EDM4hep & DD4hep for LUXE

Some first steps

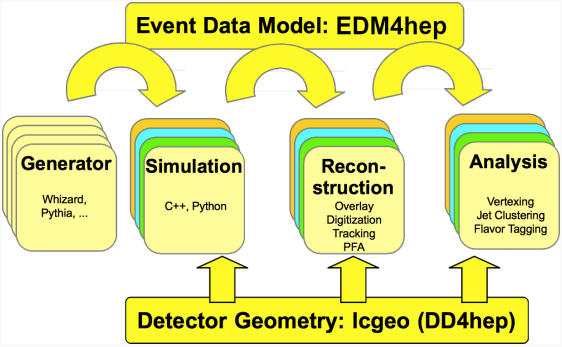
Thomas Madlener
with a lot of help from Sasha

LUXE SAS meeting

July 10, 2023

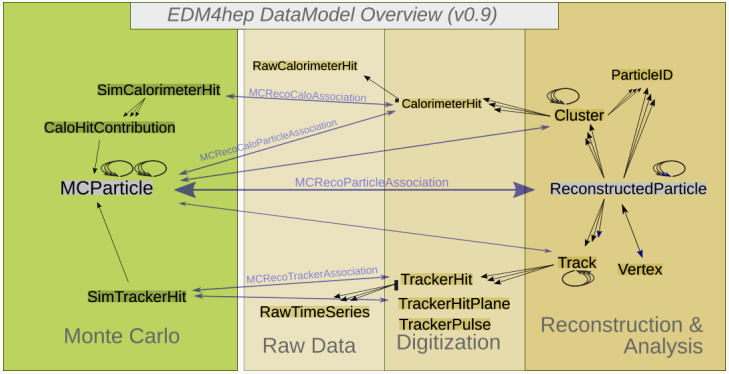
This project has received funding from the European Union's Horizon 2020 Research and Innovation programme under grant agreement No 101004761.

The EDM at the core of HEP software



- Different components of 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 - The common EDM for Key4hep



key4hep/EDM4hep

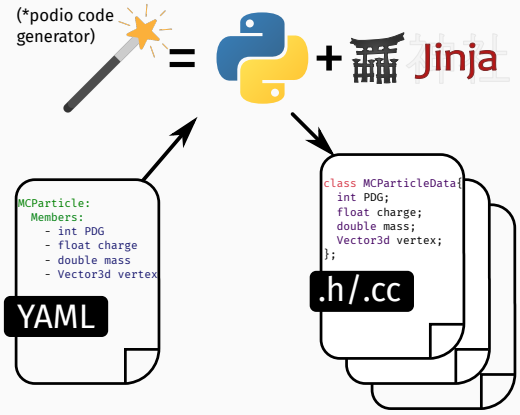
edm4hep.web.cern.ch

AIDASoft/podio

- Based on experience from LC and FCC
  - Focus on usability in analysis
- Works well for current Future Collider studies
- Quite stable over the last two years
- Generated via `podio`
- **Can be extended for LUXE specific use cases**

The podio EDM toolkit

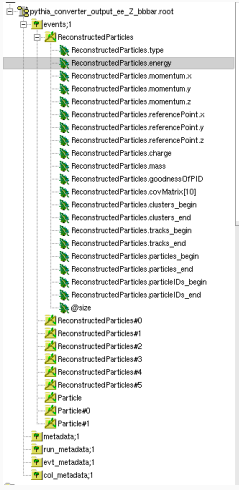
- Implementing a performant event data model (EDM) is non-trivial
- Use `podio` to generate code starting from a high level description
- Provide an easy to use interface to the users
- Main customers
  - key4hep/EDM4hep
  - eic/EDM4eic
- **Finishing schema evolution for v1.0**

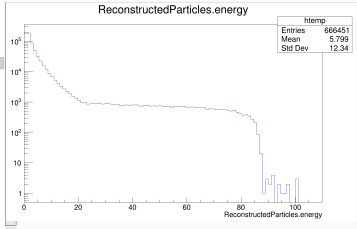


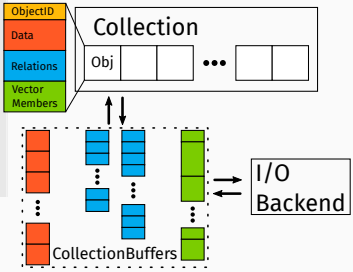
AIDASoft/podio

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**(!)
  - Can be used in `RDataFrame` or with `uproot`
- Alternative **SIO** backend
  - Persistency library used in `LCIO`
  - Complete events are stored as binary records
- Adding more I/O backends is possible



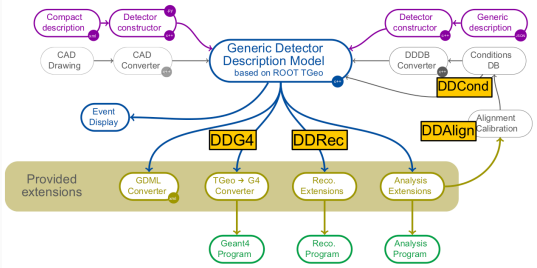


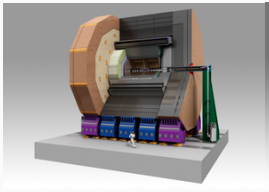


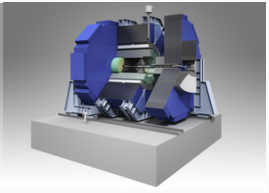
DD4hep - Detector description

dd4hep.web.cern.ch

- Complete detector description
  - Geometry, materials, visualization, readout, alignment, calibration, ...
- From a **single source of information**
  - Simulation, reconstruction, analysis
- Comes with a powerful plug-in mechanism that allows customization
- More or less “industry standard” now
  - ILC, CLIC, FCC, CEPC, EIC, LHCb, CMS, ...
- **`ddsim` - standalone simulation executable**







`lxsim` vs `ddsim`/DD4hep (at a very high level)

	`lxsim`	`ddsim`
detector description	C++	XML & C++ (detector constructors)
executable	compiled C++	python script (`ddsim`)
simulation config	`.mac` files	python (+ command line)
output	custom ROOT & HDF5 format	EDM4hep, ...

- Conceptually and (simulation) functionality wise quite similar
- Major differences:
  - `ddsim` (DD4hep) detector also encodes sensitive detectors (and their readouts) in geometry (XML)
  - `lxsim` can dump to GDML (only volumes, no info on sensitive detectors)
- DD4hep geometry can directly be used in reconstruction & analysis

Using GDML geometry with DD4hep

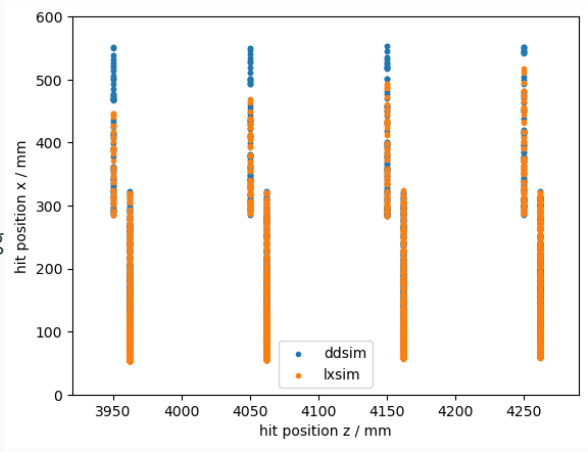
```
<detectors>
  <detector id="1" name="LUXE_GDML" type="DD4hep_GdmlDetector" vis="InvisibleWithChildren">
    <gdmlFile ref="lxgeomdump.gdml"/>
    <parent name="/world"/>
  </detector>
</detectors>

<plugins>
  <plugin name="DD4hep_PlacedVolumeProcessor">
    <arg value="-recursive"/>
    <arg value="-processor"/>
    <arg value="DD4hep_DetElementCreator"/>
    <arg value="-detector"/>
    <arg value="Tracker"/>
    <arg value="-print"/>
    <arg value="DEBUG"/>
    <arg value="-level"/>
    <arg value="4"/>
    <arg value="-material"/>
    <arg value="Silicon"/>
    <arg value="-match"/>
    <arg value="OPPPTracker"/>
    <arg value="-veto"/>
    <arg value="Support"/>
    <arg value="-type"/>
    <arg value="tracker"/>
  </plugin>
</plugins>

<fields>
  <field name="MagnetFields_Dipole" type="MultipoleMagnet" Z="0.*tesla">
    <position x="0*cm" y="0*cm" z="205.3*cm"/>
    <coefficient coefficient="-0.95*tesla" skew="0.*tesla"/>
    <shape type="Box" dx="16.5*cm" dy="3*cm" dz="61.9*cm"/>
  </field>
</fields>
```

← Almost complete DD4hep XML

- Read GDML geometry
  - (from `lxsim`)
- Need to add sensitive detectors based on heuristics
  - E.g. material, name, ...
- Magnetic field also needs to be added in XML

First steps and early results

- Particle gun at IP $(0, 0, 0)$, direction along beam axis, uniform energy range
- Silicon tracker only
- **This is more or less a technical demonstration**
  - Verify that loading GDML and using a plugin to place sensitive detectors works
  - Establish the potential of a migration in steps
  - Produce some first outputs in EDM4hep format



Next steps

Immediate

- More detailed analysis of first steps
  - Tracker hit position and energy correlations
- Make the calorimeter sensitive and repeat exercise for calorimeter hits

Workshop (goal)

- Implement “proper” tracker geometry and use that with rest in GDML
- Load tracker geometry from DD4hep to ACTS and reconstruct some tracks

Midterm

- Implement more detectors in DD4hep and replace GDML parts “one-by-one”
- Define datatypes to store hits from all detectors (new ones of necessary)

Summary

- Need a well defined EDM to facilitate reconstruction & analysis workflows
- Use EDM4hep where possible, extend as needed for LUXE
- Started to explore migration path to DD4hep
- Very first results based on GDML geometry dumped from `lxsim`
- The “real” work is yet to start and will take quite some time
  - Need to prioritize what to do first