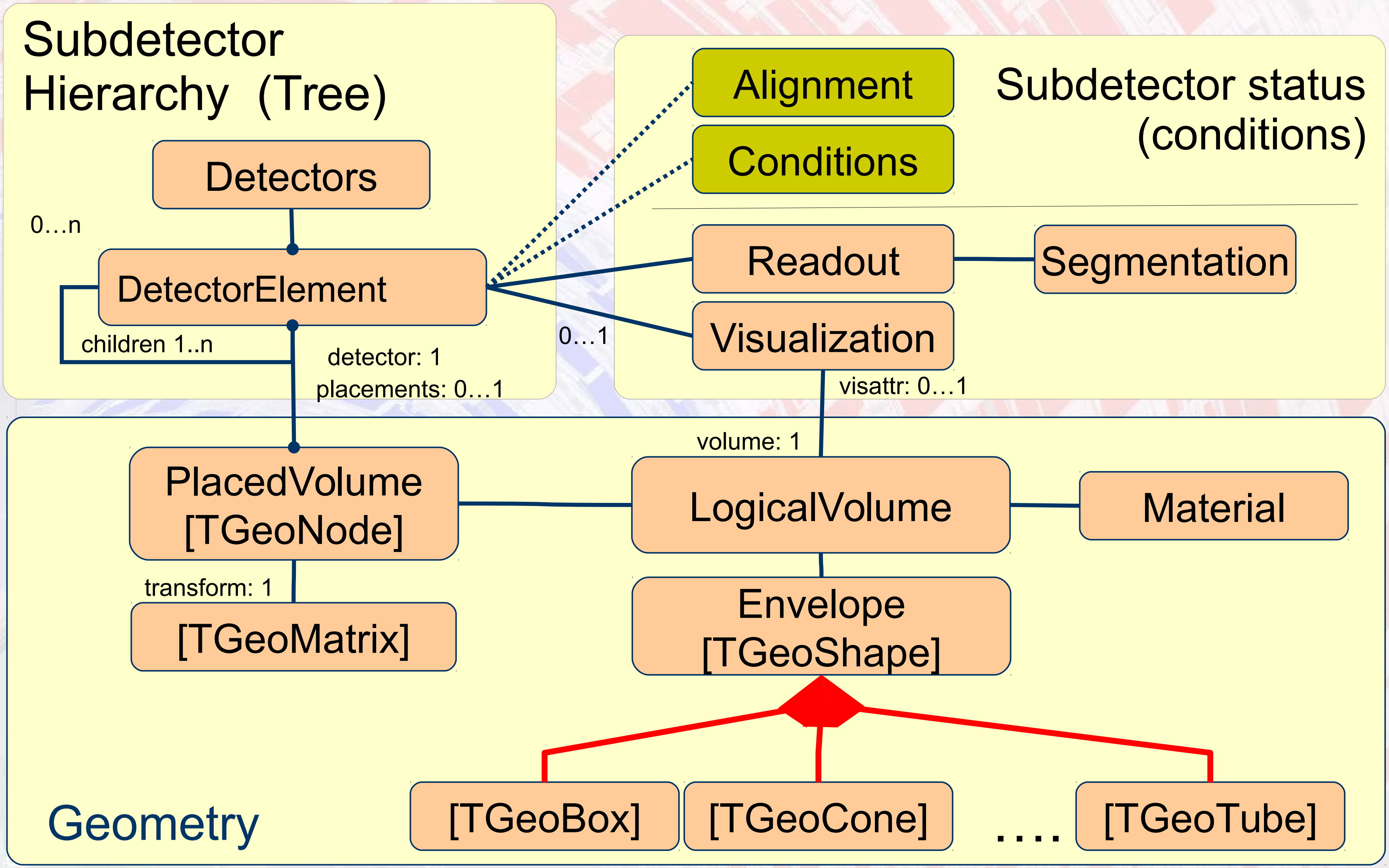
DD4hep is based on handles (smart pointers)

- Rarely deal with data directly
- Possibility of many views based on the same DE data
  - Same 'data' associated to different 'behaviors'
  - All views are consistent and creation is efficient: pointer-copy
- Be prudent: a blessing and a curse
  - User data: common knowledge



Class Diagram: Detector Element
Sort of Standard...

DD4hep



DD4hep

- Motivation and Goals
- DD4hep core
- **Detector palette for design studies**
- Simulation
- Conditions and Alignment
- Miscellaneous
- Summary

Standard Detector Palette DDDetectors

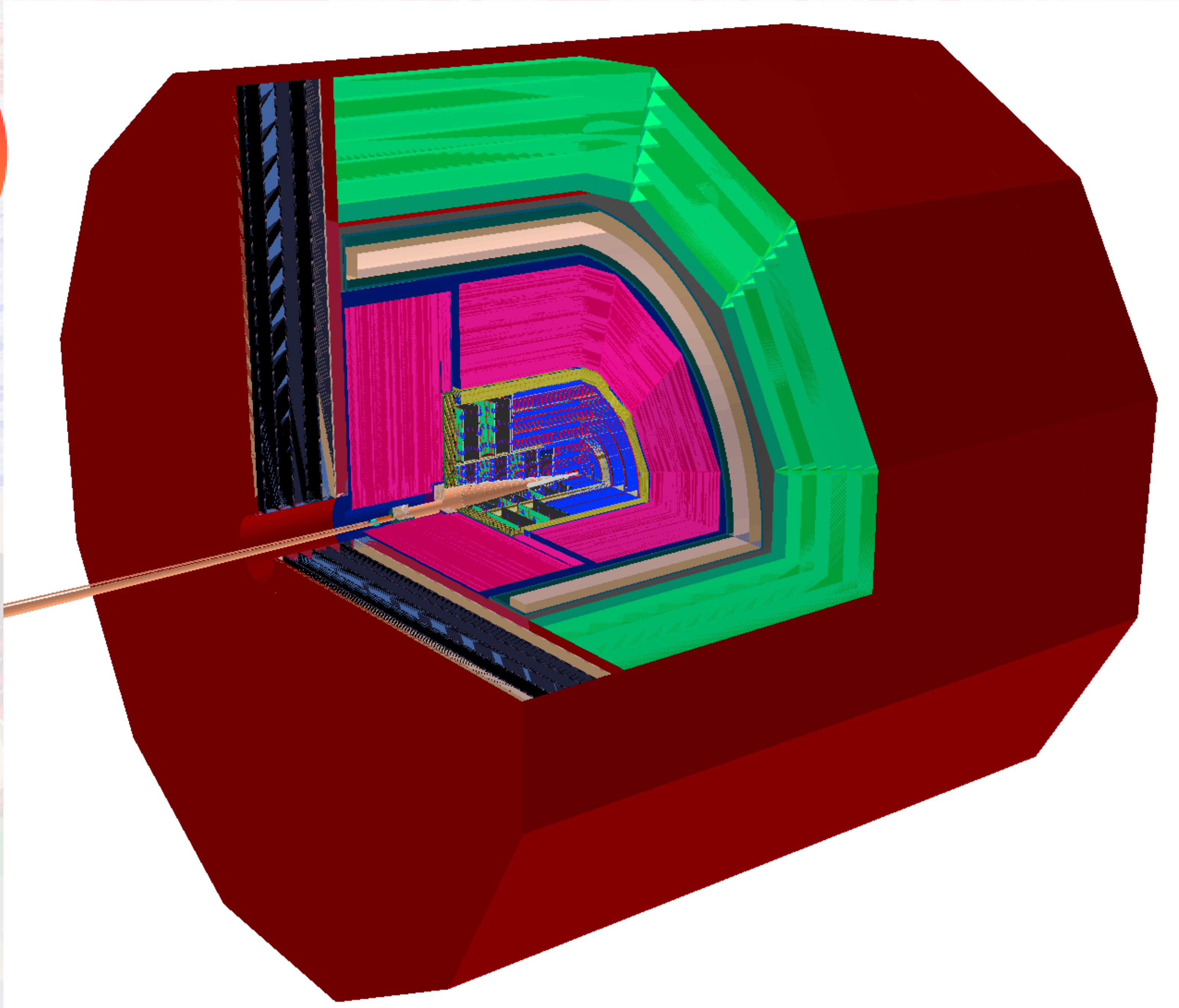
DD4hep

- **Used for design studies (LC, FCC-eh)**
- **Origin from the SiD detector model**
  - **Layer based detectors**
  - **Tracker barrel & endcap**
  - **Several calorimeter constructs**
- **Partially with measurement surfaces (F. Gaede)**
  - **Uses plugin mechanism to enhance detector elements**
    - **Mechanism to attach user defined optional data**

      => Proof that 'anticipate the unforeseen' works
    - **NOT intrusive to detector constructors**

PR: CLICdp

DD4hep

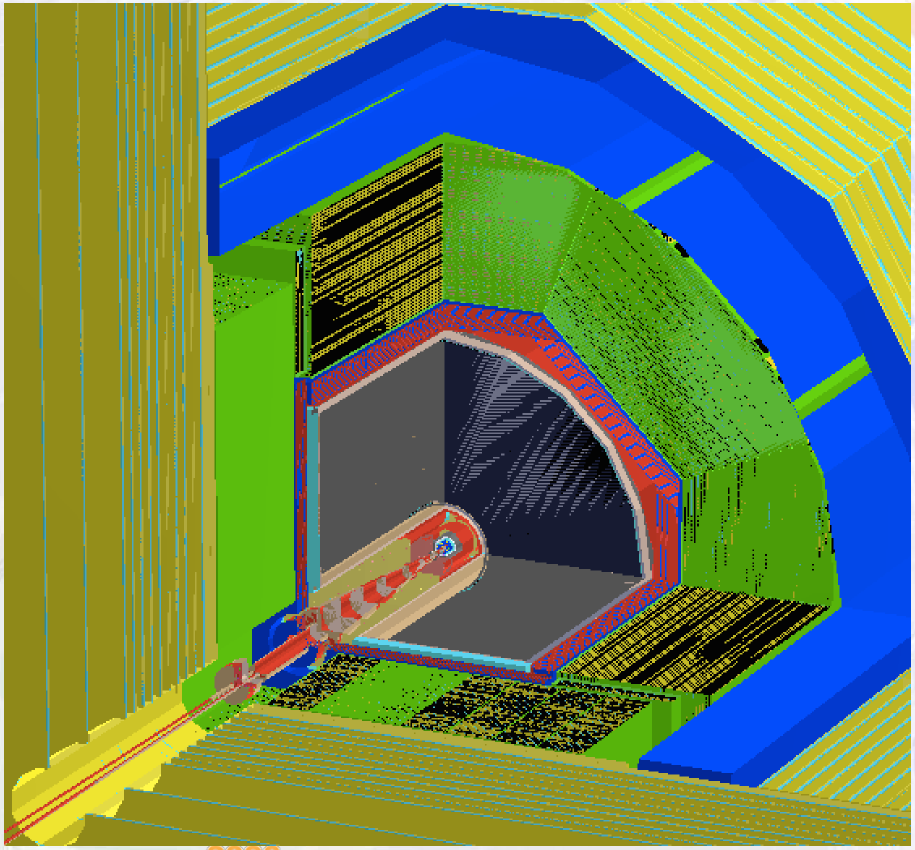


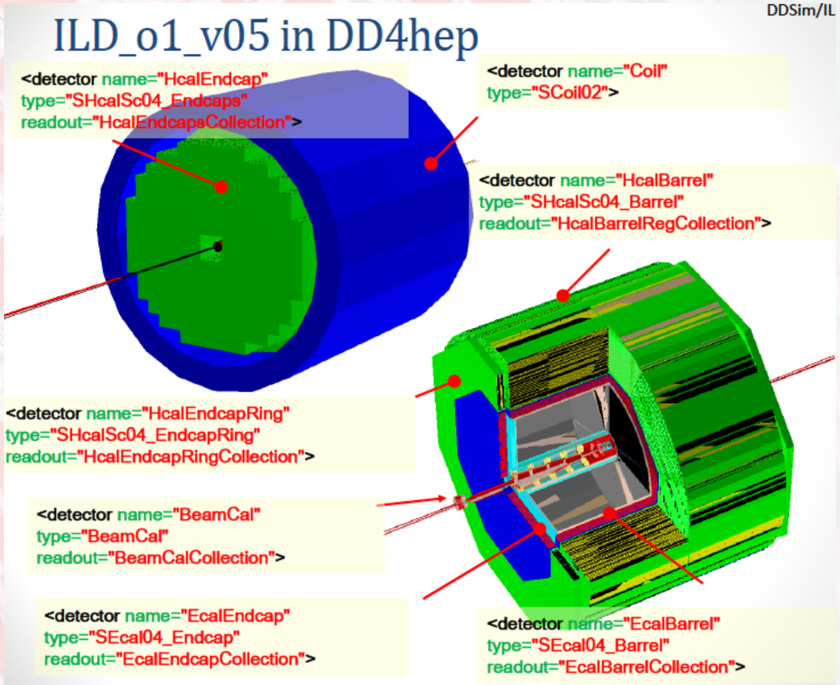
AIDA 2020

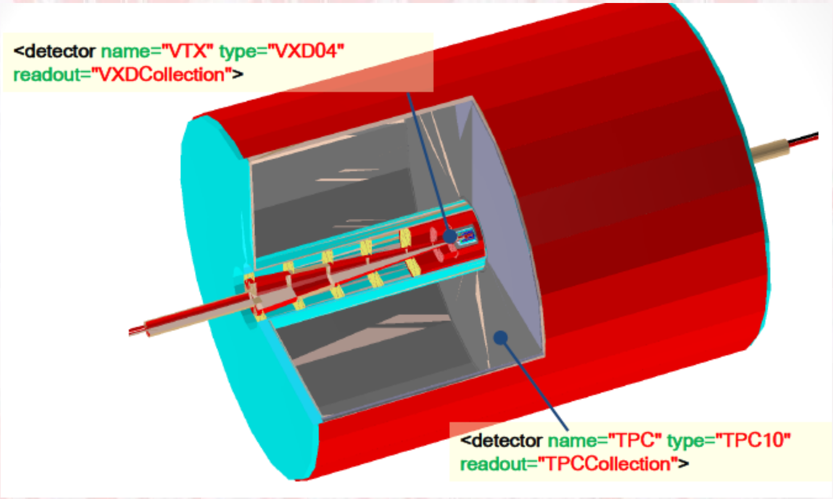
ILD Model ILD_o1_v05

(F.Gaede, L.Shaojun)

DD4hep







DD4hep

- Motivation and Goals
- DD4hep core
- Detector palette for design studies
- **Simulation**
- Conditions and Alignment
- Miscellaneous
- Summary

Simulation: DDG4

DD4hep

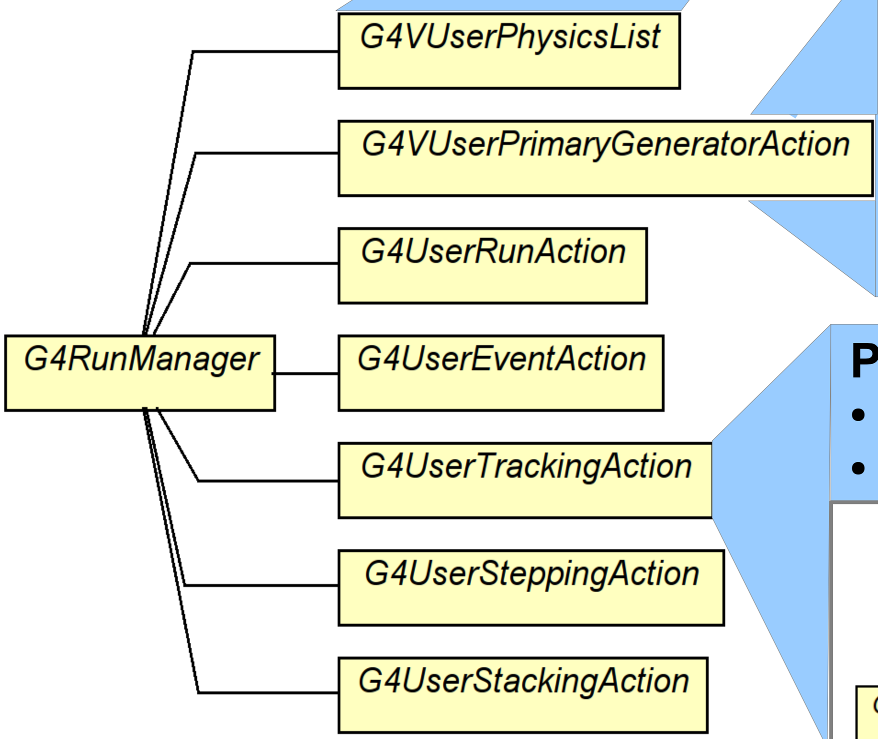
- Simulation = Geometry +
  Detector response +
  Physics
- Mature status
  - Eventual bug fixes, smaller improvements
  - Phase of constant re-validation
- Automatic geometry conversion
- Palette of standard sensitive detectors
- Support for MC truth handling

Geant4 User Hooks

...and what we do inside

DD4hep

Jamboree of plugins: flexible configuration



Flexible definition of the physics list

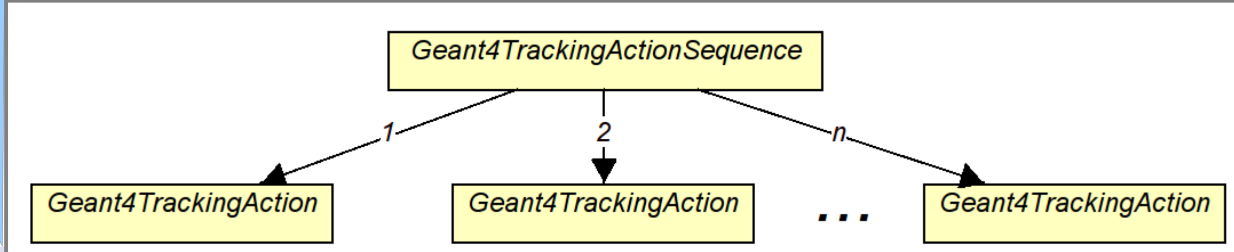
- Define particles, processes, physics constructors or use/extend predefined physics lists

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

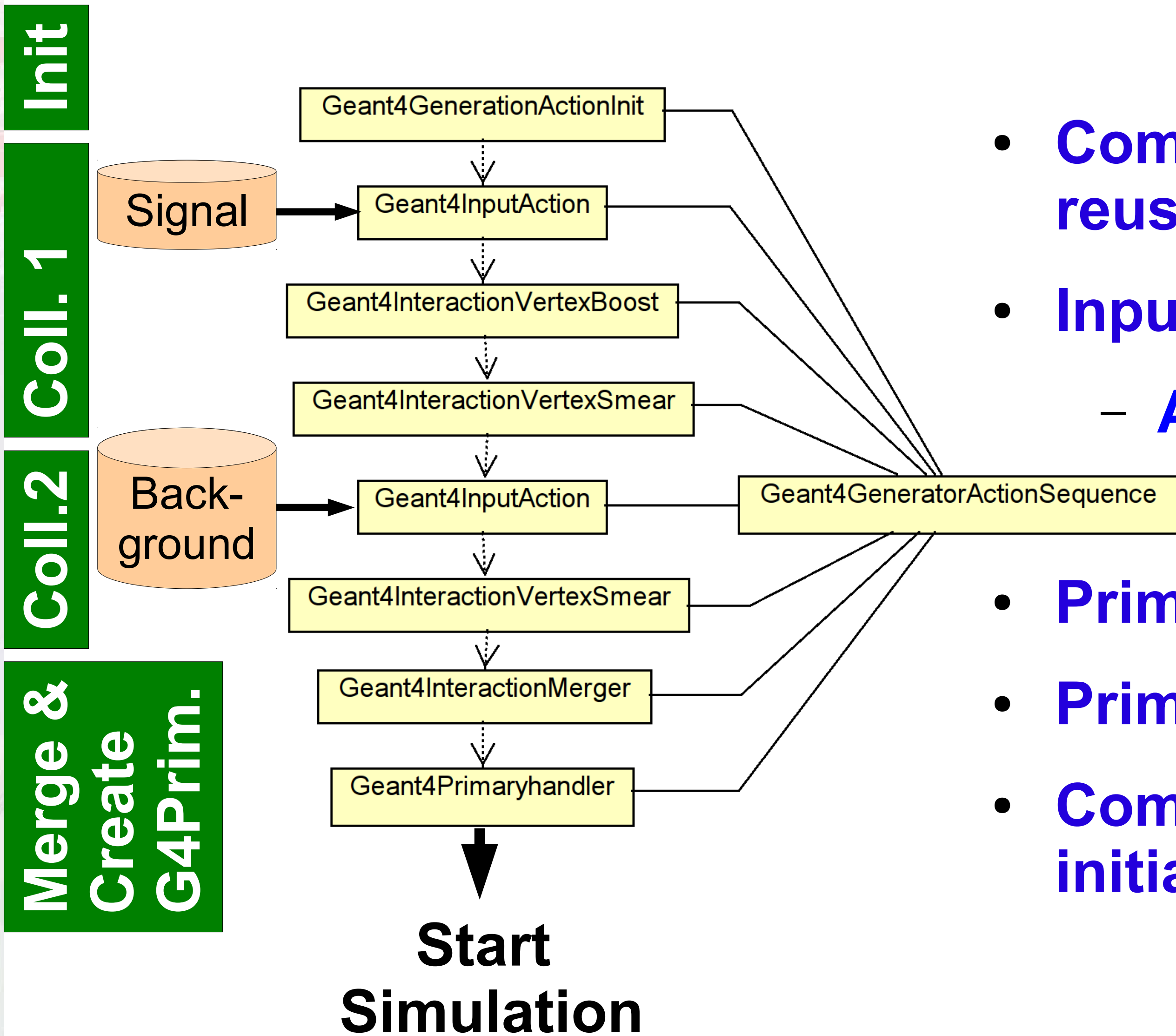
- Either as explicit object type using ABC
- Or registering a member function as callback



Example of an Action Sequence

Event Overlay with Features

DD4hep

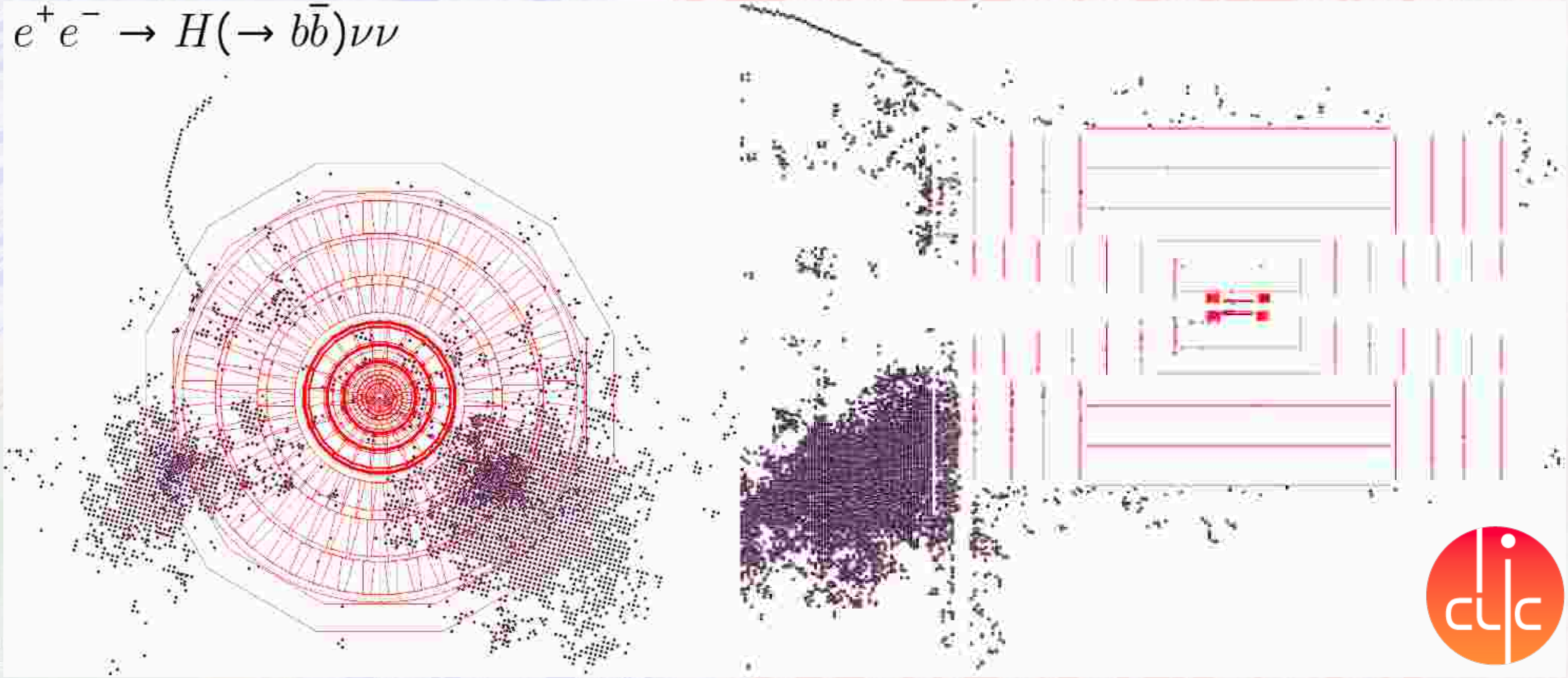


- Combine simple and reusable modules
- Input module
  - Any data format
- Primary vertex smearing
- Primary vertex boost
- Common: initialization, final merge

DDG4 in Production

DD4hep

- **Deployed for CLICdp in DIRAC**
  - **For every detector study (now ~14) central generation**
- **ILC started mass production**



DD4hep

- Motivation and Goals
- DD4hep core
- Detector palette for design studies
- Simulation
- **Conditions and Alignment**
- Miscellaneous
- Summary

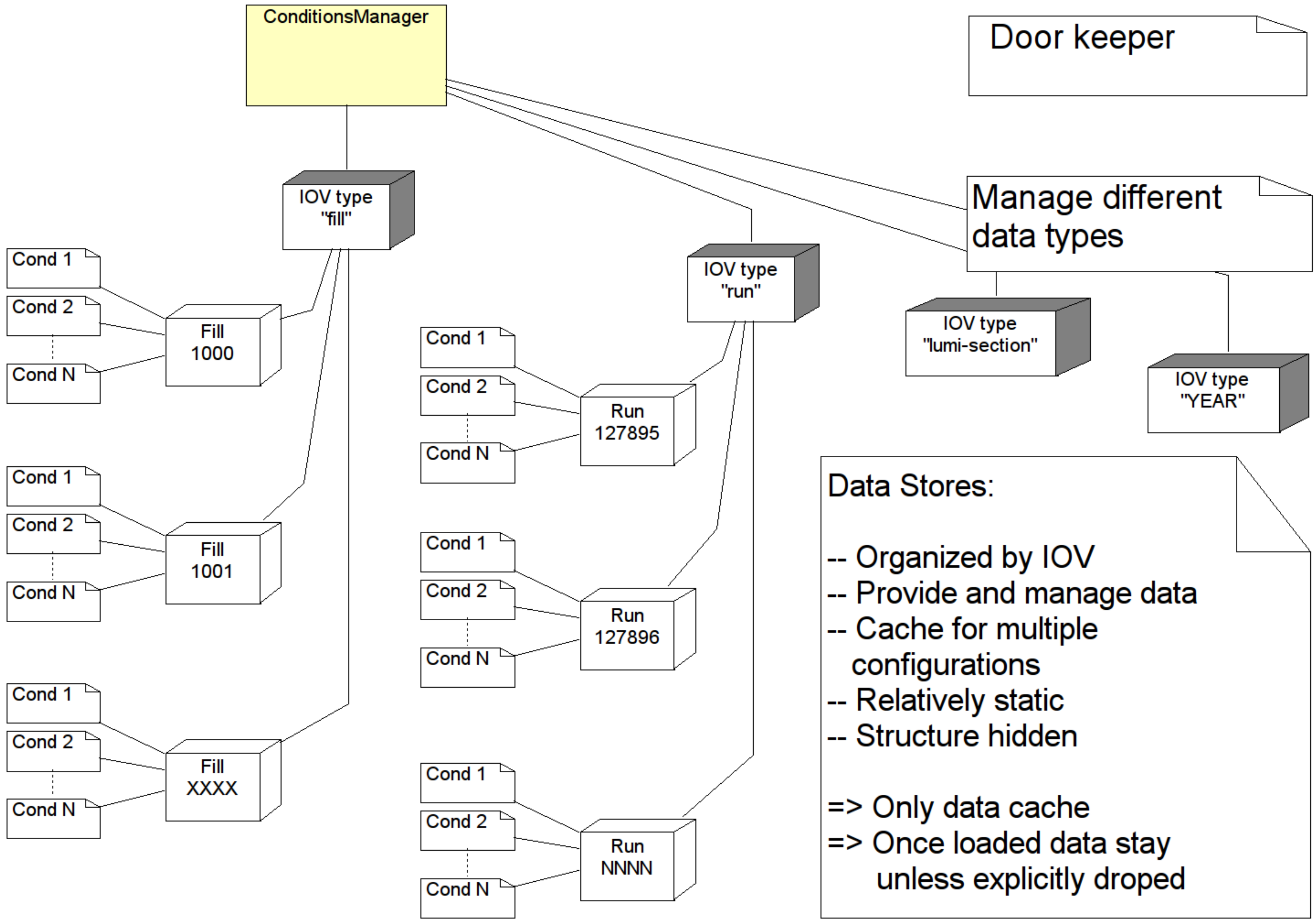
DDCond: Conditions Data

DD4hep

- Time dependent data necessary to process the detector response [of particle collisions]
  - slowly changing: every run O(1h), lumi section O(10min) ...
  - multiple conditions change in batches: require discipline
  - conditions may be the result of computation(s)
- DDCond deals with the management of these data
  - Efficient and fast, if used according to design ideas
  - Manages resources
  - Supports multi threading by design
    Well define locking points
  - Cache where necessary but no more

DDCond: The Data Cache

DD4hep



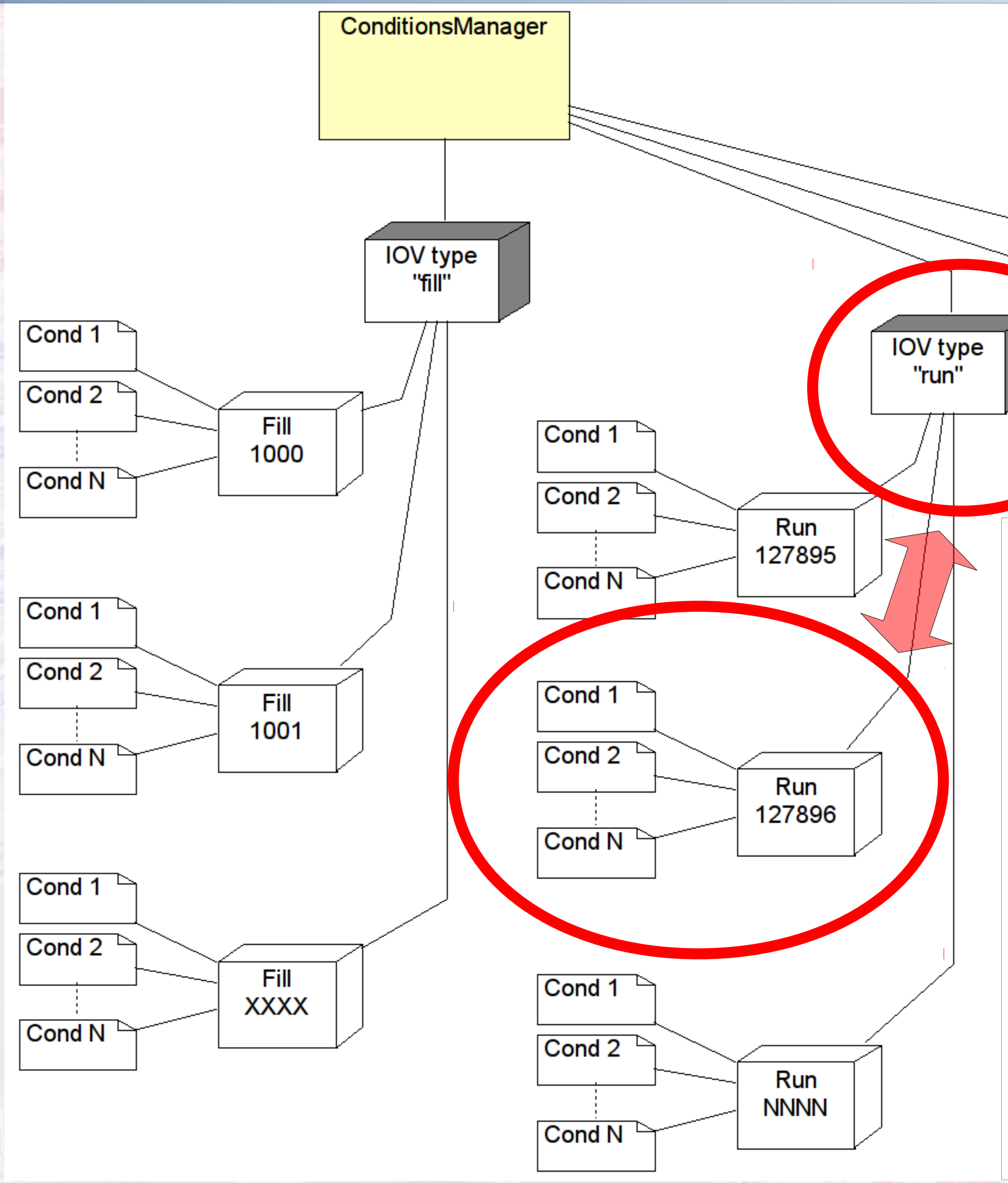
Data Stores:

-- Organized by IOV
-- Provide and manage data
-- Cache for multiple configurations
-- Relatively static
-- Structure hidden

=> Only data cache
=> Once loaded data stay unless explicitly droped

DDCond: The Data Cache

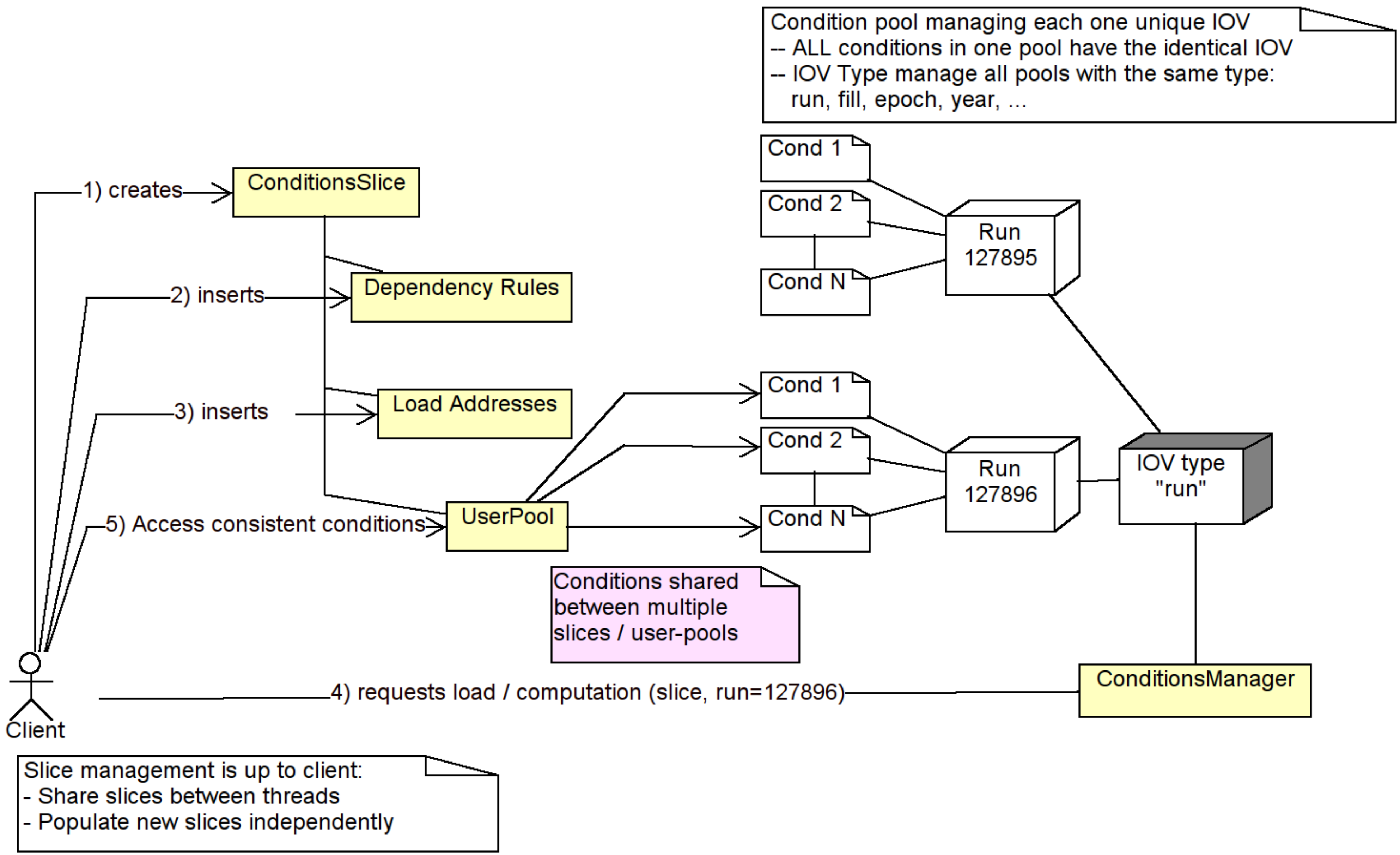
DD4hep



- Conditions sub-pools are read-only (gentleman's agreement)
- Only when sub-pools enter or leave the scene the container must be locked
- MT Detail: sub-pools referenced by shared_ptr: => usable beyond registration

DDCond: IOV Data Projection

DD4hep



DDCond: Status

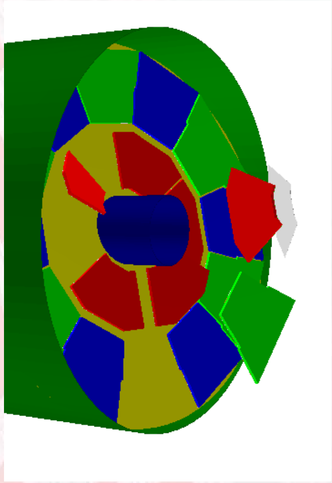
DD4hep

- Described functionality tested with LHCb conditions data
- Alignment support to handle geometry imperfections
  - Local Alignments are derived conditions
  - Convert Δ parameters
    (translation, rotation, pivot-point)
    to transformations to world or reference point

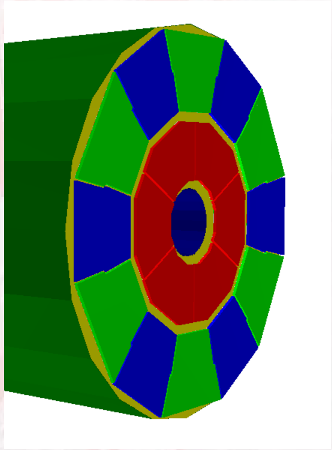
Global and Local Alignments

DD4hep

- **Global alignment corrections**
  - Physically alters geometry
    Intrinsically supported by ROOT
  - By construction not multi-threaded
  - Possibility to simulate misaligned geometries



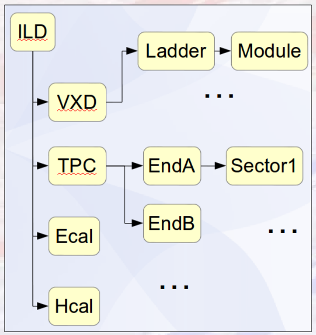
- **Local alignment corrections**
  - Geometry stays intact (either ideal or globally aligned)
  - Multi-threading supported, multiple versions
  - Local alignment corrections are conditions
  - Provide matrices from ideal geometry to world
    e.g. to adjust hit positions

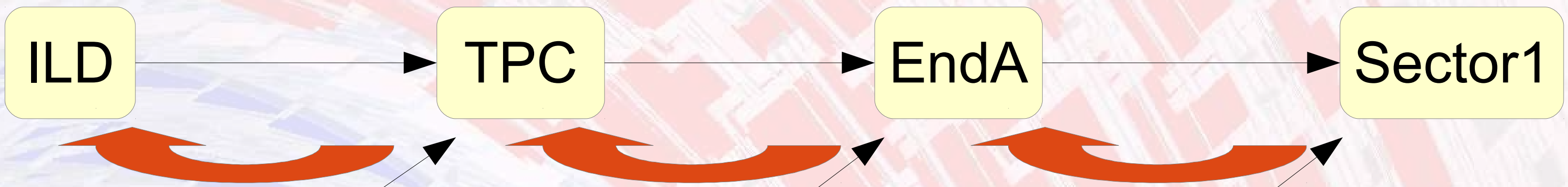


- **Both supported**

Local Alignment Δ - Parameters

DD4hep





$$Tr^{World}_{Sec\,1} = Tr^{World}_{EndA} \times \left(Tr^{Parent(EndA)}_{Sec\,1} + \Delta_{Sec\,1} \right)$$

$$Tr^{World}_{EndA} = Tr^{World}_{TPC} \times \left(Tr^{Parent(TPC)}_{EndA} + \Delta_{EndA} \right)$$

$$Tr^{World}_{TPC} = Tr^{World}_{ILD} \times \left(Tr^{Parent(ILD)}_{TPC} + \Delta_{TPC} \right)$$

- Trickle-up the hierarchy and compute the matrices the most effective way with re-use of intermediate results
- Math verified by C. Burr

DD4hep

- Motivation and Goals
- DD4hep core
- Detector palette for design studies
- Simulation
- Conditions and Alignment
- **Miscellaneous**
- Summary

Toolkit Users

DD4hep

Increasing interest in the HEP community

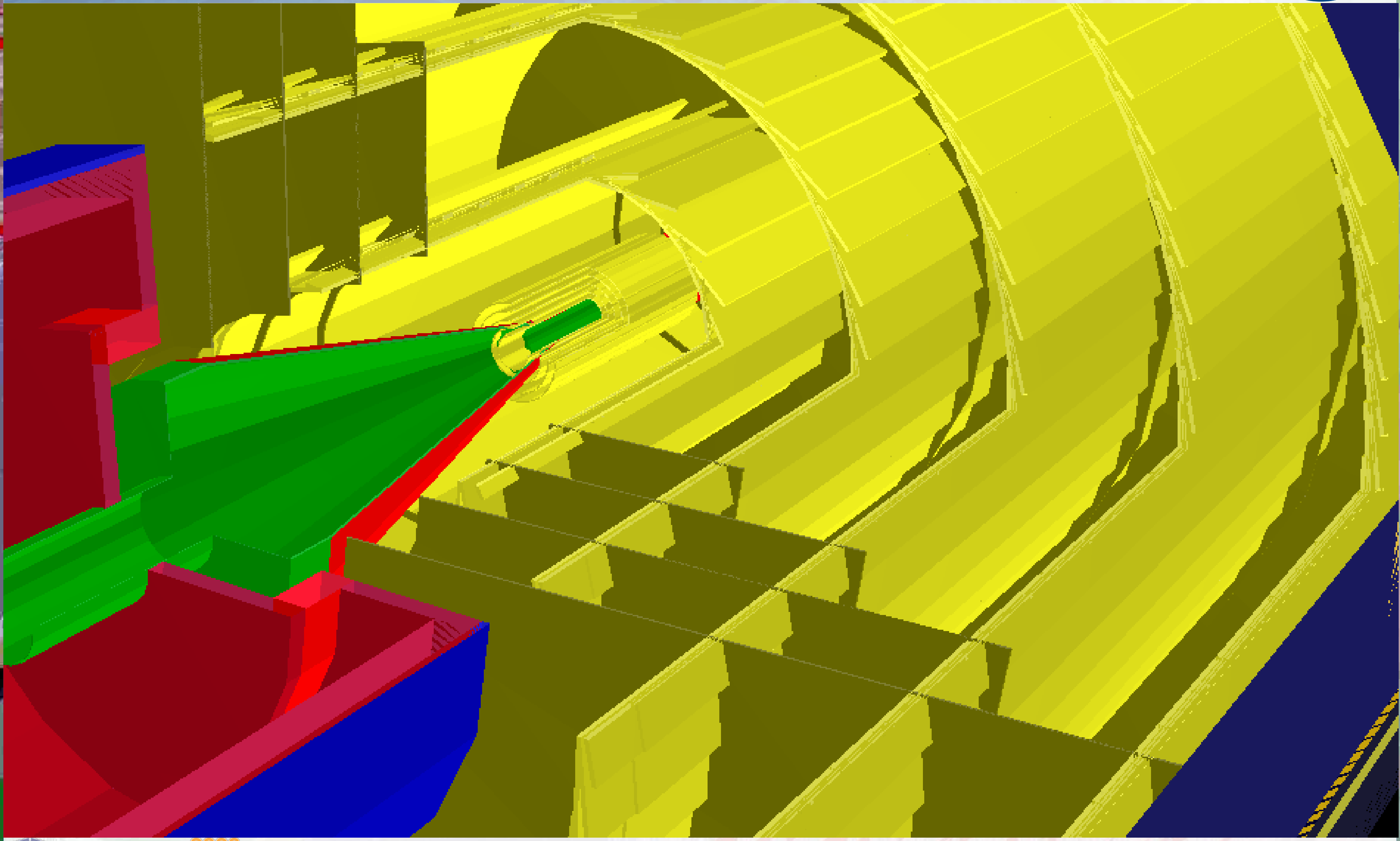
ILC	F. Gaede et al.
CLICdp	A. Sailer et al.
SiD	W. Armstrong
FCC-eh	P. Kostka et al.
FCC-hh	A. Salzburger et al.
FCC-ee	O. Viazlo (CLD design), N. Alipour, G. Voutsinas
CMS	Evaluation for upgrade started (2019) (Y.Osborne et al.)
LHCb	Evaluation for upgrade started (2019) (B.Couturier et al.)
CALICE	Calorimeter R&D, started
EIC	Evaluation started

Summary

DD4hep

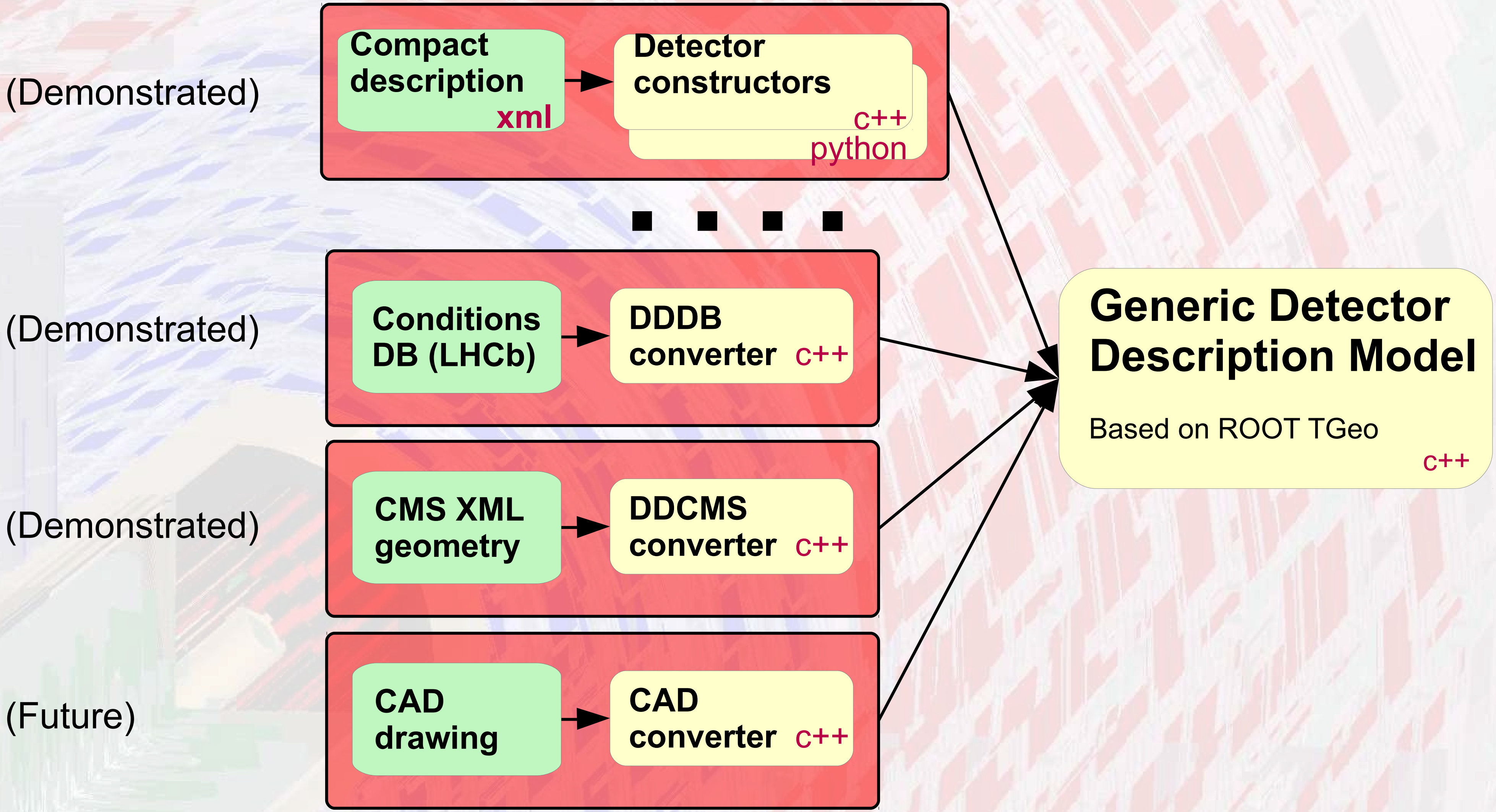
- DD4hep is getting mature
- Starts being capable of handling all aspects of detector description for the lifetime of an experiment
- Increasing interest in the community and increasing number of users
- Visit us on:
  - http://dd4hep.cern.ch
  - Up to date doxygen information
  - User Manuals: have improved but not perfect

Questions and Answers



Multiple Input Sources

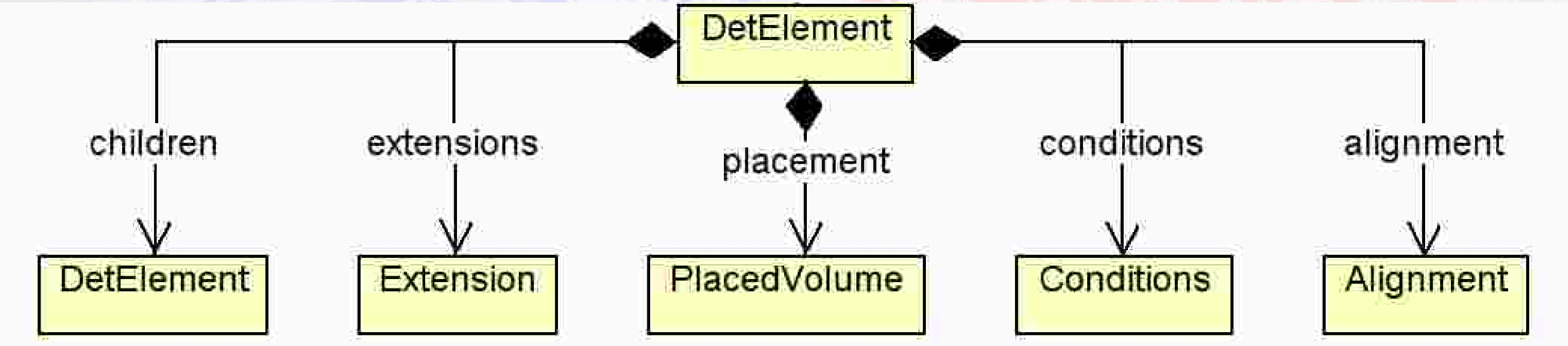
DD4hep



Get Fingers Dirty
LHCb Velo Detector

DD4hep

- People want to see "Detector elements"
  - Fully functional description of parts of the detector
    - Long term valid stuff (structure)
    - Short lived quantities (temperature, alignment, …)
- A "natural" aggregation would be similar to:

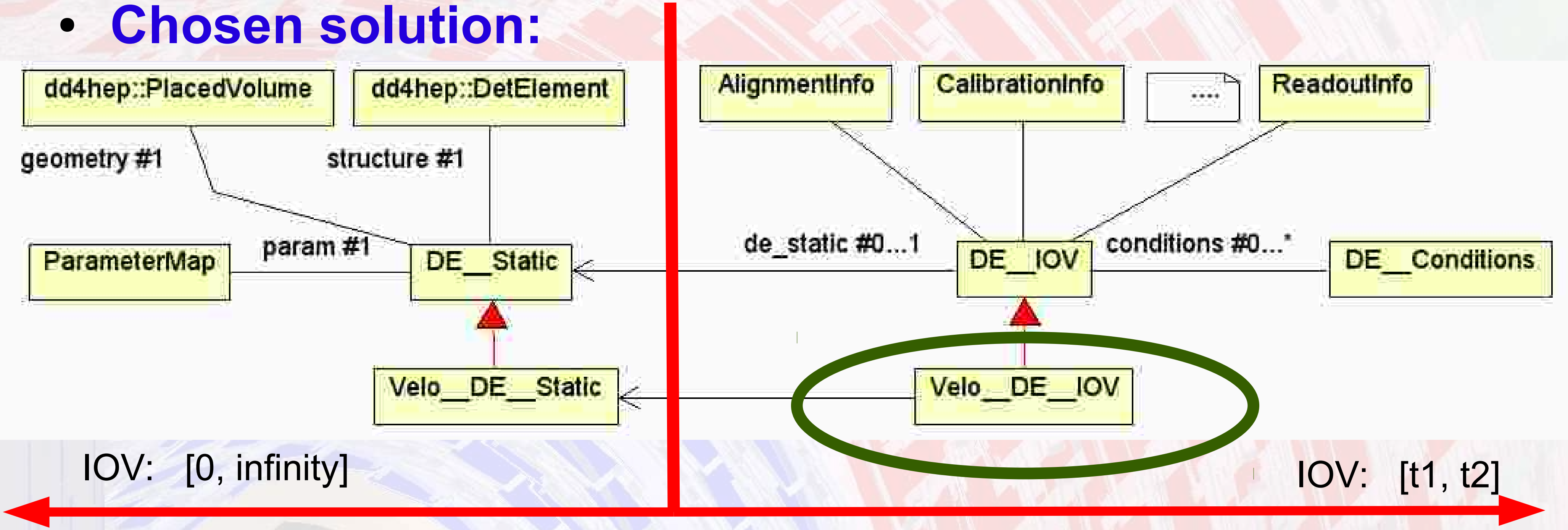


- Intuitive, but not good: violates multi-threading

Real World Use Case
LHCb Velo Detector

DD4hep

- **Chosen solution:**



- **Use IOV dependent projection for event processing**
  - **This is our new "detector element"**
  - **Keeps reference to the not changing properties**
- **Dress with facade to provide required functionality(ies)**

Real World Use Case
LHCb Velo Detector

DD4hep

- Structures are build using the derived conditions callback mechanism
  - Static part: once only
  - IOV dependent part: when not in pools
    Also fills link to static information
- Since conditions in existing pools still can be shared while preparing new IOV depending conditions
  - No locking strategy necessary
- Alignment computation incorporated
  - Reminder: alignments must be computed ‘en block’
    for an efficient computation
- Common activity with WP3 Task 3.3 (C.Burr et al.)