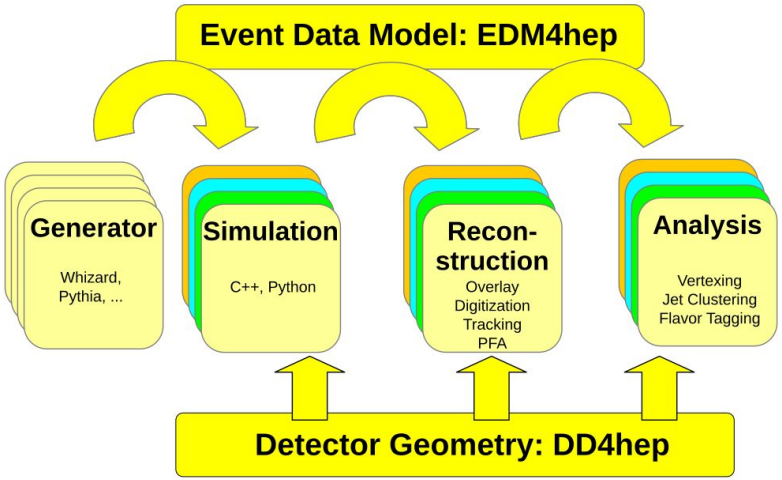
EDM4hep and PODIO

CCE Meeting March 2023

Benedikt Hegner
CERN
Stony Brook University
for the Key4hep team

The EDM at the core of HEP software

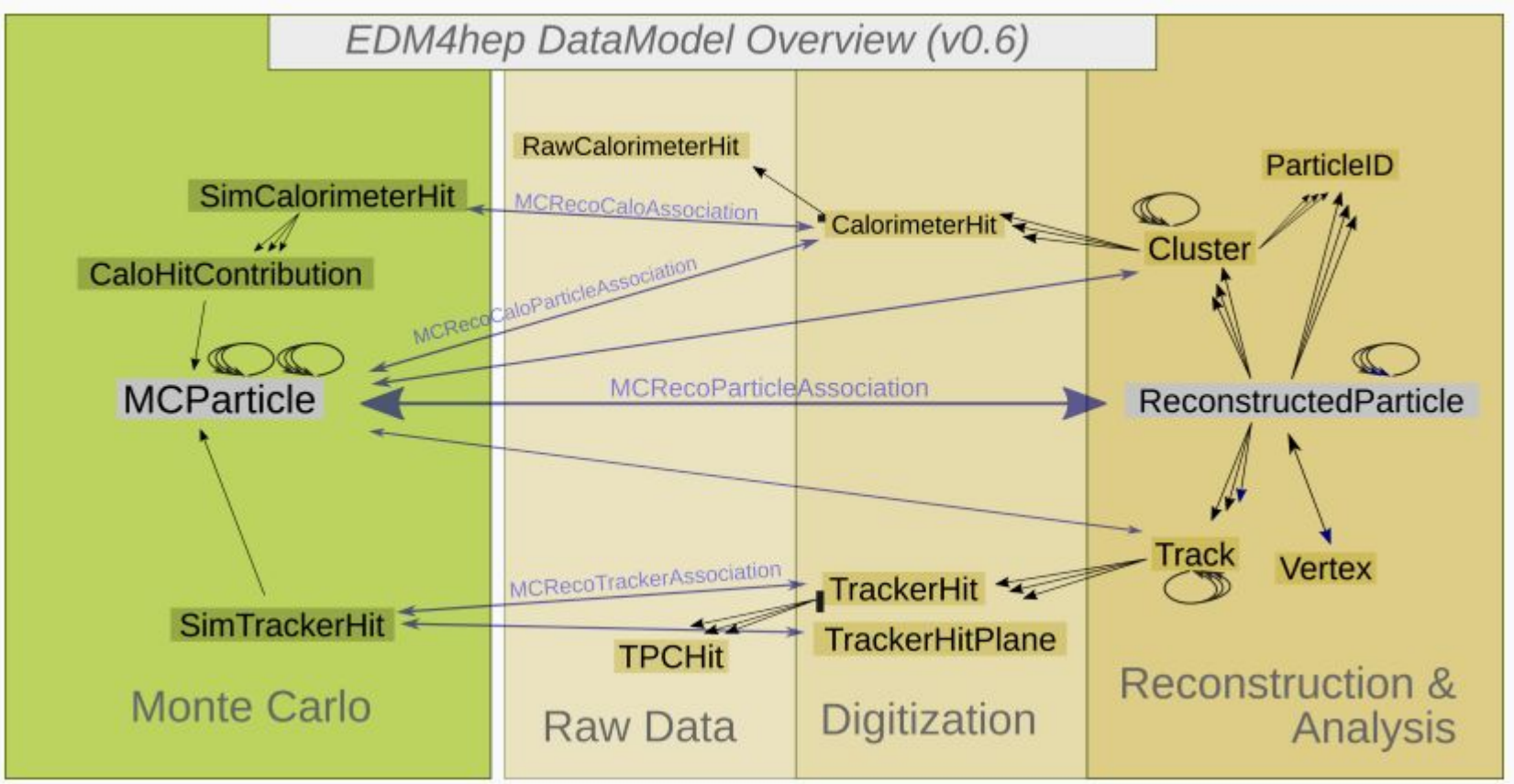
- Different components of HEP experiment software have to talk to each other
- The event data model defines the language for this communication
- Users express their ideas in the same language



EDM4hep - Goals & Motivation

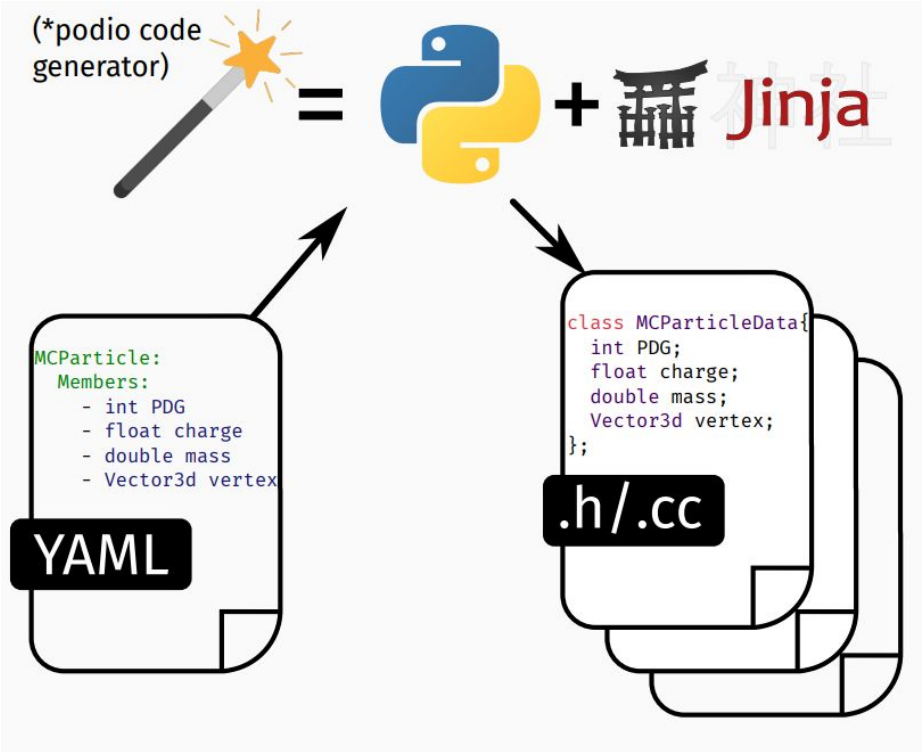
- The Key4hep project aims to develop a common software stack for all future collider projects
  - ILC, CLIC, FCC-ee & FCC-hh, CEPC, EIC, ...
- EDM4hep is the shared, common EDM that can be used by all communities in the Key4hep project (and others)
- EDM4hep has to support different use cases from these communities
- Efficiently implemented, support for multi-threading and with usage on heterogeneous resources in mind
- Built on experience from the “past” - mainly LCIO, which has been successfully shared by the LC communities

ED4hep schema



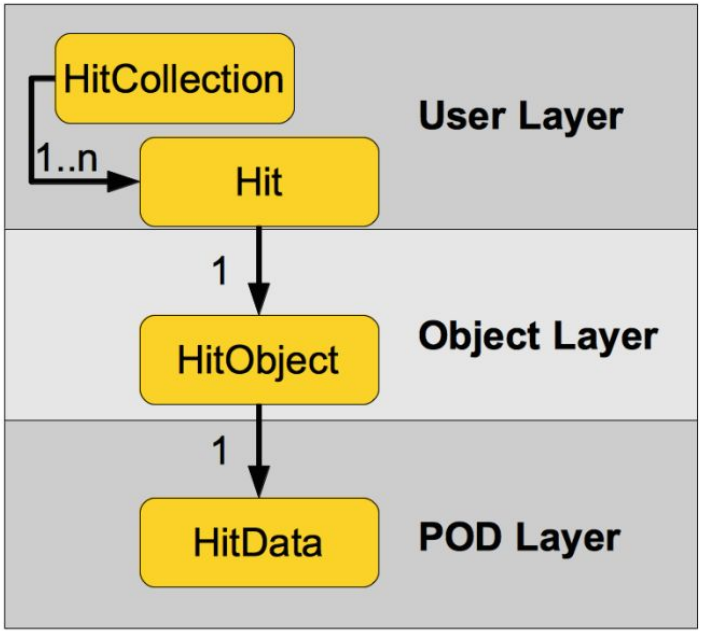
podio as generator for EDM4hep

- Traditionally HEP c++ EDMs are heavily Object Oriented
- Use podio to generate thread safe code starting from a high level description
- Provide an easy to use interface to the users



The three layers of podio

- The design of podio
  - favors **composition over inheritance**
  - uses **plain-old-data (POD)** types wherever possible
- Layered design allows for efficient memory layout and performant I/O implementation



podio - datamodel definition

- Reusable *components*
- Fixed sized arrays as members

- 1-1 and 1-N relations
- *VectorMembers*
- Additional user-provided code

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

datatypes:
  edm4hep::ReconstructedParticle:
    Description: "Reconstructed Particle"
    Author : "F.Gaede, DESY"
    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 isCompund() const { return particles_size() > 0; }\n"

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

podio - features of generated code

C++17 code with “value semantics”

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

Python bindings via PyROOT

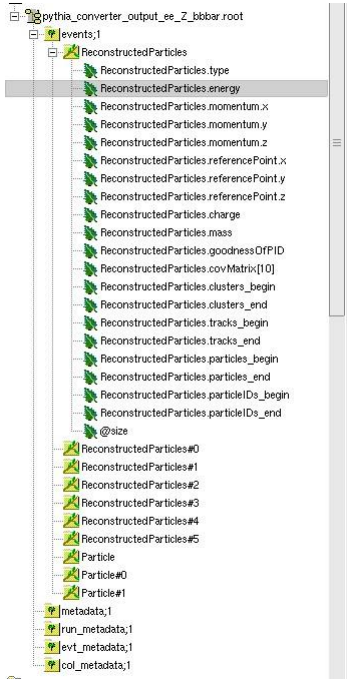
```python
recos = ReconstructedParticleCollection()
#... fill ...
for reco in recos:
  vtx = reco.getStartVertex()
  for rp in reco.getParticles():
    mom = rp.getMomentum()
```

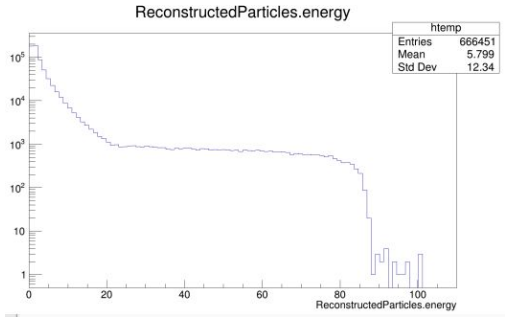
Using *RDataFrame* to read ROOT files
(*uproot* possible too)

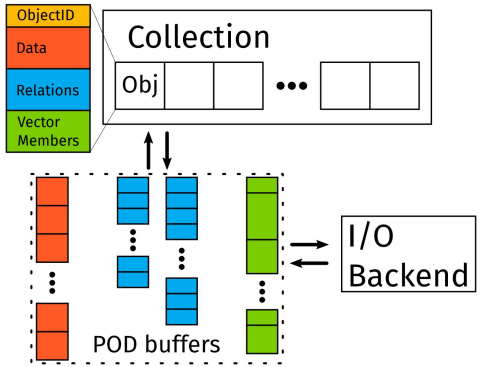
```python
d = ROOT.RDataFrame('events', 'events.root')
h = (d.Define('abs_pdg', 'abs(Particle.PDG)')
      .Define('mu_sel', 'abs_pdg == 13')
      .Define('mu_px',
              'Particle.momentum.x[mu_sel]')
      .Histo1D('mu_px'))
h.DrawCopy()
```

podio supports different I/O backends

- Default *ROOT* backend
  - POD buffers are stored as branches in a *TTree*
  - Files can be interpreted ***without EDM library***
- Alternative ***SIO*** backend
  - Persistency library used in ***LCIO***
  - Complete events are stored as binary records
- Schema can be stored within the files
- Adding more I/O backends is possible and foreseen







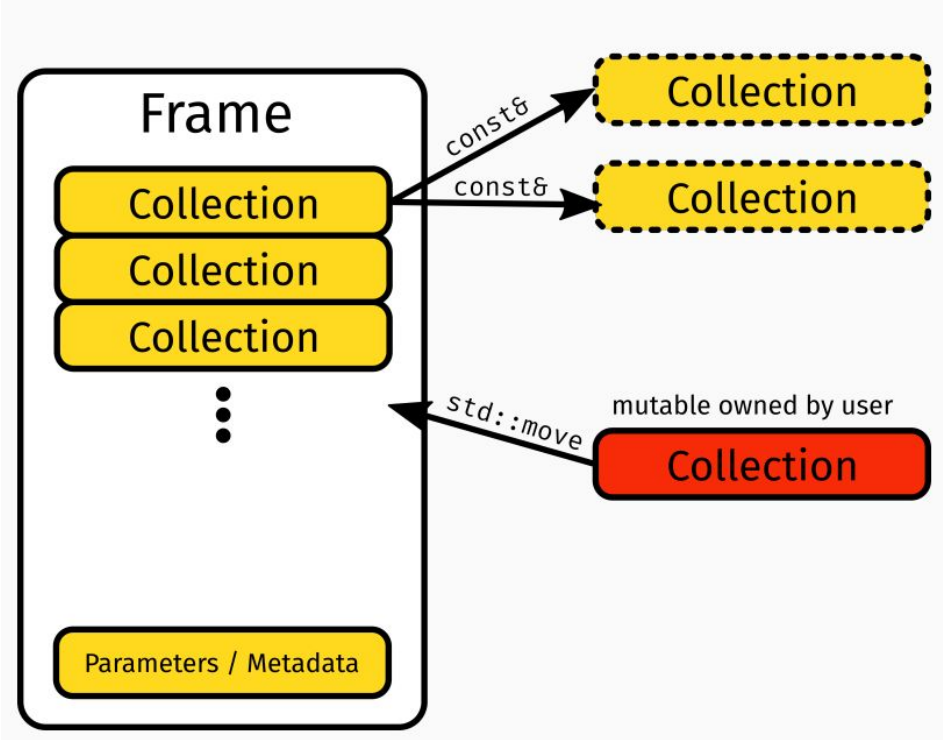
CMake for projects using podio

```cmake
find_package(PODIO)

# generate the c++ code from the yaml definition
PODIO_GENERATE_DATAMODEL(edm4hep edm4hep.yaml headers sources IO_BACKEND_HANDLERS "ROOT;SIO")
# compile the core data model shared library (no I/O)
PODIO_ADD_DATAMODEL_CORE_LIB(edm4hep "${headers}" "${sources}")
# generate and compile the ROOT I/O dictionary
PODIO_ADD_ROOT_IO_DICT(edm4hepDict edm4hep "${headers}" src/selection.xml)
# compile the SIOBlocks shared library for the SIO backend
PODIO_ADD_SIO_IO_BLOCKS(edm4hep "${headers}" "${sources}")

# Install the created targets
install(TARGETS edm4hep edm4hepDict edm4hepSioBlocks)
```

- Easy to use functions for integrating a podio generated EDM into a project
- Split into core EDM library and I/O handling for different backends
  - Pick what you need
  - I/O handling parts dynamically loaded by podio on startup

The Frame - A generalized (event) data container

- Container aggregating all relevant data
- Defines an *interval of validity* / category for contained data
  - Event, Run, readout frame, ...
- Easy to use and thread safe interface for data access
  - Immutable read access only
  - Ownership model reflected in API
- Decouples I/O from operating on the data



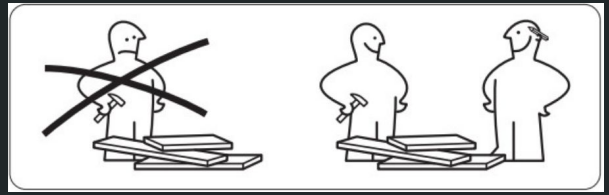
Prototyping of new data types

- podio comes with a mechanism to extend existing (“upstream”) datamodels
- EDM4hep uses this for prototyping new datatypes
  - Have to avoid to fracture EDM4hep
  - Goal is always inclusion into EDM4hep
- Used in Key4hep for some detector prototyping
  - Room for more detector concepts in EDM4hep!

Ongoing work and future plans

- Release v1.0 with backwards compatibility from then on
  - Need to finish some clean-up tasks first
- Propagate Frame based I/O to all currently existing “customers”
  - Framework integration, ddsim (DD4hep) output, ...
- Implement currently missing features
  - E.g. User defined associations between arbitrary types
  - Interface types that allow for easier high level workflows (e.g. tracker hits for different technologies)
- Start exploring work on heterogeneous resources
- Multi-language support based on abstract data model definitions
  - Did so with podio in Julia exercise during 2022 GSoC
  - Will do a prototype with Rust this year

Conclusions



- EDM4hep is the shared, common EDM for the Key4hep project
  - Community effort is a success
- It is generated via the podio EDM toolkit
- Efficient implementation of data types and flexible I/O capabilities
- podio extension mechanism can be used to add new data types
- EDM4hep is open for new data types for not yet covered detector types

Pointers to software resources

- EDM4hep
  - https://github.com/key4hep/EDM4hep
- podio
  - https://github.com/AIDASoft/podio
- Biweekly meetings for podio/EDM4hep discussion
  - https://indico.cern.ch/category/11461/