FCC Physics Software

Juraj Smieško (CERN) as part of the FCC Software team

Future Colliders for Early-Career Researchers: CZ/SK Edition

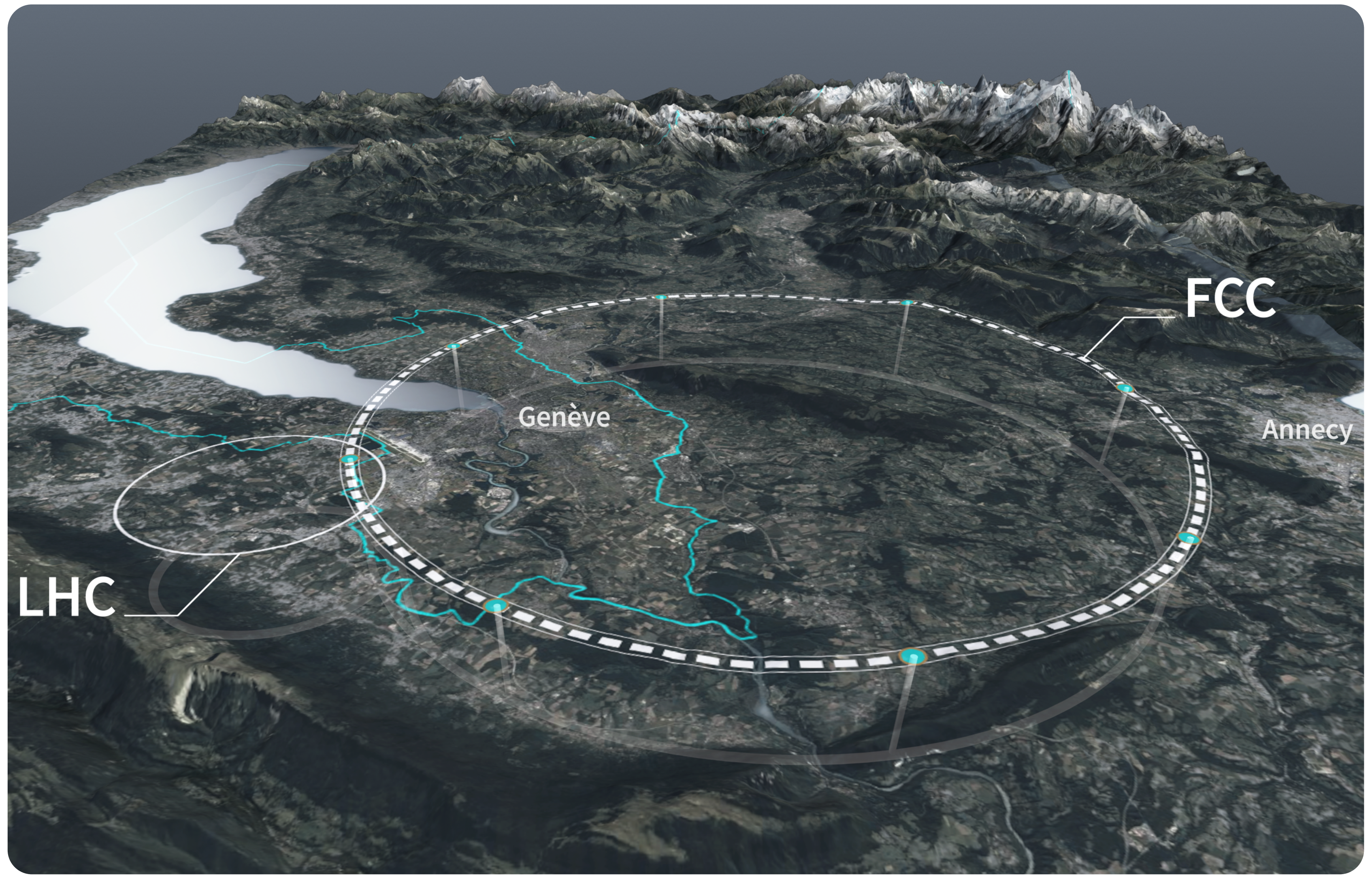
Prague, CZ

27 September 2024

Future Circular Collider

Energy and luminosity upgrade in an integrated program

- FCC-ee (Z, WW, H, ttbar):
  Highest luminosities at Z, W, ZH among proposed Higgs and EW factories with indirect discovery potential up to ~ 70 TeV
- FCC-hh (~100 TeV):
  Direct exploration of next energy frontier (~ x10 LHC) and unparalleled measurements
- Feasibility Status Report in 2025
- More than 150 institutes from 30 countries already involved



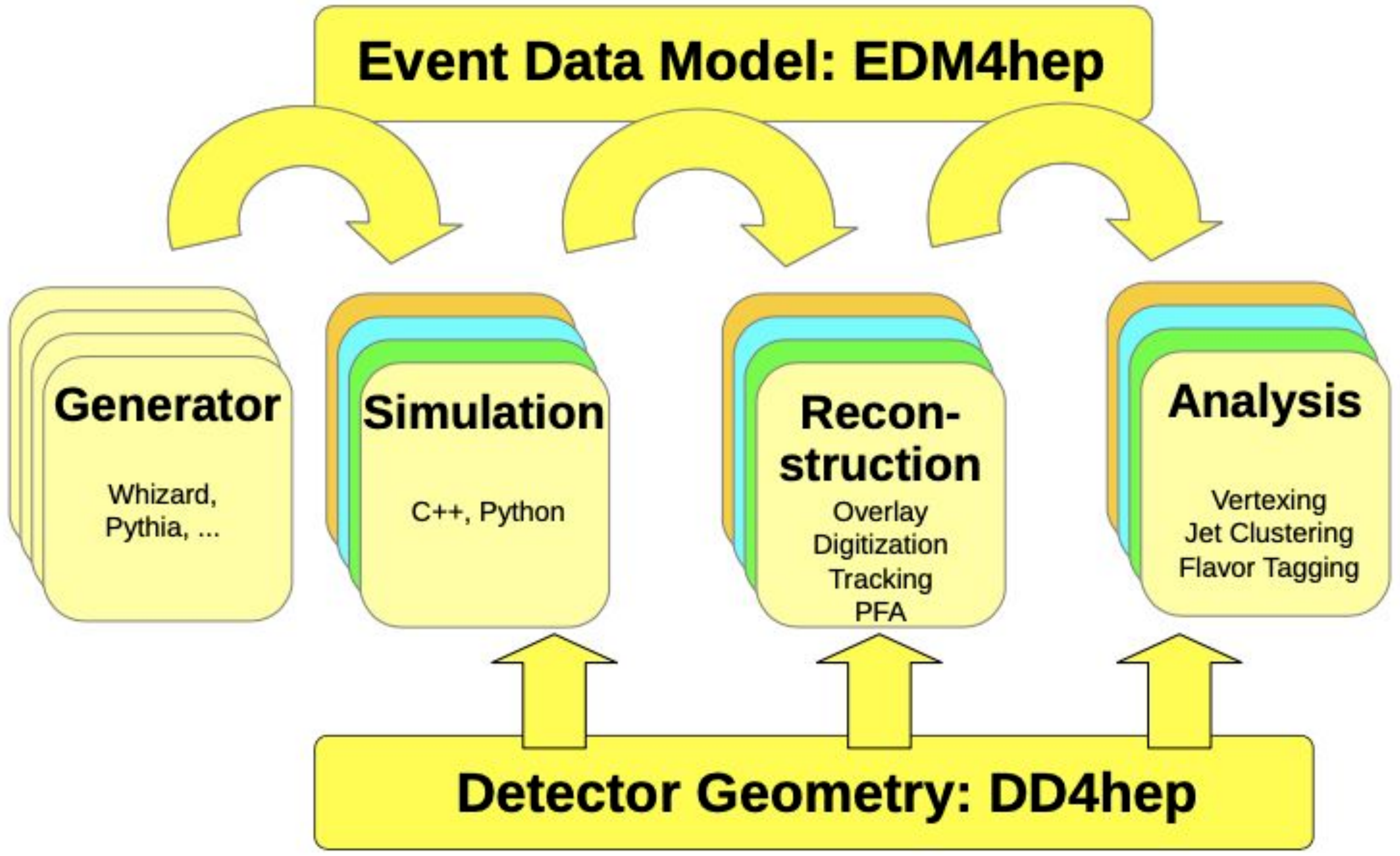
source: FCC Layout — Aerial View

Ingredients of FCC Physics Software

Key4hep

Set of common software packages, tools, and standards for different Detector/Collider Concepts

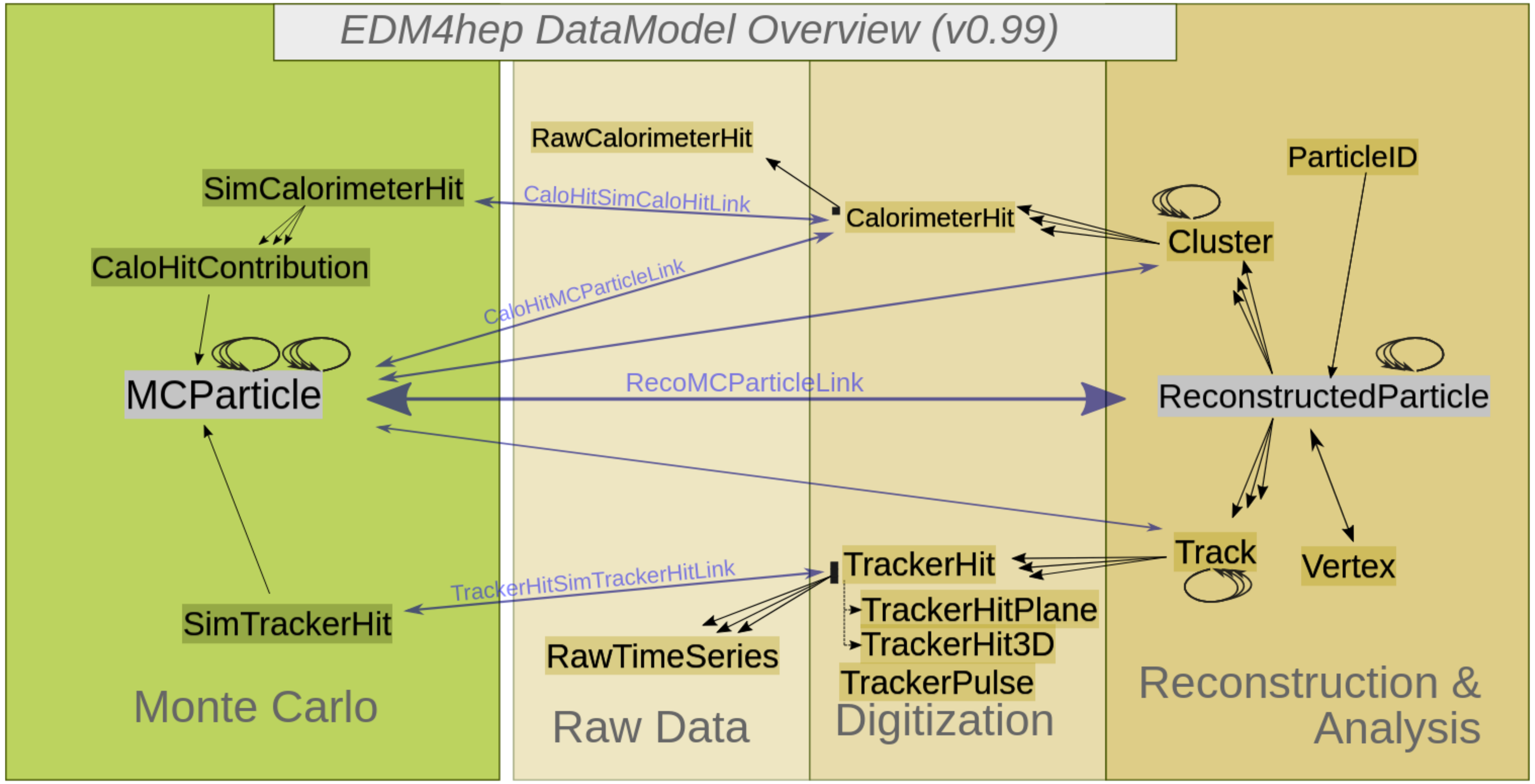
- Common effort from FCC, CLIC/ILC, EIC, CEPC, ...
  - Preserves and adapts existing functionanlity from iLCSoft, FCCSW, CEPCSW, ...
- Individual participants adjust their stack to their needs
- Main ingredients:
  - Event data model: EDM4hep
  - Data processing framework: Gaudi
  - Detector description: DD4hep
  - Software distribution: Spack
- Bi-weekly meetings
  - Tuesday, 9:00 AM GVA; Indico category



EDM4hep I.

Describes event data with the set of standard objects

- Specification in a single YAML file
- Generated with the help of Podio



EDM4hep II.

Example object:

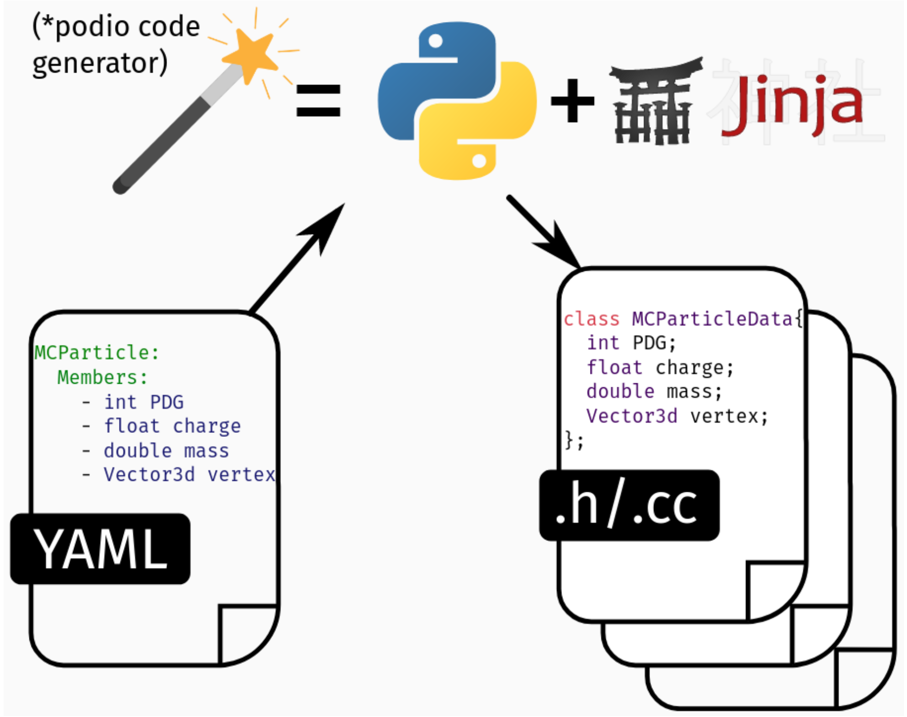
```
#-------------  CalorimeterHit
edm4hep::CalorimeterHit:
  Description: "Calorimeter hit"
  Author: "EDM4hep authors"
  Members:
    - uint64_t cellID                 // detector specific (geometrical) cell id
    - float energy [GeV]              // energy of the hit
    - float energyError [GeV]         // error of the hit energy
    - float time [ns]                 // time of the hit
    - edm4hep::Vector3f position [mm] // position of the hit in world coordinates
    - int32_t type                    // type of hit
```

- Current version: `v0.99.0`
  - Approaching version 1.0
  - Backward compatibility
- Objects can be extended / new created
- Bi-weekly discussion:
  - Tuesday, 9:00 AM GVA; Indico

Podio

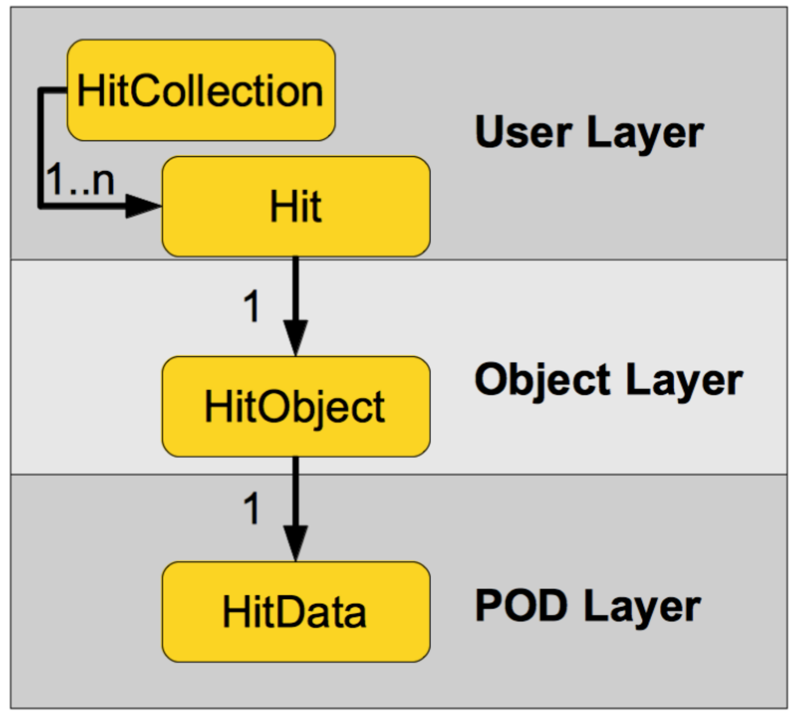
Generates Event Data Model and serves as I/O Layer

- Generates EDM from YAML files
- Employs plain-old-data (POD) data structures
- I/O machinery consists of three layers
  - POD Layer - actual data structures
  - Object Layer - helps resolve the relations
  - User Layer - full fledged EDM objects
- Supports multiple backends:
  - ROOT, SIO, ...
- Current version: `1.0.1`



```
MCParticle:
  Members:
    - int PDG
    - float charge
    - double mass
    - Vector3d vertex
```

```
class MCParticleData{
  int PDG;
  float charge;
  double mass;
  Vector3d vertex;
};
```



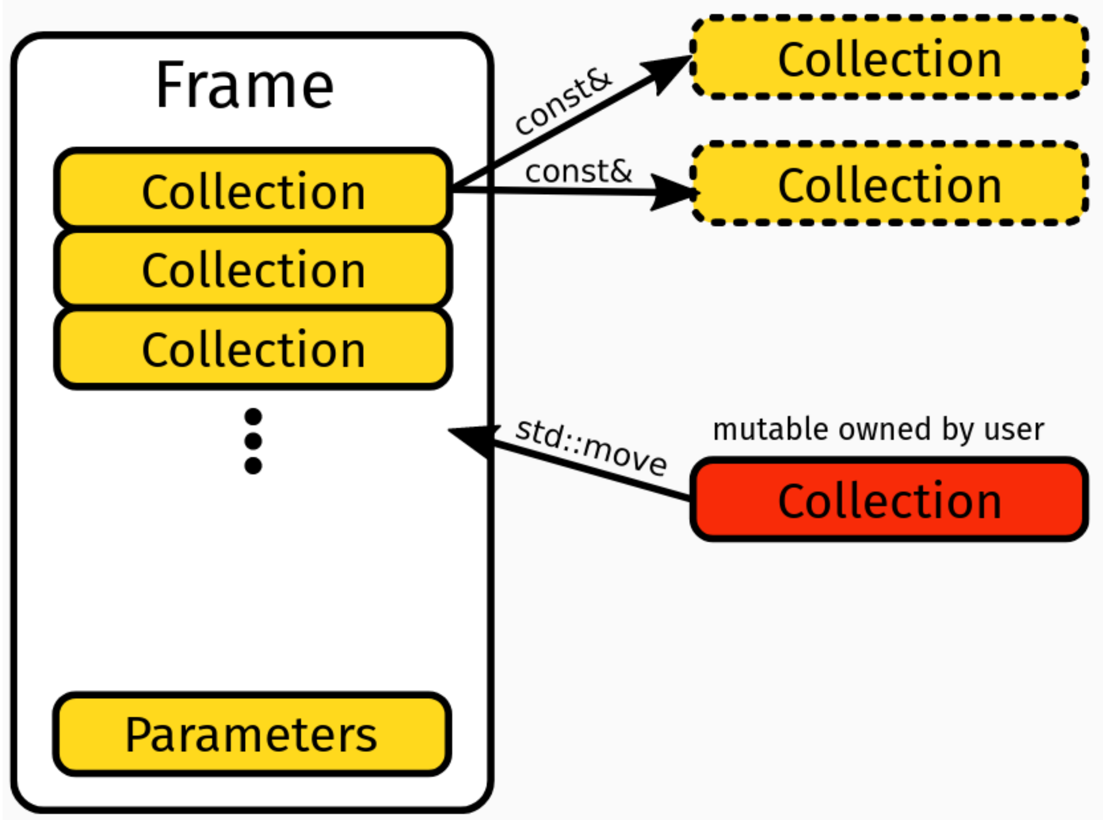
Podio Reader

Constructs the EDM4hep objects for the user

Example usage of Podio Reader in Pyhton:

```
from podio.root_io import Reader
reader = Reader("one or many input files")
for event in reader.get("events"):
  hits = store.get("hits")
  for hit in hits:
    # ...
```

To inspect contents of the EDM4hep file use: `podio-dump`



Gaudi

Battle tested event processing framework

- Job of an event processing framework
  - Stitches and steers various algorithms together
  - Controls event loop
  - Manages transient storage and I/O
- Used by current experiments: ATLAS, LHCb
- New developments: `Gaudi::Functional`
- Key4hep started life by attempting to reuse algorithms already developed
- Need for converters/wrappers:
  k4MarlinWrapper, k4CLUE, k4GaudiPandora, ...
- Selected over Marlin due to MT support

Hello World in Gaudi:

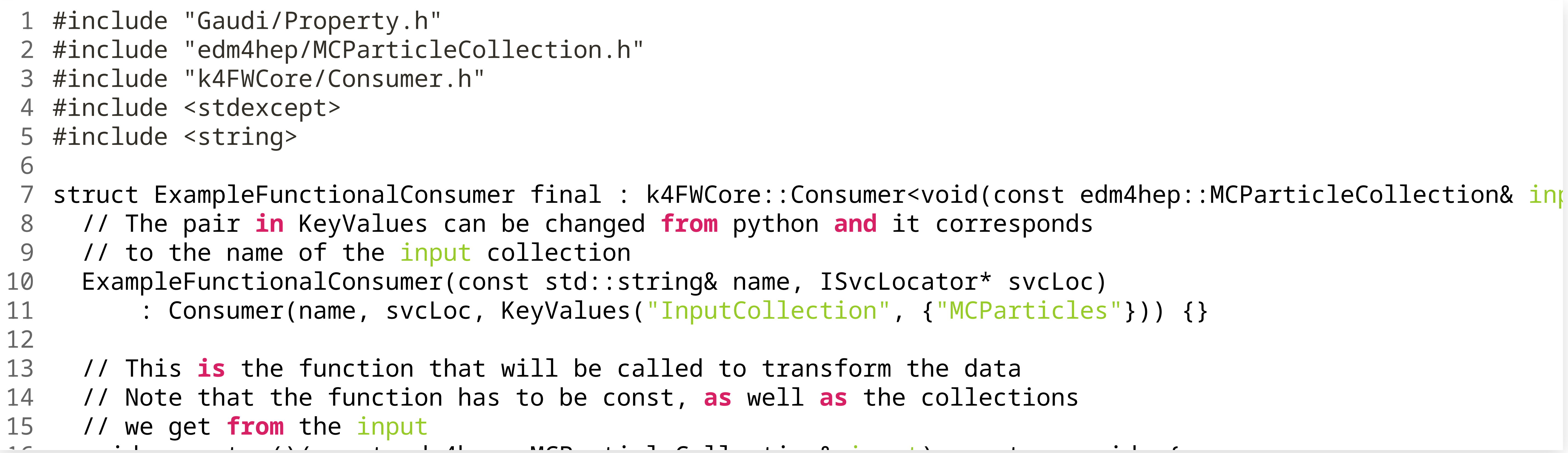
```python
from Gaudi.Configuration import *
from Configurables import HelloWorldEx

alg = HelloWorldEx()

ApplicationMgr(
    EvtMax = 10,
    EvtSel = 'NONE',
    HistogramPersistency = 'NONE',
    TopAlg = [alg],
)
```

Source: Gaudi

k4FWCore

Package with Key4hep Gaudi components

- Provides input and output of files, but also among algorithms
  - `IOSvc`, `DataHandle`, `MetaDataHandle`
- Main program to run Gaudi steering: `k4run`
- Gaudi Functional allows proper multithreading

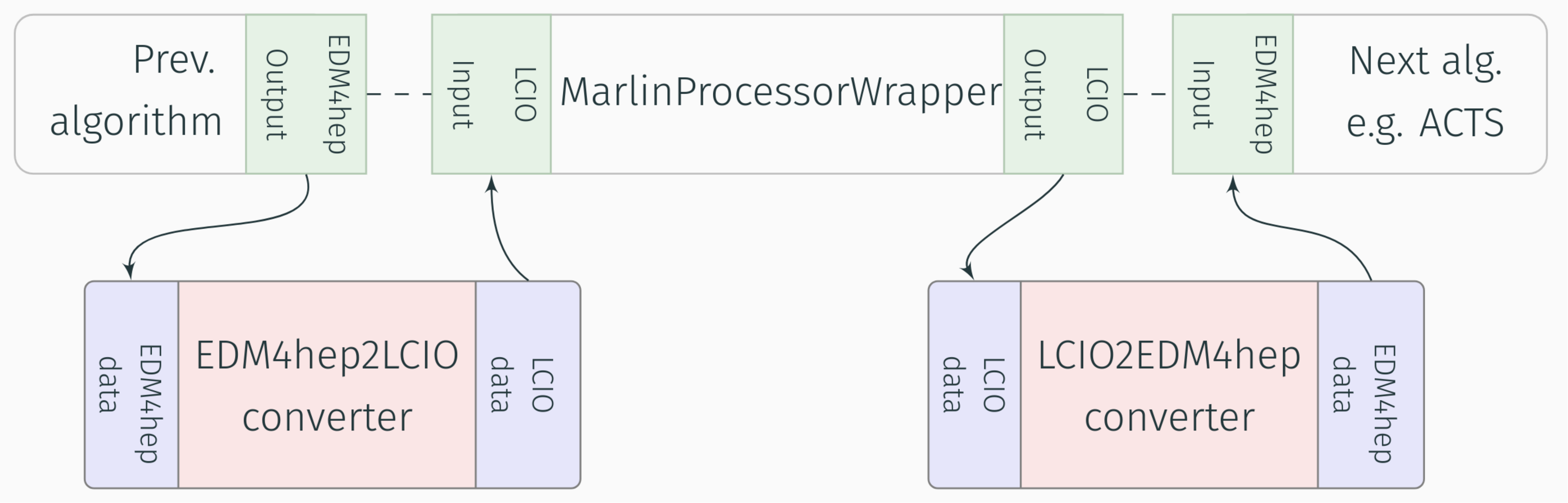


```
#include "Gaudi/Property.h"
#include "edm4hep/MCParticleCollection.h"
#include "k4FWCore/Consumer.h"
#include <stdexcept>
#include <string>

struct ExampleFunctionalConsumer final : k4FWCore::Consumer<void(const edm4hep::MCParticleCollection& inp
  // The pair in KeyValues can be changed from python and it corresponds
  // to the name of the input collection
  ExampleFunctionalConsumer(const std::string& name, ISvcLocator* svcLoc)
      : Consumer(name, svcLoc, KeyValues("InputCollection", {"MCParticles"})) {}

  // This is the function that will be called to transform the data
  // Note that the function has to be const, as well as the collections
  // we get from the input
```

LCIO ↔ EDM4hep Converters

Integration of tools developed by linear collider community

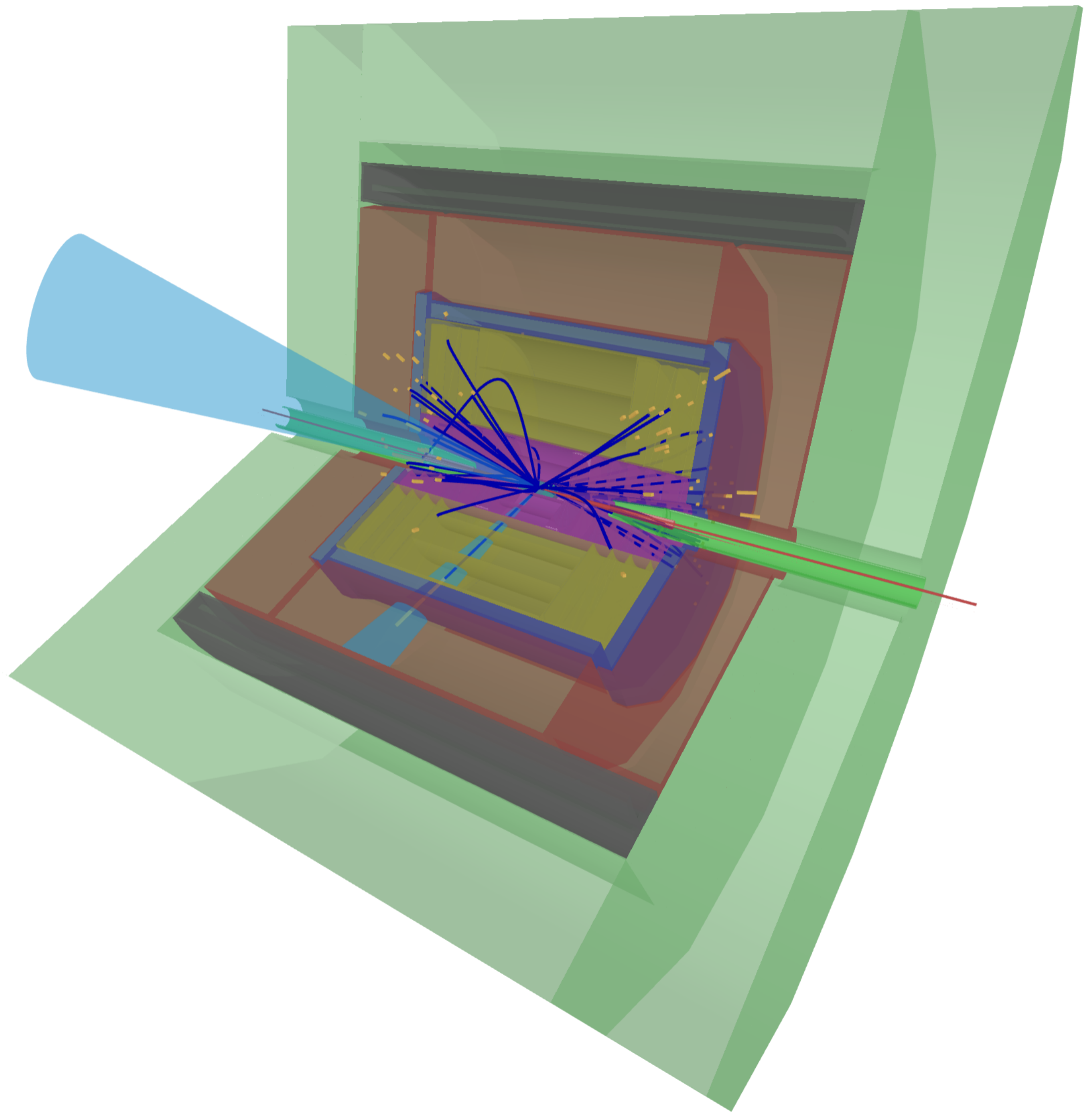
- k4MarlinWrapper wraps Marlin processor in a Gaudi algorithm and allows to run them unchanged
- LCIO ↔ EDM4hep converters do the conversion on the fly

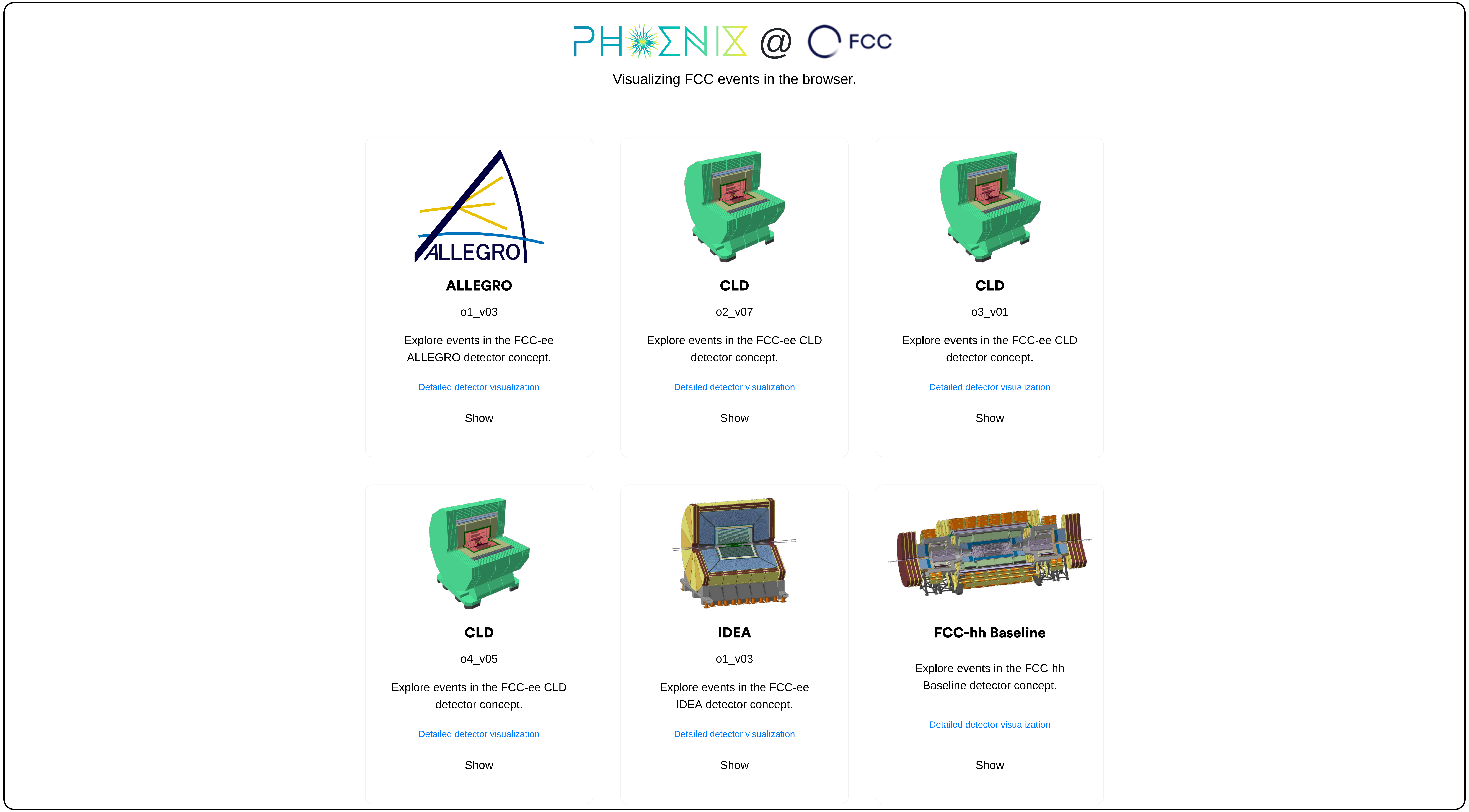


Detector description

The detector is completely described with the help of DD4hep

- The description itself done by C++ builder and XML compact file(s)
  - Every sub-detector needs specialized C++ builder class
  - The XML compact files are organized in tree structure, which allows Plug-and-Play
- XML Schema defined by LCSim
- Specialized data can be attached to each sub-detector at runtime
- Simulation for FCC-ee done with `ddsim` — standalone simulation executable
- All FCC-ee (sub)detectors collected in k4geo repository in `/FCCee`
- FCC-hh baseline detector stayed in FCCDetectors repository





PHOENIX @ FCC

Visualizing FCC events in the browser.

ALLEGRO

o1_v03

Explore events in the FCC-ee ALLEGRO detector concept.

Detailed detector visualization

Show

CLD

o2_v07

Explore events in the FCC-ee CLD detector concept.

Detailed detector visualization

Show

CLD

o3_v01

Explore events in the FCC-ee CLD detector concept.

Detailed detector visualization

Show

CLD

o4_v05

Explore events in the FCC-ee CLD detector concept.

Detailed detector visualization

Show

IDEA

o1_v03

Explore events in the FCC-ee IDEA detector concept.

Detailed detector visualization

Show

FCC-hh Baseline

Explore events in the FCC-hh Baseline detector concept.

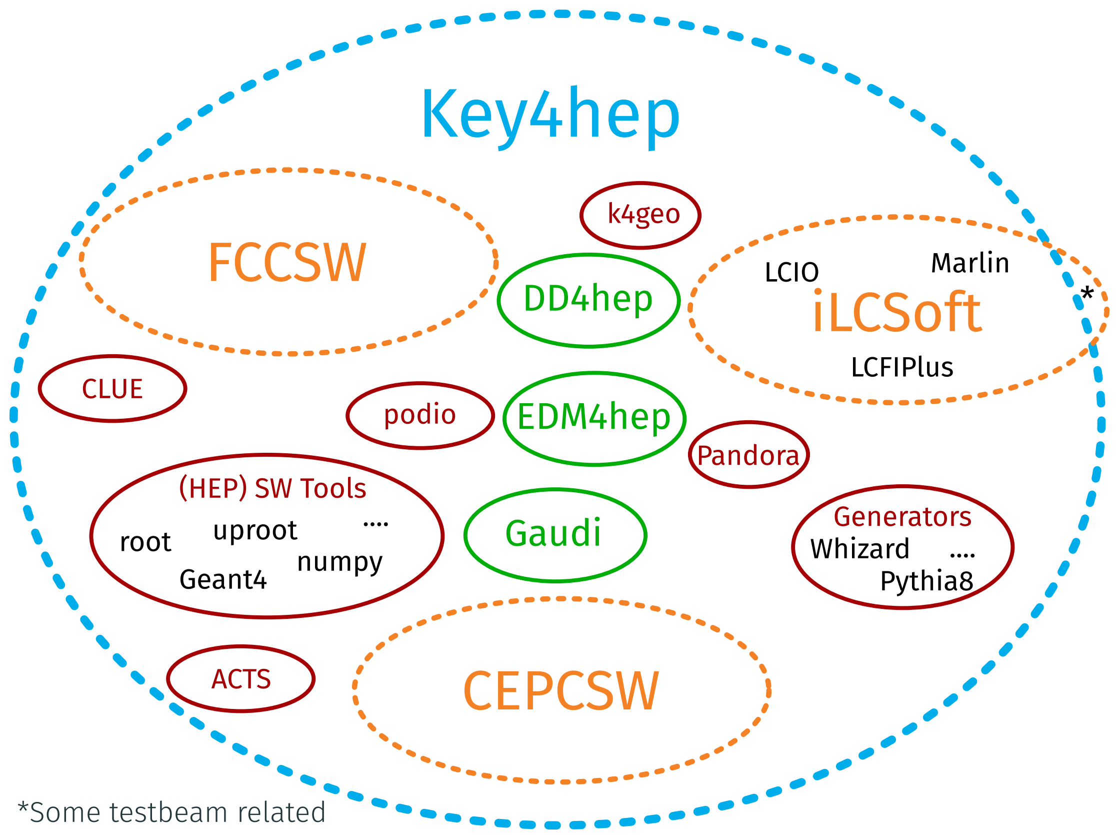
Detailed detector visualization

Show

Spack in Key4hep

Package management for supercomputing centers

- Distributes software in source form
- Every package can have multiple versions and configuration options
- Strives to not depend on the underlying OS as much as possible
- Peace of software is packaged by creating a recipe script
- The packages are stored in two repositories
  - Main Spack repository
  - Specialized Key4hep repository
- Compiled packages are published on CVMFS
  - `source /cvmfs/sw.hsf.org/key4hep/setup.sh`
  - `source /cvmfs/sw-nightlies.hsf.org/key4hep/setup.sh`
  - `source /cvmfs/fcc.cern.ch/sw/latest/setup.sh`

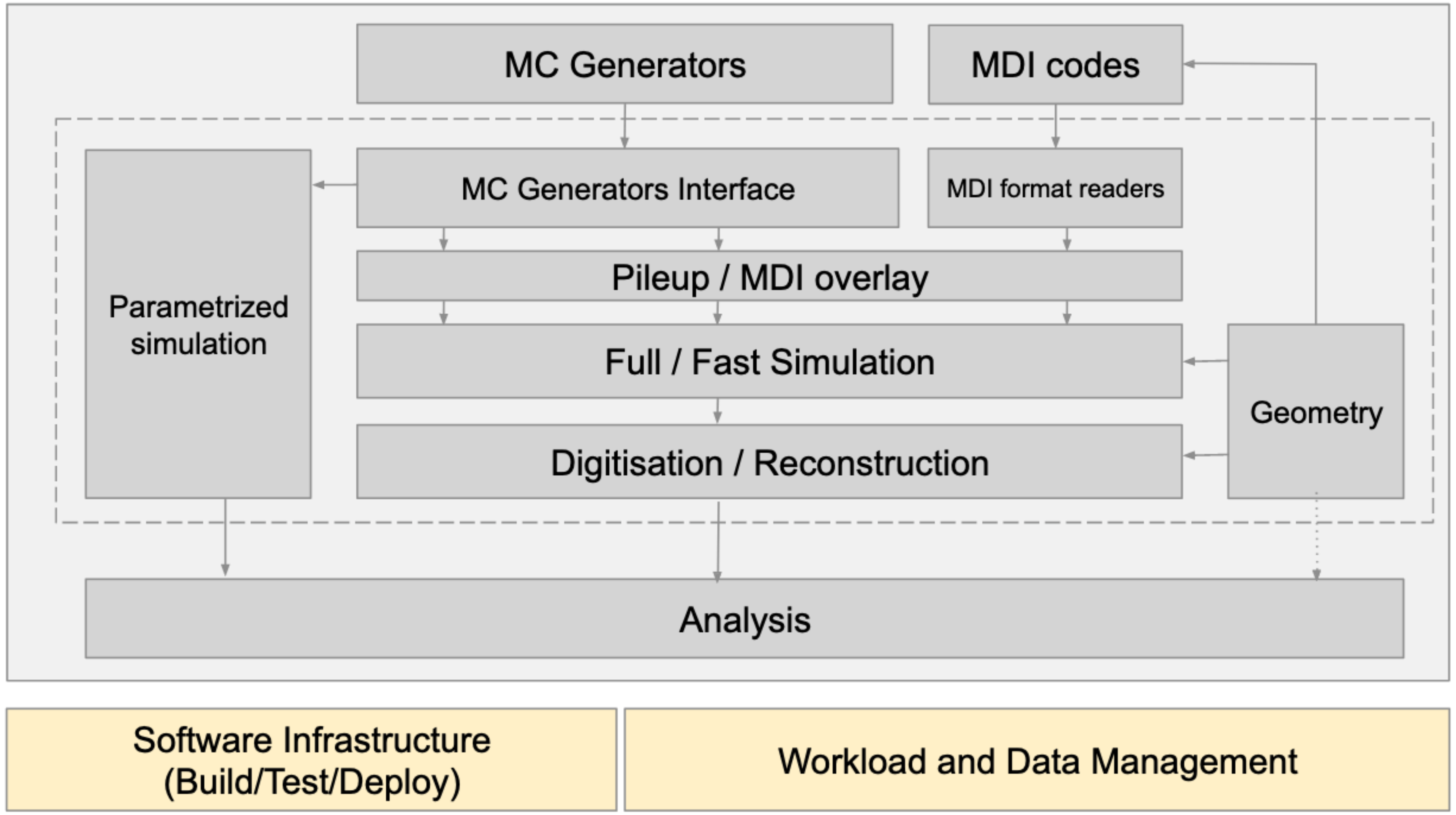


*Some testbeam related SW not yet included

source: T. Madlener

Event Processing Workflow

Event Processing Workflow



source: G. Ganis

Generation

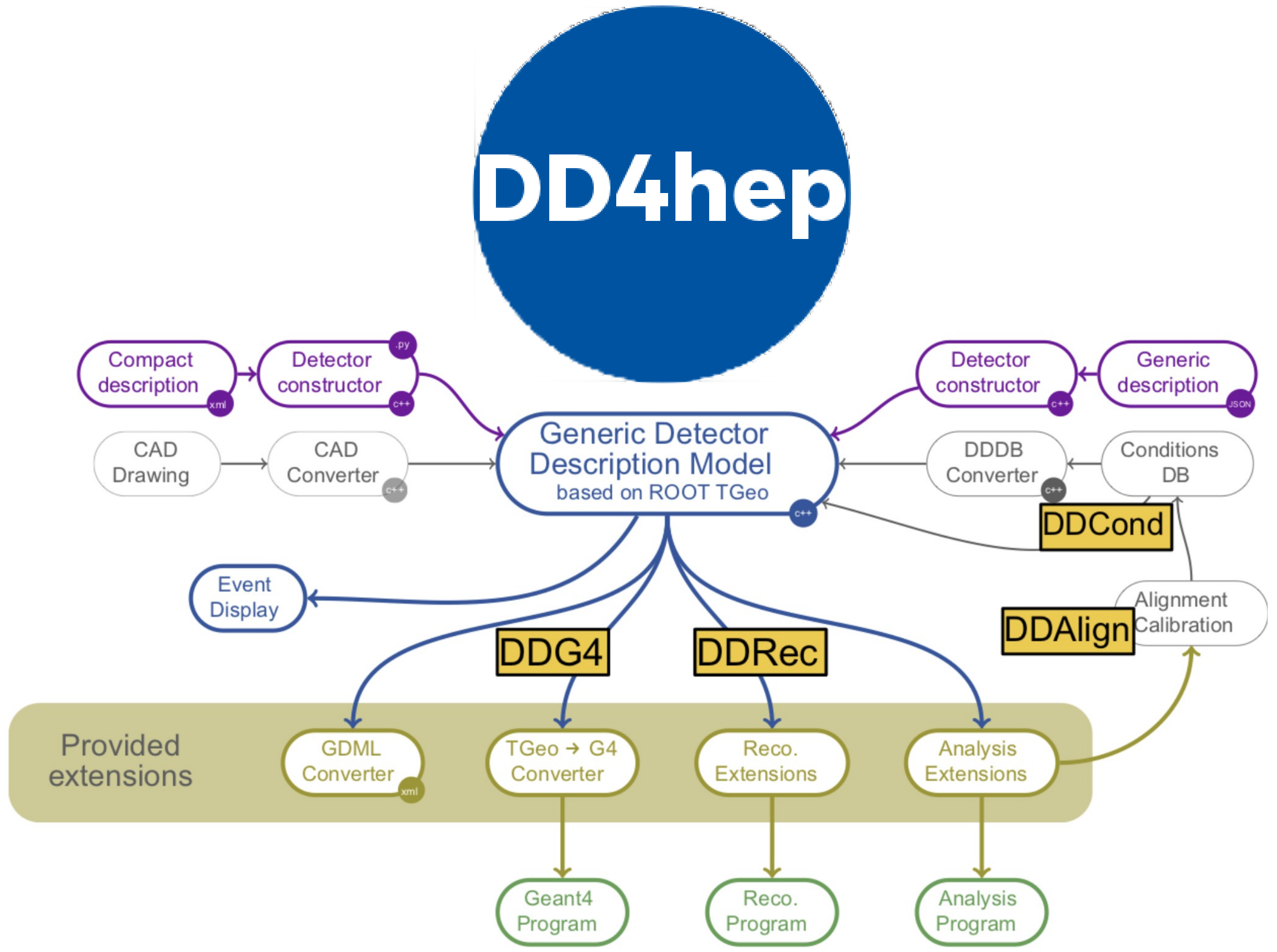
Theoretical efforts for ee generators is rumping up

- Most of the generators already packaged in Key4hep
  - MadGraph5_aMC@NLO, Pythia6/8, Herwig3, Whizard, BabaYaga, KKMCee, Guinea-Pig, Sherpa, EvtGen, ...
- Set of Gaudi algorithms and helpers packaged in k4Gen
  - Particle gun, particle filters, vertex smearing, ...
- New effort for unified generator configuration packaged in k4GeneratorsConfig
  - Integrated: BabaYaga, KKMC, MadGraph, Pythia, Sherpa, Whizard
- Any generator outputting established format (HepMC2/3, hepevt, stdhep, ...) can be input for Geant4 simulation with `ddsim`
- Prefered formats: HepMC3 and EDM4hep
  - Ongoing effort to make EDM4hep more suitable for generators
- Open topics include: ISR treatment, accuracy, Beam Energy Spread, crossing angle (+ spread), effect of the beams on final state

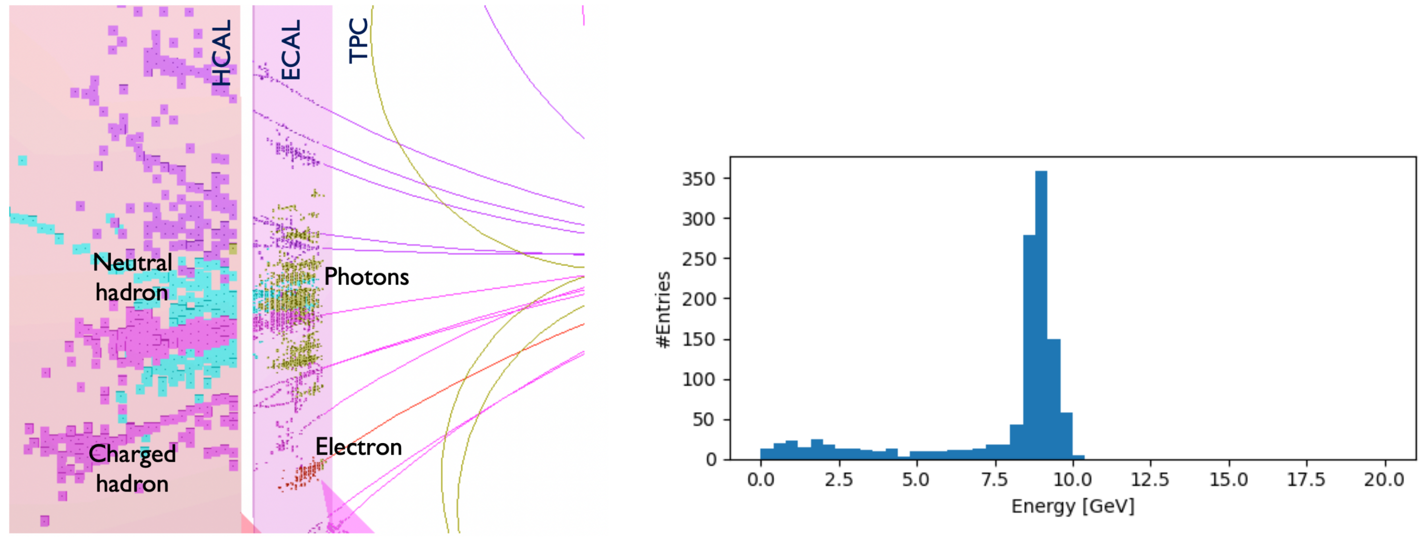
Simulation

Propagation of particles or decay products through detector

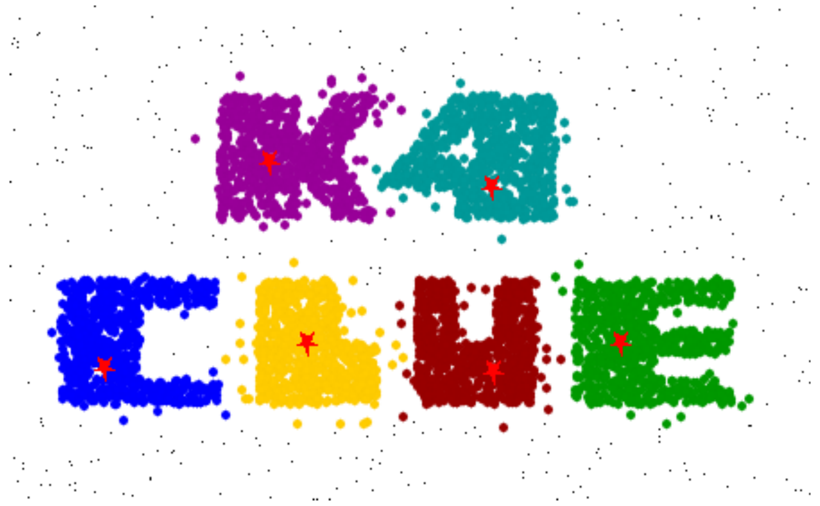
- Full simulation for FCC-ee detectors using `ddsim` (part of DD4hep)
- Fast simulation handled by k4SimDelphes
- Framework integration with k4SimGeant4 and `Gaussino` on back burner
- Ongoing work on three FCCee detector concepts IDEA, CLD and ALLEGRO almost complete
  - Effort now shifting from detector description towards Digitization and Reconstruction
  - Bi-weekly meeting, Wed 11:00 AM GVA: Indico category



Reconstruction



Pandora and Key4hep wrapper

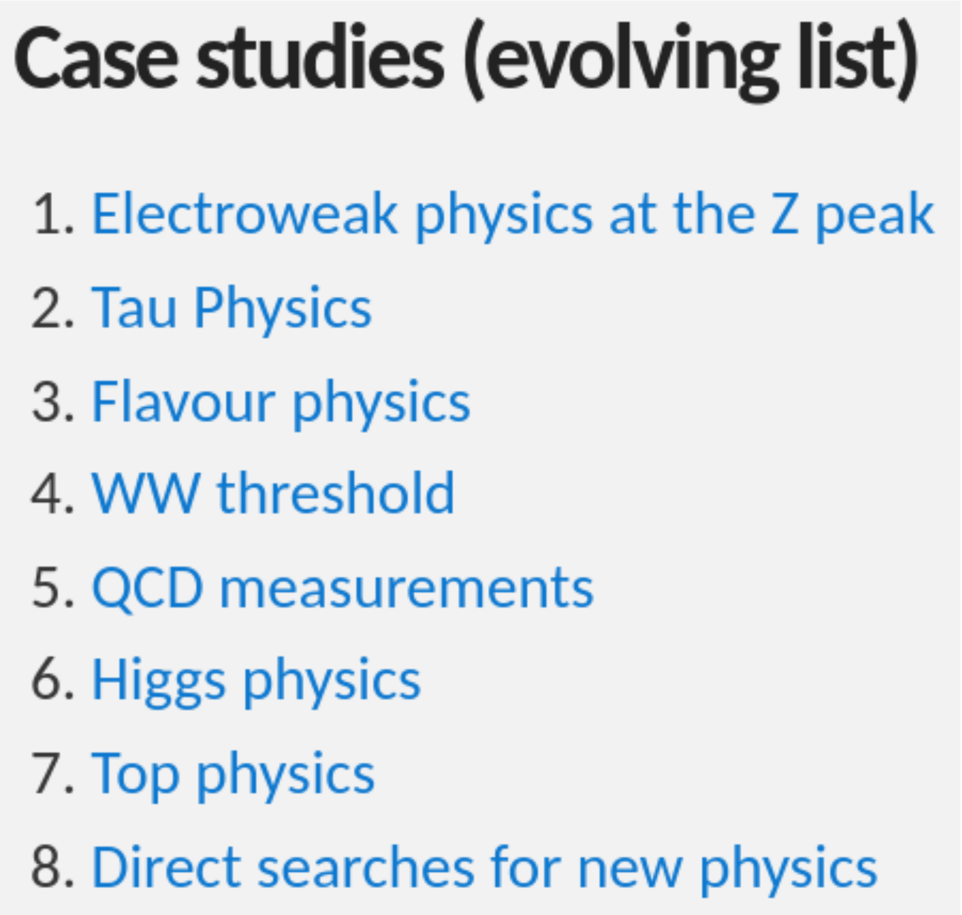


- Efforts are packaged per sub-detector type, for example
  - kRecCalorimeter: Reconstruction of Noble Liquid based calorimeter
  - k4RecTracker: vertex and tracker reconstruction as well as tracking
- Or per reconstruction solution, e.g.
  - k4GaudiPandora: Wrapping of the Particle flow framework
  - k4Clue: Clustering algorithm from HGCAL
- Some of the ongoing efforts also include
  - Particle identification with Array of RICH Cells (ARC)
  - Integration of ACTS tracking into Key4hep
  - Machine learning based flavor tagging

Analysis with FCCAnalyses

Analysis framework build on top of ROOT RDataFrame with input from EDM4hep

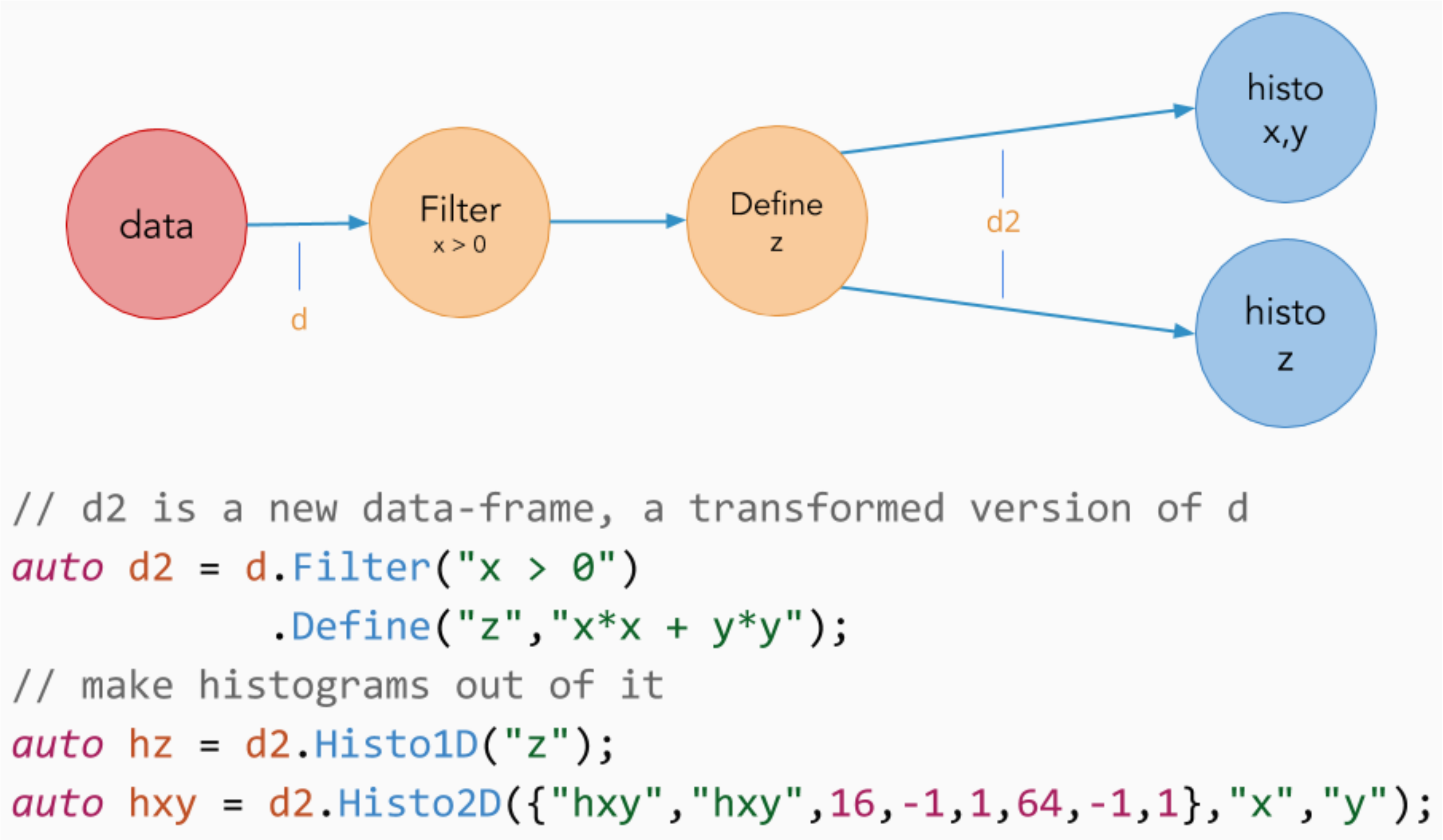
- Dependent on Key4hep Stack
- Manages input samples
- Has standard library of functions/functors
- Runs the dataframe
- Helps with histograms/plots
- Analyses Catalog:
  - FCCeePhysicsPerformance
  - FCChhPhysicsPerformance
- Bi-weekly meeting: Wed 4:00 PM GVA
  - Indico category



Case studies (evolving list)

1. Electroweak physics at the Z peak
2. Tau Physics
3. Flavour physics
4. WW threshold
5. QCD measurements
6. Higgs physics
7. Top physics
8. Direct searches for new physics

ROOT RDataFrame



```
// d2 is a new data-frame, a transformed version of d
auto d2 = d.Filter("x > 0")
           .Define("z","x*x + y*y");
// make histograms out of it
auto hz = d2.Histo1D("z");
auto hxy = d2.Histo2D({"hxy","hxy",16,-1,1,64,-1,1},"x","y");
```

- Describes processing of data as actions on table columns
  - Defines of new columns
  - Filter rules
  - Result definitions (histogram, graph)
- The actions are lazily evaluated
- Multi threading is available out of the box
- Optimized for bulk processing
- Allows integration of existing C++ libraries

Developing Key4hep / FCC Software

Access to CVMFS is crucial

Start by sourcing Key4hep stack from CVMFS

```
source /cvmfs/sw.hsf.org/key4hep/setup.sh
# or
source /cvmfs/sw-nightlies.hsf.org/key4hep/setup.sh
```

Usually, the packages are build with CMake

```
mkdir build install
cd build
cmake -DCMAKE_INSTALL_PREFIX=install ..
make -j4
make install
cd ..
```

To make your local version visible in your current shell, run

```
k4_local_repo
```

Documentation

- FCC Software: Main page with signpost
- FCC Tutorials: Tutorials on how to get started with FCC Software
- Key4hep Documentation: Growing documentation of the Key4hep and its components.
- FCC-ee Detector Full Sim: FCC-ee detectors implementation, simulation and reconstruction documentation
- FCC Software Glossary: A glossary of HEP and FCC-specific terms and concepts.
- ALEPH Documentation: Resurrecting ALEPH data in EDM4hep format (CERN log-in required).

Conclusions

- FCC is main stakeholder in the Key4hep stack project, which is becoming established stack delivering physics results
- Strive for integration and/or interoperability continues
- EDM4hep reached version 1.0 — backwards compatibility from this release
- Functional Gaudi on the way
- Simulation, FullSim and Recontruction far from complete
- Plenty of work ahead of us and You can join our meetings
  - FCC Software Indico Category
  - Key4hep Indico Category