Proposal for Data Model improvements

FCC software meeting

July 28, 2016

Joschka Lingemann
EP-SFT - CERN

Reminder on PODIO and EDM

PODIO: Plain old data input / output

- Store information in simple structs
- No object inheritance!
- Objects can have members, OneToOneRelation and OneToManyRelation (new)

Having no inheritance has some consequences

- Re-usable code: Composition to re-gain some of the benefits
- Disallow changes after creation: Again composition (tag + jet)

Components (or PODs)

Helpers

```
fcc::BareHit:
  Cellid : ulonglong
  Energy : float
  Time : float
  Bits : unsigned

fcc::BareCluster:
  Energy : float
  Time : float
  Position : fcc::Point
  Bits : unsigned

fcc::BareParticle:
  Type: int
  Charge: int
  Status: unsigned
  Vertex: fcc::Point
  P4: fcc::LorentzVector
  Bits: unsigned

fcc::BareJet:
  P4: fcc::LorentzVector
  Area: float
  Bits: unsigned
```

for DD4hep compatibility need `ulonglong` (was unsigned)

Actual Primitives

```
fcc::LorentzVector:
  Mass : float
  Px : float
  Py : float
  Pz : float

fcc::Point:
  X : float
  Y : float
  Z : float
```

to ease writing of generic algorithms operating on Bare* instead of MCParticle vs Particle, etc.

What exactly do we mean with `BareParticle::Type`? I assume HepMC

Add PDGId?

Generated event

Status quo:

- MC particle with relation to vertices
- Vertex with timing and position
- Association mother to daughter

```
fcc::MCParticle:
  […]
  Members:
   - fcc::BareParticle Core
  OneToOneRelations:
   - fcc::GenVertex StartVertex
   - fcc::GenVertex EndVertex

fcc::GenVertex:
  […]
  Members:
   - fcc::Point Position
   - float Ctau
```

Proposal:

- Drop the associations and reconstruct on demand as graph
- Or move to OneToManyRelation

```
fcc::MCParticleAssociation:
 […]
 OneToOneRelations:
  - fcc::MCParticle Mother
  - fcc::MCParticle Daughter
```

deprecate?

Hit, BareHit, Cluster and Sim*

BareHit stores position by CellId:

- may need to extend (next slide)

HitClusterAssociation:

- Not needed, can add

  OneToManyRelation to Cluster

Make things more uniform? Add SimTrackHit?

- raises questions down the road

  (more later)

```
fcc::BareHit:
  Cellid : ulonglong
  Energy : float
  Time : float
  Bits : unsigned
```

```
fcc::CaloHit:
  […]
  Members:
   - fcc::BareHit Core

fcc::CaloCluster:
  […]
  Members:
   - fcc::BareCluster Core
  OneToManyRelations :
   - fcc::CaloHit Hits
```

Hit position

Within FCCSW: Can get position with geometry

- DD4hep has `f(cellId) -> position`
- Do we need something for standalone?
- Could simplify some of the early-on studies?

Simplest solution that can be “dropped” if not needed:

- PositionedHit: Position as member, OneToOneRelation to Hit

Tracks and additional tracking information

Can deprecate some associations

If we go with SimTrackHit:

- Would also need SimTracks

  (again, more later)

```
fcc::TrackState:
  […]
  Members:
   - float Location
   - float Omega
   - float D0
   - float Z0
```

```
fcc::Track:
  […]
  Members:
   - float Chi2
   - unsigned Ndf
   - unsigned Bits
  OneToManyRelations :
   - fcc::TrackCluster Clusters
   - fcc::TrackState States
```

previously separate associations

Particles

Previously we had:

- Particle + Associations to tracks and clusters

```
fcc::Particle :
  […]
  Members :
   - fcc::BareParticle Core
  OneToManyRelations:
   - fcc::Track Tracks
   - fcc::CaloCluster Clusters
```

Could be changed to:

- Particle with relations to tracks and clusters
- Problematic(?): Need SimParticle for SimClusters and SimTracks

SimHits & Co

Why SimHits?

- Used to store (eventual) additional info only available in simulation

Another possible approach (again, composition):

- Add SimHit with members for additional info and OneToOneRelation to Hit?
  - Simple to “drop” if not needed…
  - Solves previously mentioned issues
- No need of BareHit anymore(?)

Tags

Proposal: Drop associations

- Just keep the “tag” collection

  and add relation to particle / jet
- Rename? (more later)

```
fcc::TaggedJet :
  […]
  Members :
   - float Tag
  OneToOneRelations :
   - fcc::Jet Jet
```

Float covers both use cases

(had separately int and float)?

Reconstructed Vertices

Associating tracks and vertices

- VertexTrackAssociation
  - No real way around?
  - Array of weights and OneToManyRelation for track?

    Error prone..

Could reshuffle information to make

usage simpler…

Reconstructed Vertices (cont.)

```
fcc::Vertex:
 […]
 Members:
   - float Chi2
   - unsigned Ndf
   - fcc::Point Position
   - unsigned Bits

fcc::VertexTrackAssociation:
  […]
  Members:
   - float Weight
  OneToOneRelations:
     - fcc::Track Track
     - fcc::Vertex Vertex
```

status quo

```
fcc::Vertex:
 […]
 Members:
   - float Chi2
   - unsigned Ndf
   - fcc::Point Position
   - unsigned Bits
 OneToManyRelations:
   - fcc::WeightedTrack Tracks

fcc::WeightedTrack:
  […]
  Members:
   - float Weight
  OneToOneRelations:
     - fcc::Track Track
```

Some renaming to make usage clearer

The association collections seem to imply (example particle):

- Run over ParticleCollection
- Search ParticleTagAssociationCollection to get the Tag

If we rename to “TaggedParticleCollection” the user may be more inclined to:

- Run over TaggedParticleCollection
- Get the particle through reference

N.b.: Capitalisation of members? Would like to go to lower case (naming conventions)

Summary

Proposed changes due to recent additions in PODIO:

- Associations can mostly be deprecated

Rename remaining associations to make usage clearer

Add some convenience (hit position) collections

Will open PR incl. input from this discussion:

- Also worked on documentation explaining our choices

Related

Changes in PODIO:

- Optionally any members of POD members can be exposed to the object
  - Solves some of the transparency issues
  - No need to check the “*Data” object
- Some fixes to relations
  - Needed for integration of DAG