EDM Toolkit - **PODIO**

and a short note on ACTS

F.Gaede, DESY

AIDA2020 Annual Meeting, Apr 4,2019

Outline

- Introduction and brief Reminder on PODIO
- Status of PODIO
- Recent developments
- Future Plans
- Short Note on ACTS activities
- Summary and Outlook

Introduction

- PODIO is a new EDM toolkit developed in AIDA2020
- based on the use of **PODs** for the event data objects (**P**lain-**O**ld-**D**ata object)

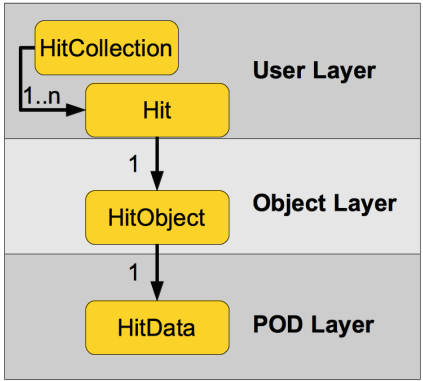
- PODIO originally developed in context of the **FCC** study
  - addressing the problem in a generic way
  - allowing potential re-use by other HEP groups
  - planned application to **LC** (LCIO) - see later

- one of the first projects adopted by the **HEP Software Foundation**

- see next few slides for a reminder of PODIO features ...

Implementation: the three PODIO layers

- user layer (API):
  - handles to EDM objects (e.g. **Hit**)
  - collection of EDM object handles (e.g. **HitCollection**).
- object layer
  - transient objects (e.g. **HitObject**) handling vector members and *references* to other objects
- POD layer
  - the actual POD data structures holding the persistent information (e.g. **HitData**)



Object ownership in PODIO

clear design of ownership (hard to make mistakes) in two stages:

objects added to event store are *owned by event store*

```
auto& hits = store.create<HitCollection>("hits") ;
auto h1 = hits.create( 1.,2.,3.,42.) ; // init w/ values
auto h2 = hits.create() ; // default construct
h2.energy( 42.) ;
```

objects created standalone are *reference counted* and automatically garbage collected:

```
auto h3 = Hit() ;
auto h4 = Hit() ;
hits.push_back( h3 ) ;
//  h1,h2,h3 are automatically deleted with collection
//  h4 is garbage collected
```

Relations between Objects

allow to have 1-1, 1-N or N-M relationships, e.g.

```
auto& hits = store.create<HitCollection>("hits");
auto& clusters = store.create<ClusterCollection>("clusters");
auto hit1 = hits.create(); auto hit2 = hits.create();
auto cluster = clusters.create();
cluster.addHit(hit1);
cluster.addHit(hit2);
```

referenced objects can be accessed via iterator or directly

```
for( auto h = cluster.Hits_begin(),end = cluster.Hits_end(); h!=end ++h){
    std::cout << h->energy() << std::endl;
}
auto hit  = cluster.Hits(42);
```

also standalone relations between arbitrary EDM objects

Code generation

- code (C++/Python) for the EDM classes is generated from yaml files
- EDM objects (data structures) are built from
  - basic type data members
  - components (structs of basic types)
  - references to other objects
- additional user code (member functions) can be defined in the yaml files

```
# LCIO MCParticle
  MCParticle:
    Description: "LCIO MC Particle"
    Author : "F.Gaede, B. Hegner"
    Members:
      - int pDG               // PDG code of the particle
      - int generatorStatus   // status as defined by the generator
      - int simulatorStatus   // status from the simulation
        #...
    OneToManyRelations:
      - MCParticle parents // The parents of this particle.
      - MCParticle daughters // The daughters this particle.
    ExtraCode:
      const_declaration:
      "bool isCreatedInSimulation() const {
          return simulatorStatus() != 0 ;
       } \n"
```

Python Interface

- Python is treated as first class citizen in the provided library
- can use *pythonic* code for iterators etc.
- implemented with PyROOT and some special usability code in Python

Python code example:

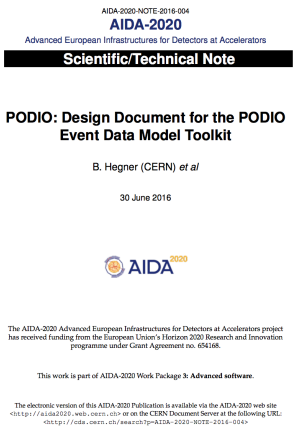
```
store = EventStore(filenames)
for i, event in enumerate(store):
  hits = store.get("hits")
  for h in hits:
    print h.energy()
```

Status: Milestones and Deliverables

Name	What	When
MS19	Design document for EDM Toolkit	M14
MS90	Application of EDM Toolkit to LC	M44
D3.4	Event Data Model Toolkit	M40

status

- all Milestones and Deliverables **reached on time**
- plan to continue improving PODIO, nevertheless
- some minor feature development done in context of FCC



AIDA-2020-NOTE-2016-004

AIDA-2020

Advanced European Infrastructures for Detectors at Accelerators

Scientific/Technical Note

PODIO: Design Document for the PODIO Event Data Model Toolkit

B. Hegner (CERN) *et al*

30 June 2016

AIDA 2020

The AIDA-2020 Advanced European Infrastructures for Detectors at Accelerators project has received funding from the European Union's Horizon 2020 Research and Innovation programme under Grant Agreement no. 654168.

This work is part of AIDA-2020 Work Package 3: **Advanced software**.

Recent Developments in PODIO

- moved Github repository to: https://github.com/aidasoft/podio
- added some standard templates from iLCSoft for release notes, issues, etc.

- addressed a few issues needed for MS90
  - add `CollectionBase::size()` method
  - implement **vector member** streaming

- some minor bugs and issues fixed

v00-09

- 2018-12-20 Frank Gaede (PR#39)
  - add some fixes and improvements
    - fix forward declarations in Object template when using a namespace for the EDM
    - fix array getter names when using the get/set syntax
    - add missing treatment for include statements in component's header files
    - handle array members in ostream operators
  - add CollectionBase::size() member function
    - allows to access collection size w/o knowing the concrete type
    - method is already generated in implementation classes
- 2018-12-06 Frank Gaede (PR#38)
  - add code generation for I/O of vector members - vector members are treated analogous to the reference vectors,i.e. streamed as one large vector per collection
  - updated tests/datamodel accordingly (using clang-format)
- 2018-11-30 Frank Gaede (PR#37)
  - handle references and vector members in collection's ostream operators
- 2018-11-30 Frank Gaede (PR#36)
  - add github templates for releaase notes, issues and contribution guidelines
  - add ReleaseNotes.md - contains all commit logs so far (v00-08)

Vector Members in PODIO

- strictly speaking vector members break the *PODness* of the data classes
- treat vector members analogous to the reference vectors:
  - hold vector in *Object*-layer, start/end in *POD*-layer and iterator in *User*-layer
  - stream vector members as one large vector per collection
  - after reading keep in one large vector

```
plcio::LCGenericObject:
  Description: "LCIO LCGenericObject"
  Author : "F.Gaede, DESY"
  Members:
    - int isFixedSize              // all objects have a fixed size
    - std::string typeName         // type name of the user class
    - std::string dataDescription // data description string
  VectorMembers:
    - int    intVals    // integer value at given index.
    - float  floatVals  // float value at given index.
    - double doubleVals // double value at given index.
```

LCIO EDM in PODIO: pLCIO

- implemented almost **complete LCIO EDM** in PODIO
  - dedicated package **pLCIO**: https://stash.desy.de/projects/IL/repos/plcio
  - yaml file with LCIO EDM:
    https://stash.desy.de/projects/IL/repos/plcio/browse/config/lcio.yaml
  - example job from original LCIO:
    https://stash.desy.de/projects/IL/repos/plcio/browse/examples/simjob.cpp
- original idea to be able to create classes that are almost 100% backward compatible did not fully work out
- true for most of the actual member functions of the EDM classes
- not true for handling of collections and collection types, creation of objects, user defined parameters, ...
- nevertheless a transition from LCIO to pLCIO would be feasible at '*reasonable cost*'

Next Steps and Plans

implement direct binary I/O making use of array-of-POD - **still pending**

- use new '*thread safe*' implementation of the SIO layer from LCIO (see talk R.Ete)
- like to benchmark the reading performance against current ROOT I/O
  - Note: plan to also benchmark against **pLCIO existing LCIO**

modify the treatment of constness

- current implementation has extra types for *const* objects, e.g. *ConstMCParticle*
- prototype implementation exists that ensures constness transparently (B.Hegner)
  - still just needs to be merged with *master* branch

have HDF5 I/O layer

- started GSoC project under umbrella of HSF (G.Stewart)
- https://hepsoftwarefoundation.org/gsoc/2019/proposal_PODIOHDF5.html

Summary and Outlook

- EDM toolkit PODIO developed in context of FCC and LC with general HEP in mind
- have now formally fulfilled AID2020 requirements:
  - D3.4 EDM toolkit
  - MS90 Application to LC: pLCIO
- PODIO is actively used by FCC
- transition from LCIO to pLCIO for LC not yet decided

Outlook

- develop alternative binary I/O (using new SIO)
- hope to get a working HDF5 I/O implementation from *GSoC*
- start initial benchmarking of LCIO against pLCIO

Short Note on DESY ACTS activities

- started to look into ACTS and adopting it to the LC tracking code by implementing the MarlinTrk interface, that has been developed in the first AIDA project and that is also implemented by aidaTT
- it turned out however, much to our surprise, that the currently available ACTS code does not yet offer even the simplest implementation of a Kalman-Filter, something that would of course be absolutely required for any meaningful application of ACTS to LC physics studies
- therefore we had decided to postpone the implementation of ACTS for LC
- it is at this moment unclear what will be possible with ACTS in the last year of AIDA2020
- in any case we will follow the development closely and **start to contribute to ACTS**

Links and Pointers

- GitHub repository + docs:
  - https://github.com/aidasoft/podio
- doxygen page:
  - https://fccsw.web.cern.ch/fccsw/podio/index.html
- issue tracker (use Github issues):
  - https://github.com/aidasoft/podio
- plcio (EDM for LCIO w/ podio) git repository:
  - https://stash.desy.de/projects/IL/repos/plcio
- PODIO Library Design Document:
  - http://cds.cern.ch/record/2212785