EDM4hep - a common event data model for HEP experiments

ICHEP 2022, Bologna (IT)

09.07.22

Placido Fernandez Declara (CERN), Frank Gaede (DESY), Gerardo Ganis (CERN), Benedikt Hegner (CERN), Clement Helsens (KIT), Thomas Madlener (DESY), Andre Sailer (CERN), Graeme A Stewart (CERN), Valentin Volkl (CERN)

HELMHOLTZ RESEARCH FOR GRAND CHALLENGES

HSF HEP Software Foundation

CERN

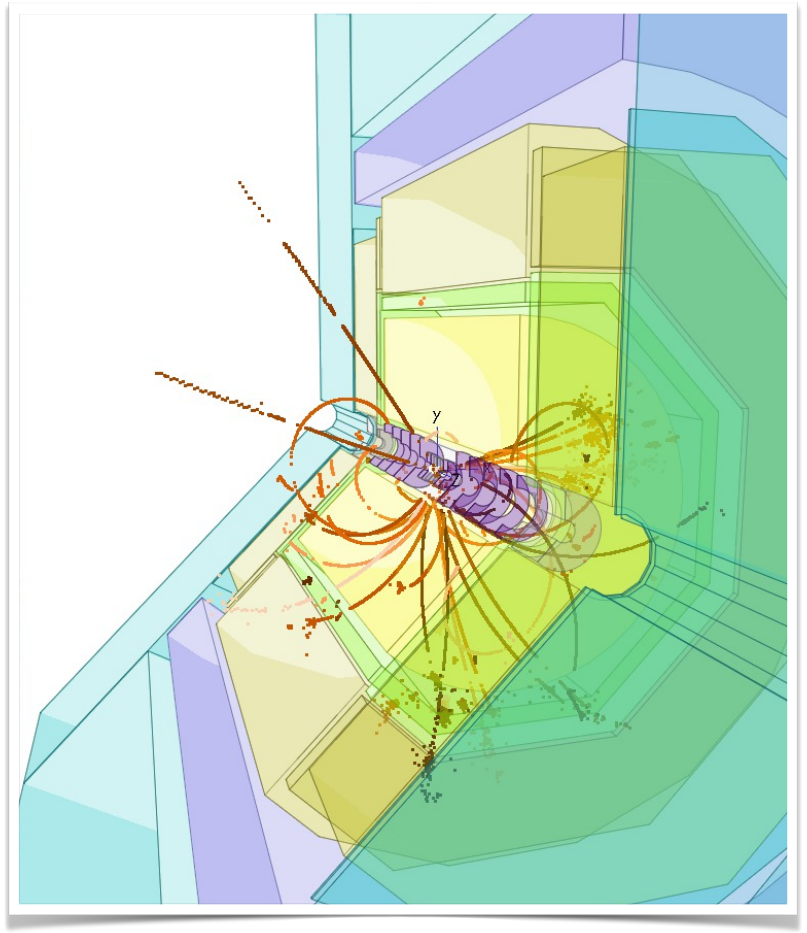
AIDA innova

FTX

DESY.

Outline

- Introduction
  - EDMs and Key4hep
- PODIO
  - recent developments
- EDM4hep
- Applications of EDM4hep
- Summary



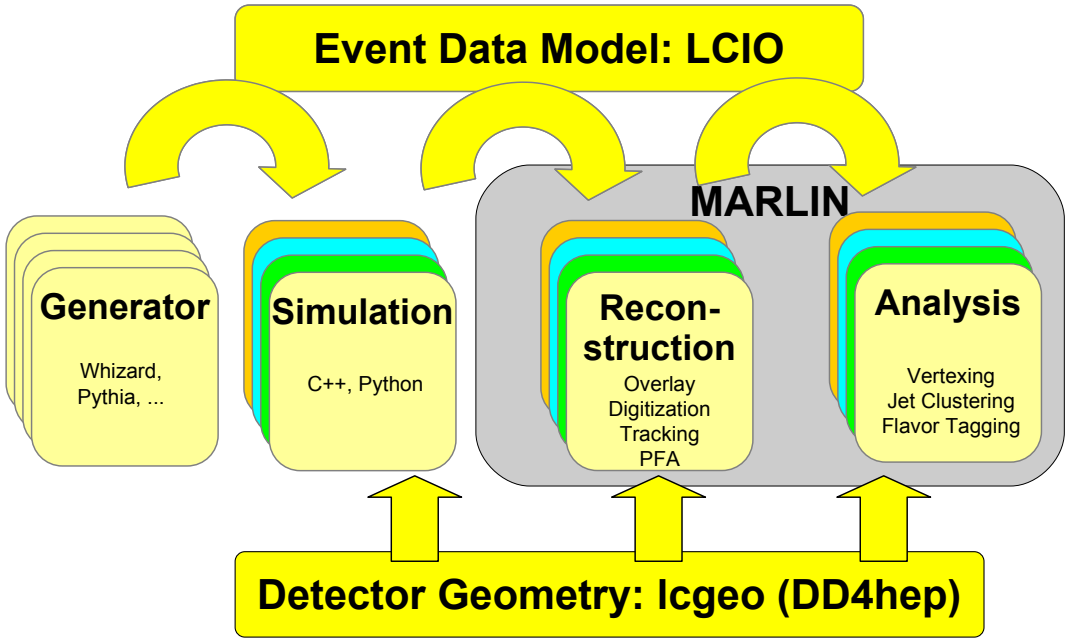
This project has received funding from the European Union's Horizon 2020 Research and Innovation programme under grant agreement No 101004761.

Introduction

EDM is central to any HEP software framework

- the Event Data Model (EDM) is at the **core of every HEP processing framework**
- it defines the **language** that is used throughout the processing chain: gen-sim-rec-ana
- **LCIO** had this role in iLCSoft - the framework used for >15 years in linear collider studies
- the agreement to extend this approach in the larger **Key4hep ecosystem** provided a great opportunity to
  - **modernise the EDM and underlying persistency** and file format(s)
- => **EDM4hep** - based on **PODIO** edm-toolkit

schematic view of iLCSoft

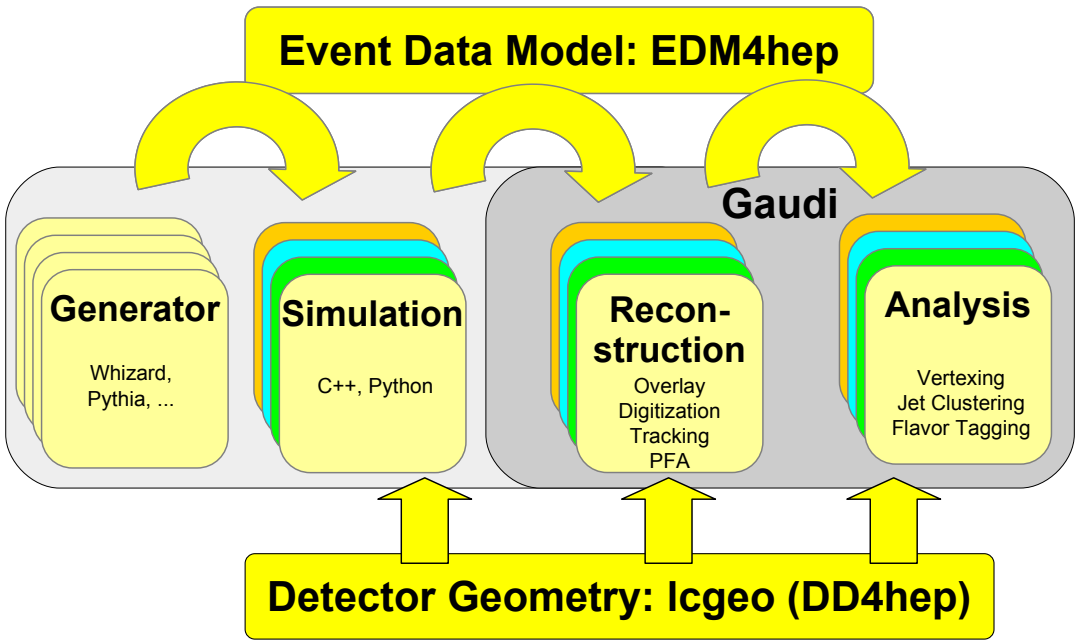


Introduction

EDM is central to any HEP software framework

- the Event Data Model (EDM) is at the **core of every HEP processing framework**
- it defines the **language** that is used throughout the processing chain: gen-sim-rec-ana
- **LCIO** had this role in iLCSoft - the framework used for >15 years in linear collider studies
- the agreement to extend this approach in the larger **Key4hep ecosystem** provided a great opportunity to
  - **modernise the EDM and underlying persistency** and file format(s)
- => **EDM4hep** - based on **PODIO** edm-toolkit

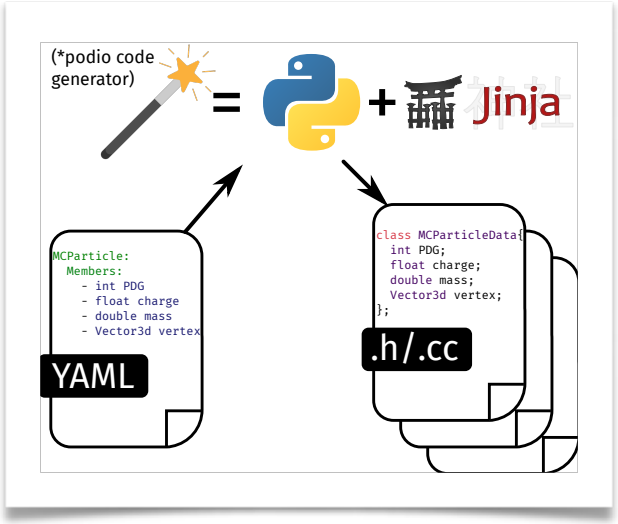
schematic view of Key4hep



see next talk about **Key4hep**…

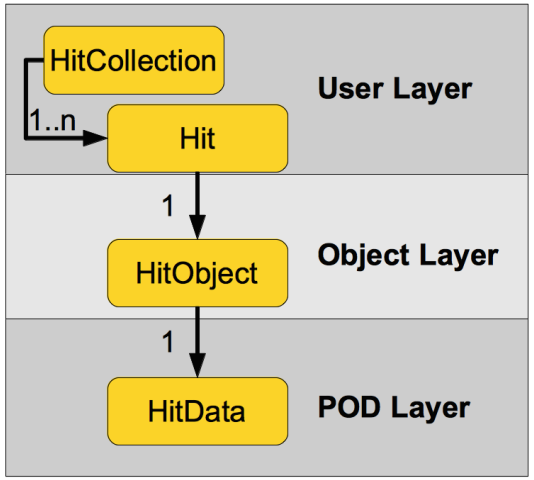
PODIO

the EDM toolkit



- PODIO developed in **AIDA2020** in context of FCC study
  - CHEP2016 - J. Phys.: Conf. Ser. 898 072039
  - CHEP2019 - EPJ Web Conf. 245 (2020) 05024
  - CHEP2021 - EPJ Web Conf. 251 (2021) 03026
- first used in **FCC-edm/pLCIO** and now for **EDM4hep**

- use ***yaml*-files** to define EDM objects then **generate C++** code via **Python/Jinja** scripts
- three layers of classes:
  - **POD** layer - the actual data in array of structs
  - **Object** layer - add relations and vector members
  - **User** layer - thin handles and collections



PODIO

EDM definition

- date model entirely defined w/ simple ***yaml*-file**:
- basic components
- data members:
  - basic types, std::array, components,…
- one-to-one, one-to-many relations
- (simple) user code
- vector members
  - (breaking *PODness*)

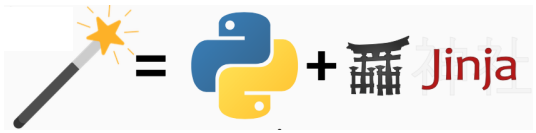
```yaml
components: edm4hep::Vector3f:
  Members: [float x, float y, float z]

datatypes:

  edm4hep::ReconstructedParticle:
    Description: "Reconstructed Particle"
    Members:
      - edm4hep::Vector3f momentum // [GeV] particle momentum
      - std::array<float, 10> covMatrix // energy-momentum covariance
    OneToOneRelations:
      - edm4hep::Vertex startVertex // start vertex associated to this particle
    OneToManyRelations:
      - edm4hep::Cluster clusters // clusters that have been used for this particle
      - edm4hep::ReconstructedParticle particles // associated particles
    ExtraCode:
      declaration: "bool isCompound() const { return particles_size() > 0; }\n"

  edm4hep::ParticleID:
    VectorMembers:
    - float parameters // hypothesis params
```

from edm4hep.yaml

PODIO

EDM definition

- date model entirely defined w/ simple ***yaml*-file**:
- basic components
- data members:
  - basic types, std::array, components,…
- one-to-one, one-to-many relations
- (simple) user code
- vector members
  - (breaking *PODness*)




```
class ReconstructedParticle {

public:
  ReconstructedParticle();
  ~ReconstructedParticle();

  const edm4hep::Vector3f& getMomentum() const;

  const std::array<float, 10>& getCovMatrix() const;
  const float& getCovMatrix(size_t i) const;

  const edm4hep::Vertex getStartVertex() const;

  unsigned int clusters_size() const;
  edm4hep::Cluster getClusters(unsigned int) const;
  std::vector<edm4hep::Cluster>::const_iterator clusters_begin() const;
  std::vector<edm4hep::Cluster>::const_iterator clusters_end() const;
  podio::RelationRange<edm4hep::Cluster> getClusters() const;

  bool isCompound() const { return particles_size() > 0 ;}

  // ...

private:
  ReconstructedParticleObj* m_obj;
};
```

C++ code generated from edm4hep.yaml (comments omitted)

PODIO

accessing the data - via generated code

C++ code w/ value-semantics
- fast and easy to use
- recommended access in experiment software

```cpp
auto recos = ReconstructedParticleCollection();
// ... fill ...
for (auto reco : recos) {
  auto vtx = reco.getStartVertex();
  for (auto rp : reco.getParticles()) {
    auto mom = rp.getMomentum(); }
}
```

```python
recos = ReconstructedParticleCollection()
#... fill ...
for reco in recos:
    vtx = reco.getStartVertex()
    for rp in reco.getParticles():
        mom = rp.getMomentum()
```

Pythonic interface
- generated w/ ROOT dict
- convenient for quick checks
  - rather slow…

access via RDataFrame
- read ROOT files directly
- fast, yet less convenient for more complex tasks (relations,…)

```python
d = ROOT.RDataFrame('events', 'events.root')
h = (d.Define('abs_pdg', 'abs(Particle.PDG)')
      .Define('mu_sel', 'abs_pdg == 13')
      .Define('mu_px', 'Particle.momentum.x[mu_sel]')
      .Histo1D('mu_px'))
h.DrawCopy()
```

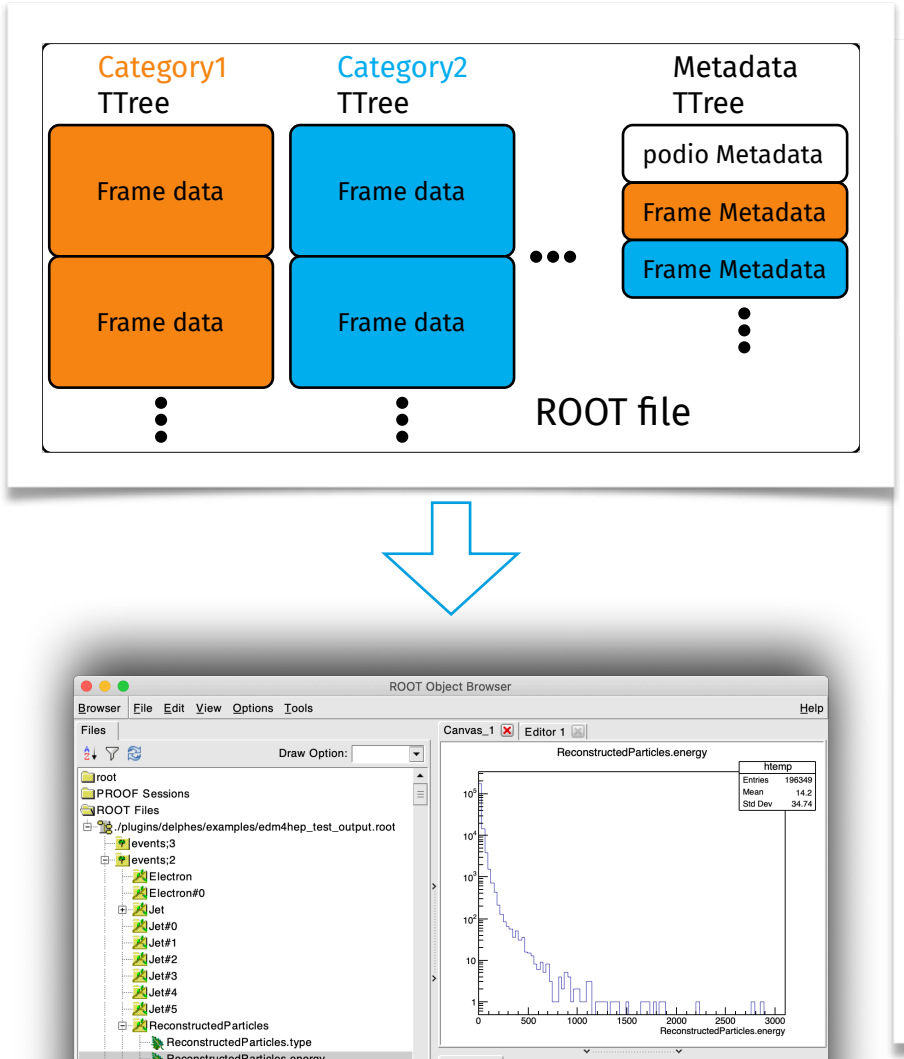
PODIO

I/O backends and File formats

- **default** I/O backend: **ROOT**
  - POD buffers are stored as branches in a *TTree*
- files can also be read without EDM library(!)
  - e.g. RDataFrame, TBrowser, uproot,…

- **alternative** I/O backend: **SIO**
  - persistency library used in LCIO
  - complete events are stored as binary records
  - … faster reading of complete events…

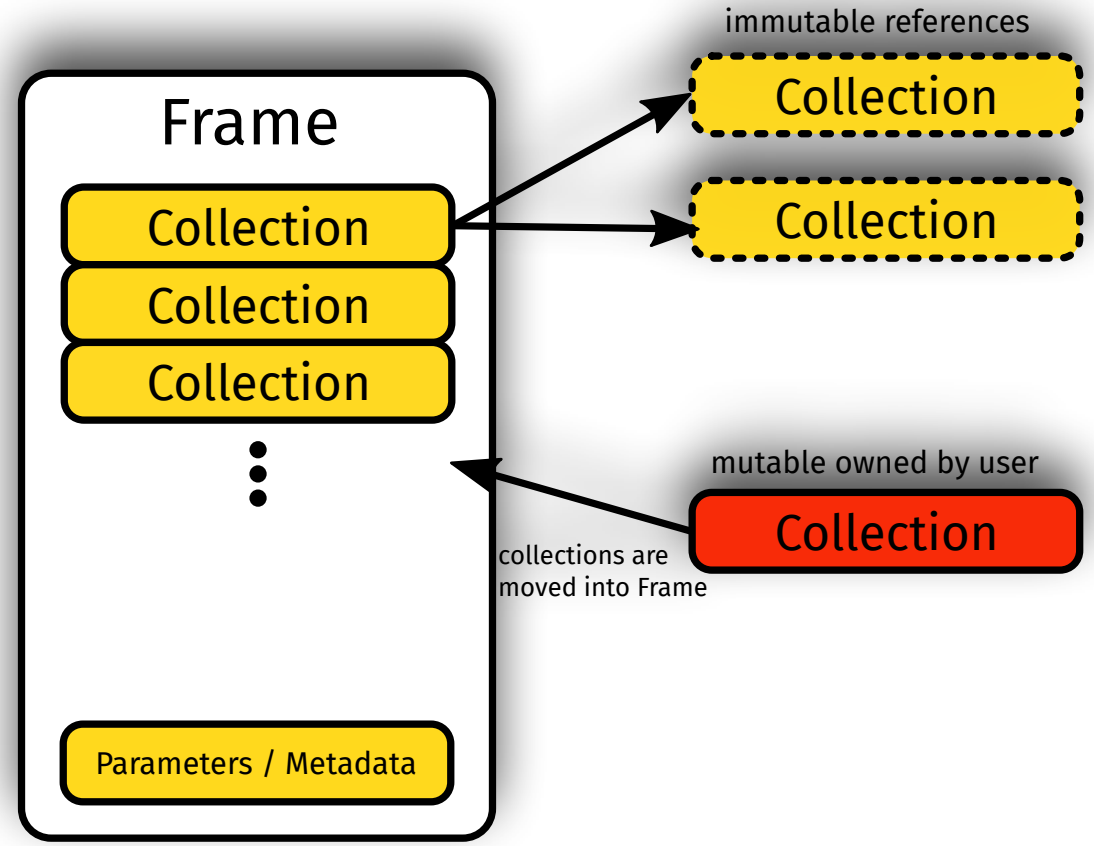
- adding more I/O backends is possible
  - e.g HDF5



PODIO recent developments

Frames

- replaced initial (and experimental) EventStore in PODIO w/ a **new Frame concept**
- Frames can hold data for any validity range, e.g.
  - events, runs, lumi sections, …
  - previously ‘events only’
- attempt to guarantee ***thread-safety***:
  - after a collection has been put (moved !) into a frame only *immutable* access possible
- implemented for ROOTWriter and SIOWriter
- prepare for additional features:
  - hooks schema evolution
  - potential lazy unpacking
  - …



file format changed with introduction of Frames
should be last non-backward compatible change in PODIO

PODIO recent developments

schema evolution

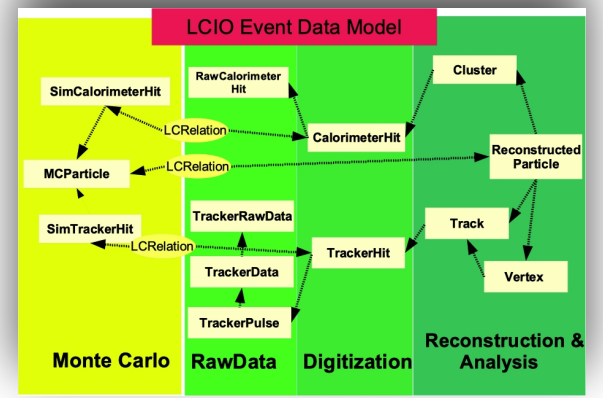
- schema evolution at data model definition:
  - data model description allows **automatic diffing** of versions: `podioDiff version1.yaml version2.yaml`
  - checks which changes are detected, supported automatically and whether there are ambiguities
  - the entire **migration code** is generated at **compile time** from the schemata given
- schema evolution at runtime depends on the I/O back-end
  - in memory **only the latest version is available** - schema evolution is applied in reading layer
  - for ROOT backend, most schema evolution is handled by **ROOT automatically**
  - for SIO schema evolution will be done with the **auto-generated code** created during data model definition.
- schema evolution and metadata:
  - data files and data models carry a **version number** with them
    - potentially nice to have: storing the schema alongside the data

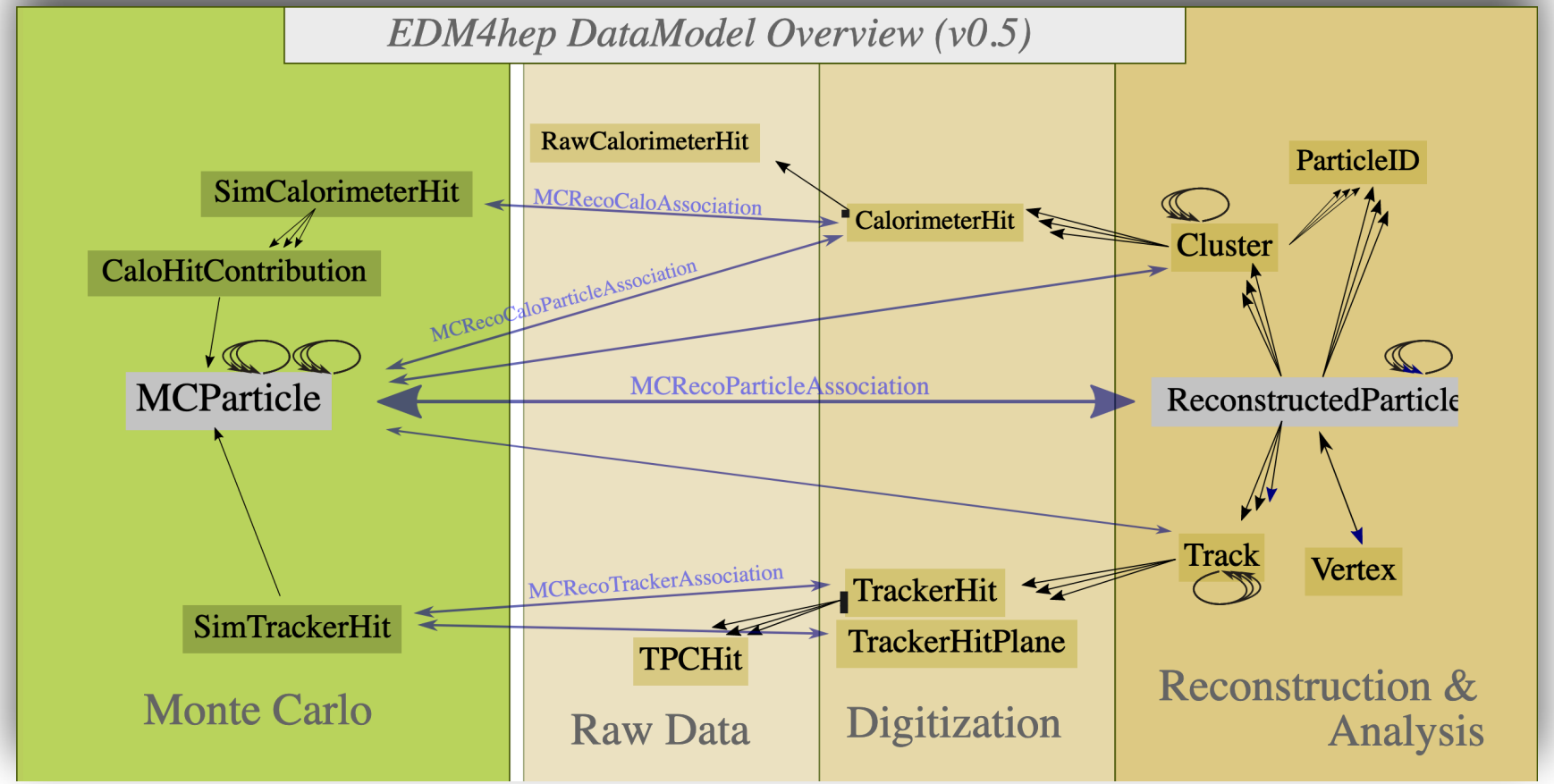
to be released soon

EDM4hep

the actual event data model

for more details: https://edm4hep.web.cern.ch/



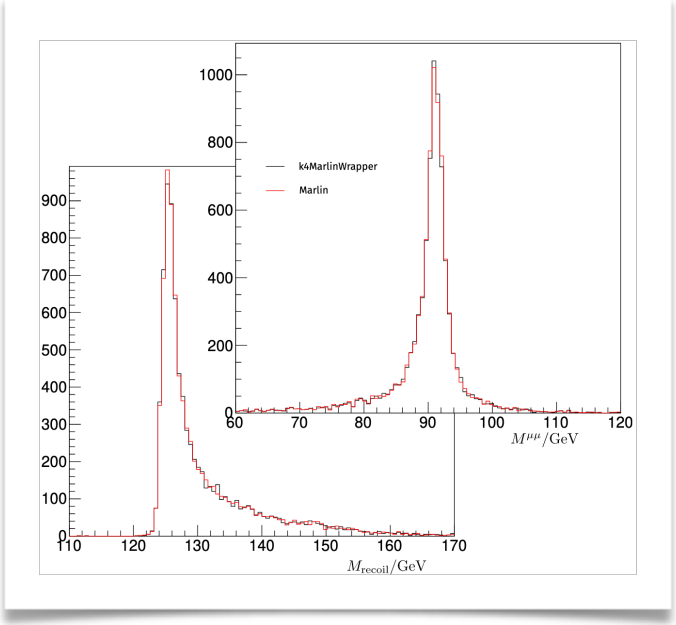


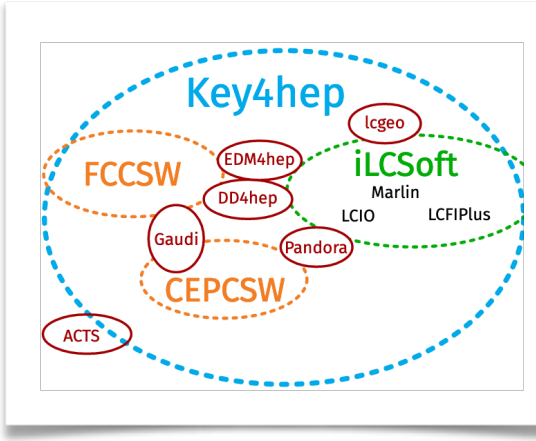
- **hierarchical** EDM from MC-truth (MCParticle) to high level objects (ReconstructedParticles)
  - too a large extent **one-to-one correspondence w/ LCIO**
  - should serve the needs of **all lepton colliders** - potentially some extensions/add-ons for others ?

EDM4hep in use

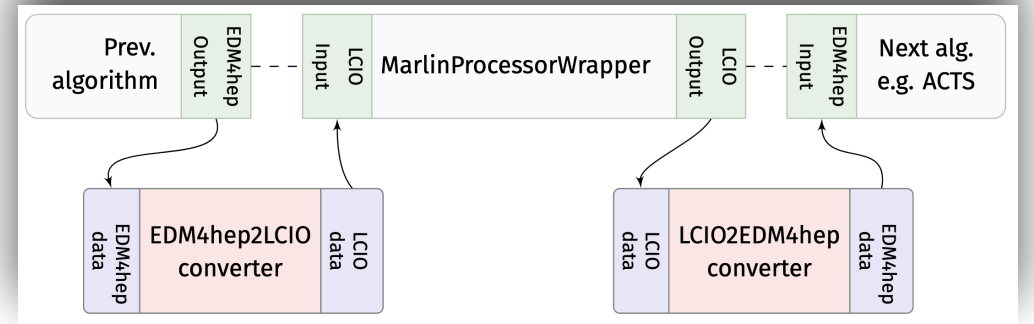
for the linear collider community: ILC, CLIC

- linear collider community is moving to Key4hep and **EDM4hep** in adiabatic way
- need to **preserve** all existing **code base** and algorithms
- possible via MarlinWrapper (see Key4hep talk)





- LCIO <-> EDM4hep conversion available
- can run full simulation and reconstruction chain as before and write **EDM4hep output**
- plan to finalise the physics validation of EDM4hep output this summer



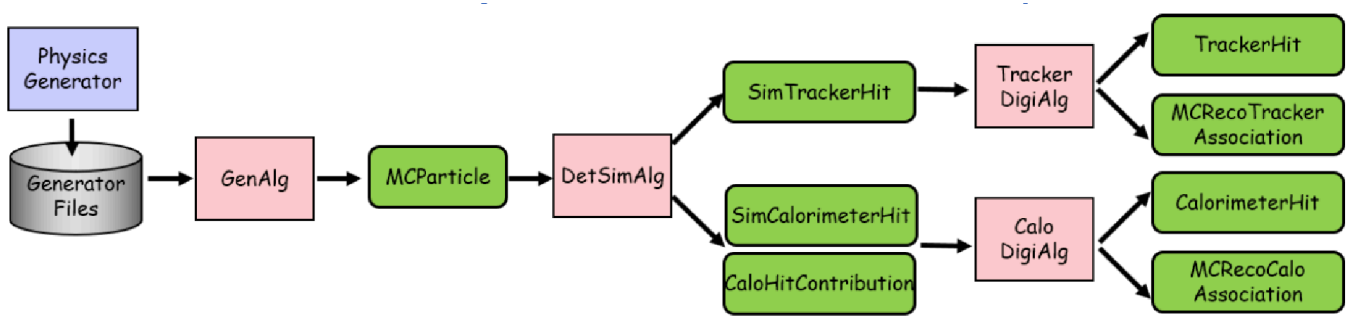
EDM4hep in use

for the circular collider community: FCCee, CEPC

- **EDM4hep** used in **FCCAnalysis**
  - Python scripts using **RDataFrame** on EDM4hep ROOT files directly
  - **fast** yet not using the EDM4hep API
  - the only **production use** of EDM4hep so far

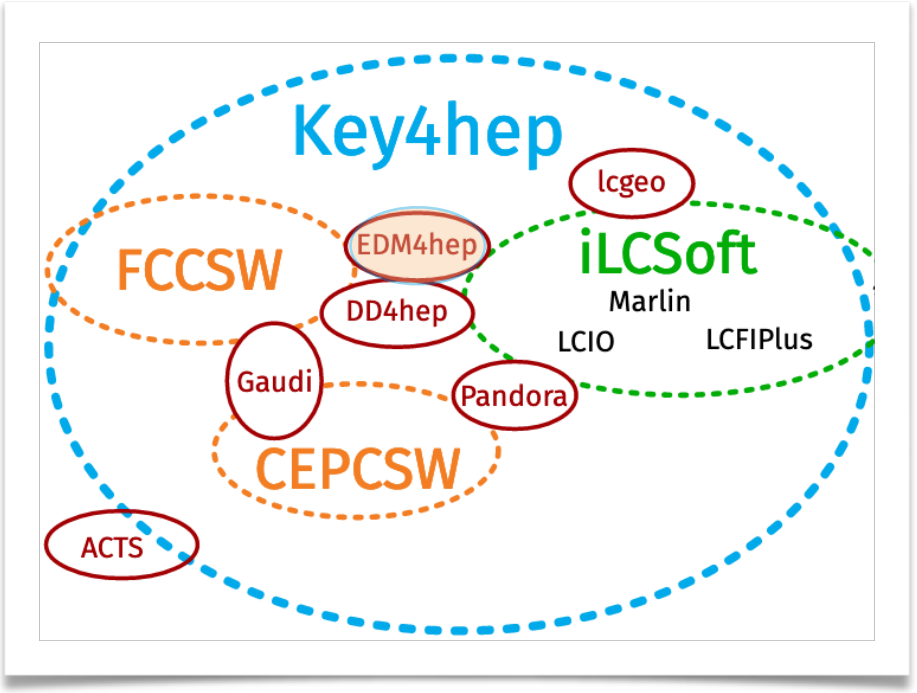
```
(self.df
# define an alias for muon index collection
.Alias("Muon0", "Muon#0.index")
# define the muon collection
.Define("muons",  "ReconstructedParticle::get(Muon0, ReconstructedParticles)"
#select muons on pT
.Define("selected_muons", "ReconstructedParticle::sel_pt(10.)(muons)")
variables = {
    "mz":{"name":"zed_leptonic_m","title":"m_{Z} [GeV]","bin":125,"xmin":0,"xmax":250},
    "mz_zoom":{"name":"zed_leptonic_m","title":"m_{Z} [GeV]","bin":40,"xmin":80,"xmax":100},
    "leptonic_recoil_m":{"name":"zed_leptonic_recoil_m","title":"Z leptonic recoil [GeV]","bi
    "leptonic_recoil_m_zoom":{"name":"zed_leptonic_recoil_m","title":"Z leptonic recoil [GeV]
    "leptonic_recoil_m_zoom1":{"name":"zed_leptonic_recoil_m","title":"Z leptonic recoil [GeV
    "leptonic_recoil_m_zoom2":{"name":"zed_leptonic_recoil_m","title":"Z leptonic recoil [GeV
    "leptonic_recoil_m_zoom3":{"name":"zed_leptonic_recoil_m","title":"Z leptonic recoil [GeV
    "leptonic_recoil_m_zoom4":{"name":"zed_leptonic_recoil_m","title":"Z leptonic recoil [GeV
# find zed candidates from  di-muon resonances
.Define("zed_leptonic",         "ReconstructedParticle::resonanceBuilder(91)(
# write branch with zed mass
.Define("zed_leptonic_m",       "ReconstructedParticle::get_mass(zed_leptonic
# write branch with zed transverse momenta
.Define("zed_leptonic_pt",      "ReconstructedParticle::get_pt(zed_leptonic)"
# calculate recoil of zed_leptonic
.Define("zed_leptonic_recoil",  "ReconstructedParticle::recoilBuilder(240)(ze
# write branch with recoil mass
.Define("zed_leptonic_recoil_m","ReconstructedParticle::get_mass(zed_leptonic
.Define("zed_leptonic_charge","ReconstructedParticle::get_charge(zed_leptonic
```

- CEPC community considering to port existing iLCSoft algorithms from Marlin to Gaudi
  - replacing **LCIO** with **EDM4hep**



Summary and Outlook

- **EDM4hep** is a **common event data model** for HEP
  - developed in context of **Key4hep** software eco system
- based the EDM toolkit PODIO
- **PODIO** recently undergone significant new developments:
  - introduction of **Frames** (thread-safety)
  - **schema evolution**
- EDM4hep adopted by **ILC, CLIC, FCC, CEPC**
  - under investigation by **EIC**



aiming to have **first production releases** v1.0 this summer
will guarantee **backward compatibility** from then on

Pointers to documentation

and code repositories

- Key4hep
  - introduction: key4hep.github.io/key4hep-doc
  - key4hep - github organisation
- **EDM4hep**
  - github: key4hep/EDM4hep cern.ch/edm4hep
- **PODIO**
  - github: AIDASoft/podio

- k4MarlinWrapper
  - github: key4hep/k4MarlinWrapper
- FCCAnalyses
  - github: HEP-FCC/FCCAnalyses