ePIC RICH Simulation Synergies

Christopher Dilks

EIC RICH Consortium
Inaugural Meeting

16 June 2023

Outline

- **Overview of ePIC Software**
- **Geometry**
- **Reconstruction**
  - Data Model
  - Algorithms
- **Benchmarks**

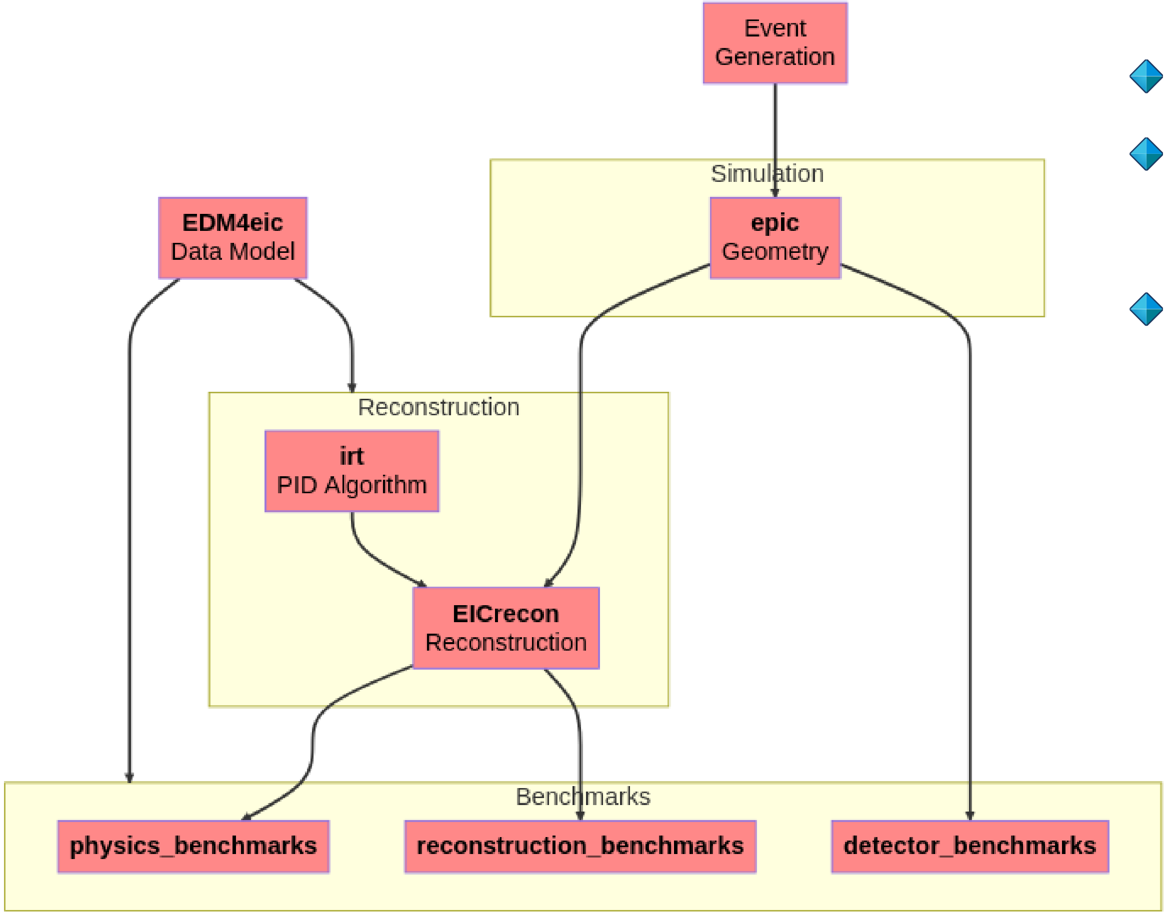
Opportunities for collaboration and synergy will be discussed along the way

Outline

- **Overview of ePIC Software**
- **Geometry**
- **Reconstruction**
  - Data Model
  - Algorithms
- **Benchmarks**

Opportunities for collaboration and synergy will be discussed along the way

Overview of ePIC Software: Simulation



- Repository names in **bold**
- All in the EIC Organization on Github
  - Except for benchmarks, on EICweb
- For Write Access:
  - Become a member
  - Then *after* you have access,
    Join the ePIC Devs team

drich-dev: Developer Tools for the dRICH

- **https://github.com/eic/drich-dev**
  - “Gateway” to ePIC software for the dRICH
  - Includes (historical) support for the pfRICH
  - Aims to be a starting point for Cherenkov simulations and reconstruction in general
- **dRICH Tutorial Series**
  - https://github.com/eic/drich-dev/blob/tutorial/doc/tutorials/README.md
  - Not *completely* focused on the dRICH
  - Weekly, Friday at 10AM US/Eastern
    - See https://indico.bnl.gov/event/19679/ for the first one
  - Tutorial #2 of 6 starting in ~1 hour
    - https://indico.bnl.gov/event/19680/
    - Meeting ID and PW: 91350671308 357599

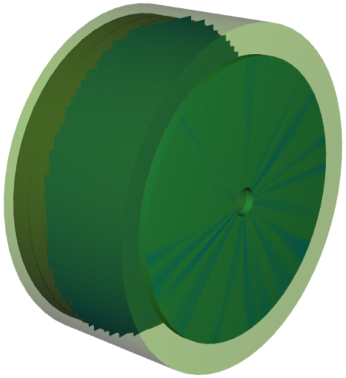
Outline

- Overview of ePIC Software
- **Geometry**
- Reconstruction
  - Data Model
  - Algorithms
- Benchmarks

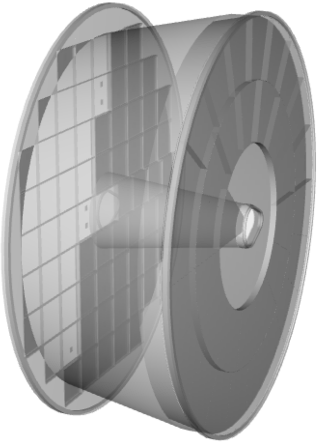
Geometry: Based on DD4hep

https://github.com/eic/epic

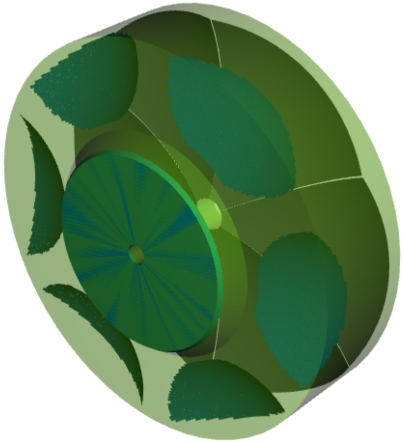
pfRICH
(legacy DD4hep)



pfRICH
(GDML → DD4hep)



dRICH



Synergy

- Shared material and surface properties
- Shared common definitions
- Try to keep conventions the same between the detectors, where applicable (e.g., dRICH and pfRICH)
- Share bug fixes and improvements

pfRICH in DD4hep

- **Legacy DD4hep design**
  - Used in ATHENA, re-scaled for ePIC
  - Not really used nowadays… but it *also* serves as a standalone RICH example in the DD4hep software itself, to help guarantee stability of Cherenkov physics for all DD4hep users (beyond ePIC)
- **Standalone Geant4 pfRICH:**
  - Export to GDML → Import in DD4hep
  - Things to think about:
    - Sustainability: this GDML creation should be reproducible by the `epic` DD4hep code
    - Global Connection: the pfRICH geometry creation should use the global geometry parameters of ePIC, so its positioning and size can be set and read by `epic`
    - Activation: the GDML pfRICH sensors should be made DD4hep-sensitive and have a DD4hep readout, similar to the dRICH sensors
    - Alternative: "port" the standalone pfRICH to DD4hep

Outline

- Overview of ePIC Software
- Geometry
- **Reconstruction**
  - Data Model
  - Algorithms
- Benchmarks

Reconstruction Framework: EICrecon (JANA2 based)

<u>Two types of Objects:</u>

https://github.com/eic/EICrecon

🔷 **Collection**

- A set of objects, such as “digitized hits”, or “PID hypotheses”
- Defined as “datatype” in the Event Data Model (EDM) – see next slides

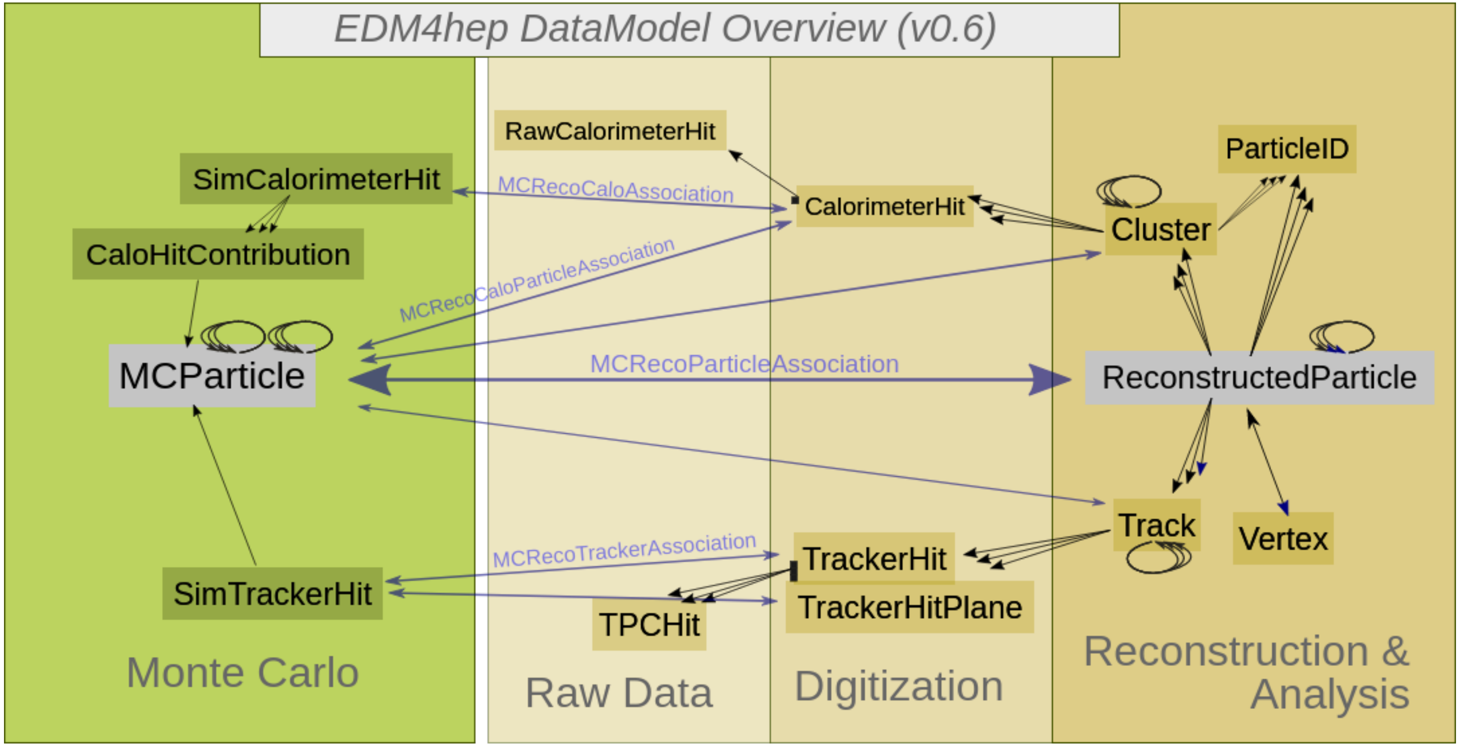
🔷 **Algorithm**

- An algorithm transforms collection(s) into collection(s)
- Examples:
  - Digitizer
    - Input: truth-level simulated hits
    - Output: digitized raw hits

- Algorithms should be:
  - Configurable – allow (external) configuration to tune for specific use cases or subsystems
  - Focused – don’t write a monolith
  - Shareable – some algorithms can be useful for multiple subsystems
  - Not dependent on EICrecon or JANA2 – Modularity → Standalone IRT is an example
  - See Sylvester’s CHEP 2023 talk

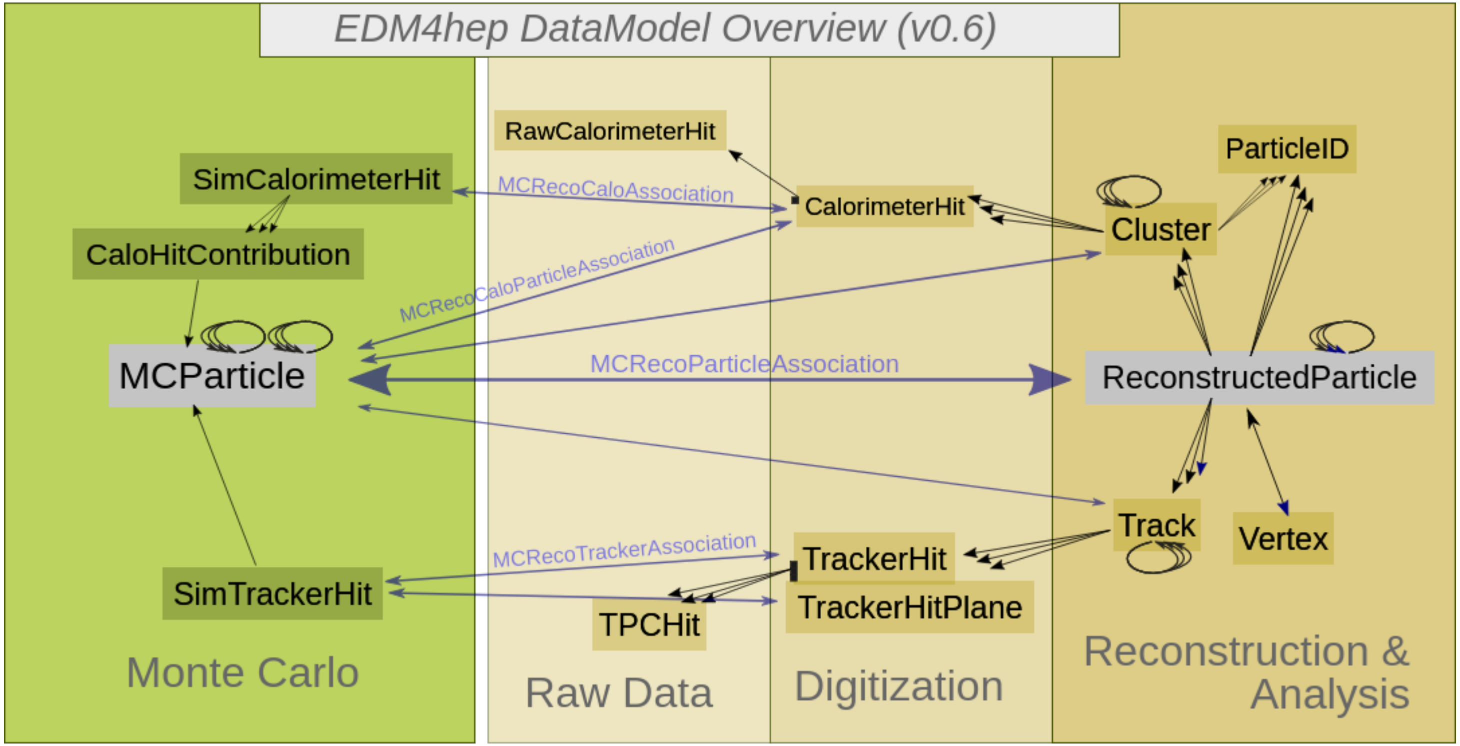
Event Data Model (EDM) – at the General level

- **EDM4hep: https://github.com/key4hep/EDM4hep**
  - General data model shared by several HEP experiments



Event Data Model (EDM) – at the General level

- **EDM4hep: https://github.com/key4hep/EDM4hep**
  - General data model shared by several HEP experiments

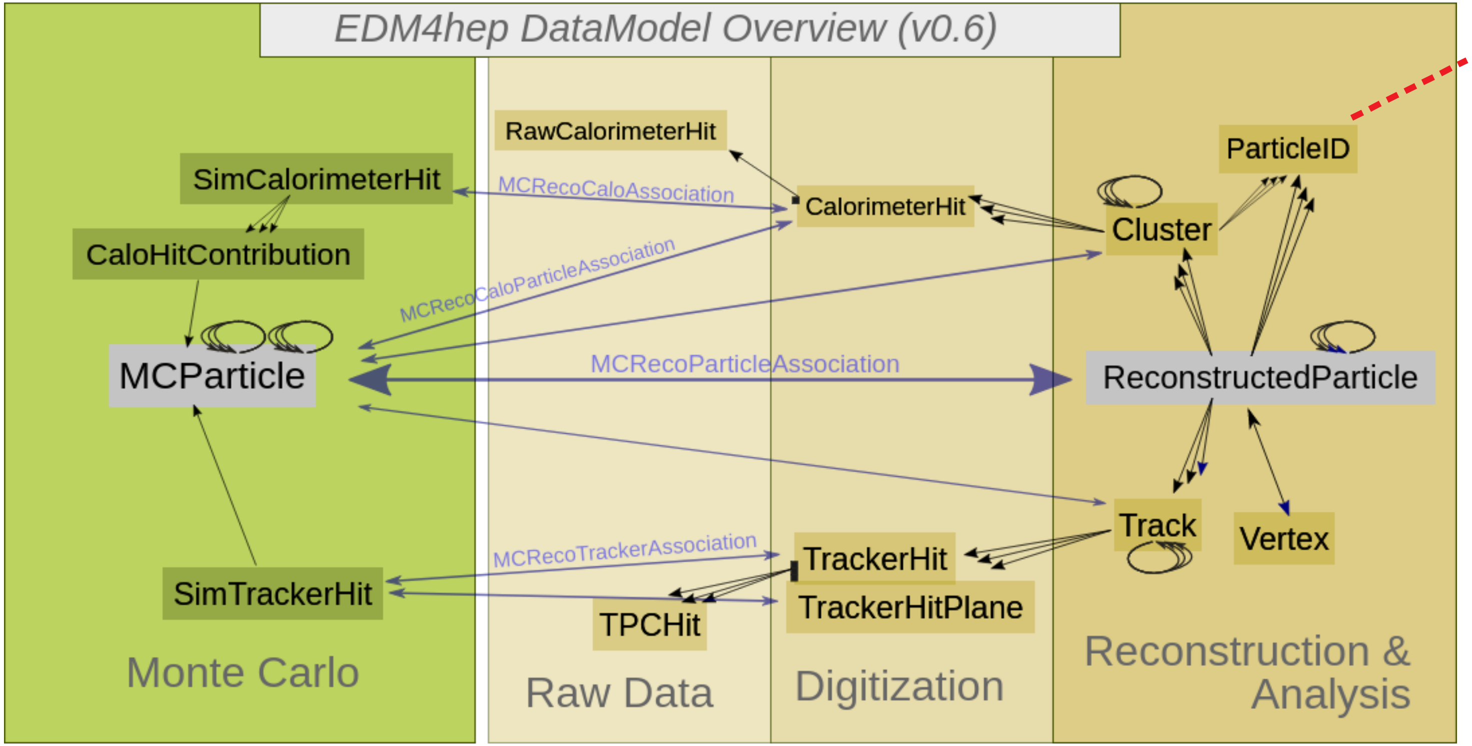


In output ROOT files, each **datatype** becomes a **TTree branch**

(can also use a PODIO frame reader)

Event Data Model (EDM) – at the General level

- **EDM4hep: https://github.com/key4hep/EDM4hep**
  - General data model shared by several HEP experiments



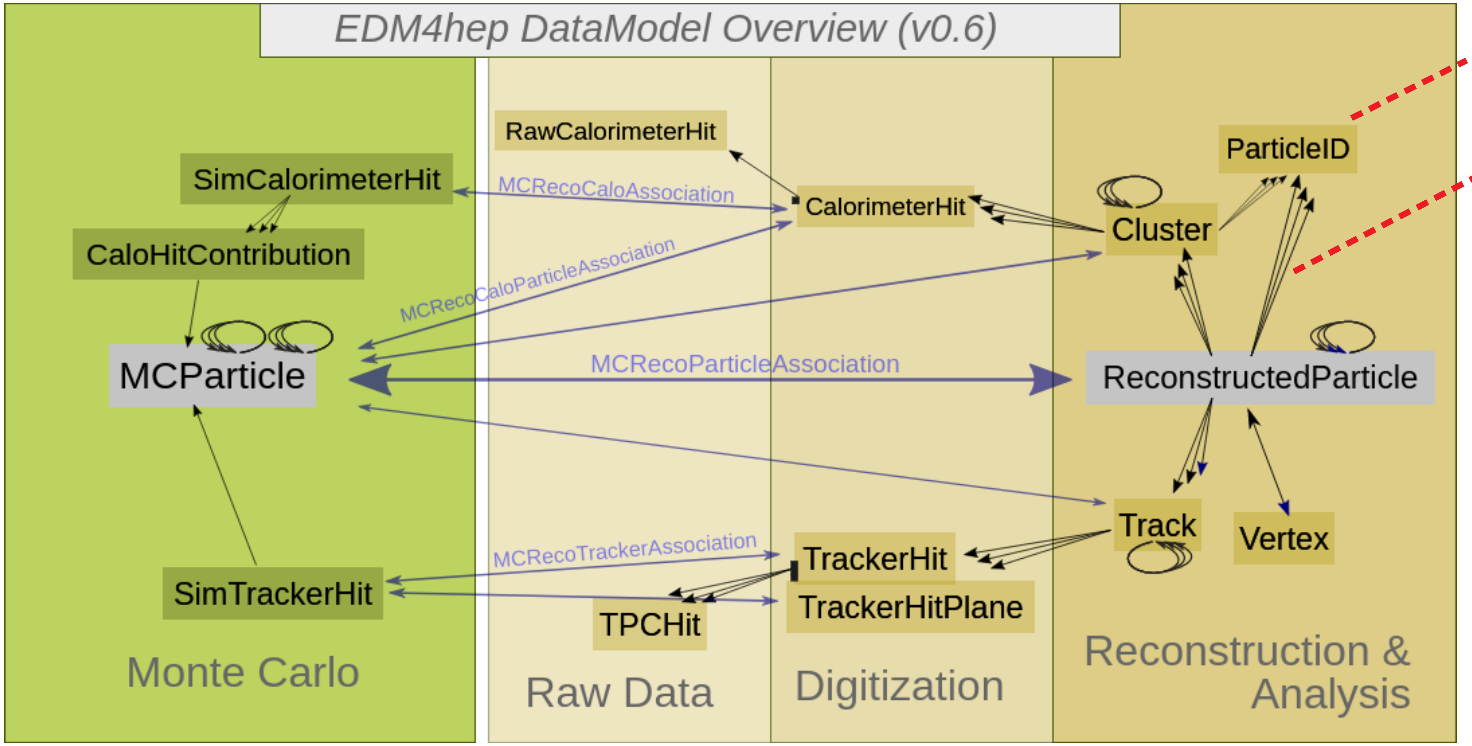
- "**ParticleID**" is the main datatype for PID

In output ROOT files, each **datatype** becomes a **TTree branch**

(can also use a PODIO frame reader)

Event Data Model (EDM) – at the General level

- **EDM4hep: https://github.com/key4hep/EDM4hep**
  - General data model shared by several HEP experiments



- "**ParticleID**" is the main datatype for PID
- One-to-many relation from "**ReconstructedParticle**" datatype to "**ParticleID**"

In output ROOT files, each **datatype** becomes a **TTree branch**

(can also use a PODIO frame reader)

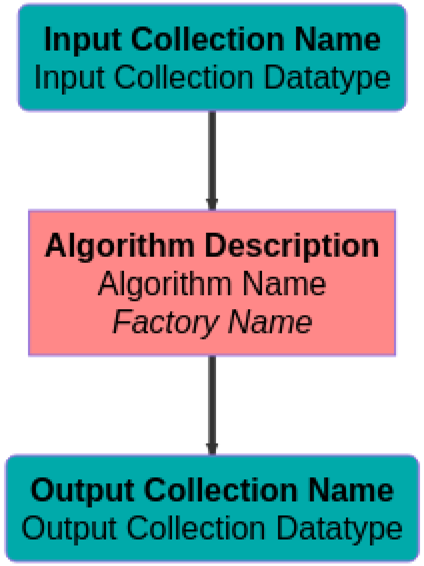
EDM – at the EIC level

- **EDM4eic: https://github.com/eic/EDM4eic**
  - Experiment-specific data model; extends EDM4hep
  - Allows deviations from EDM4hep, where needed, e.g.,
    - edm4hep::ReconstructedParticle vs. edm4eic::ReconstructedParticle
    - Custom datatype for Cherenkov physics
    - Custom datatype for TOF physics

To view the data models, see the YAML files:

- EDM4hep: https://github.com/key4hep/EDM4hep/blob/master/edm4hep.yaml
- EDM4eic: https://github.com/eic/EDM4eic/blob/main/edm4eic.yaml

Algorithms & Collections → Reconstruction Building Blocks

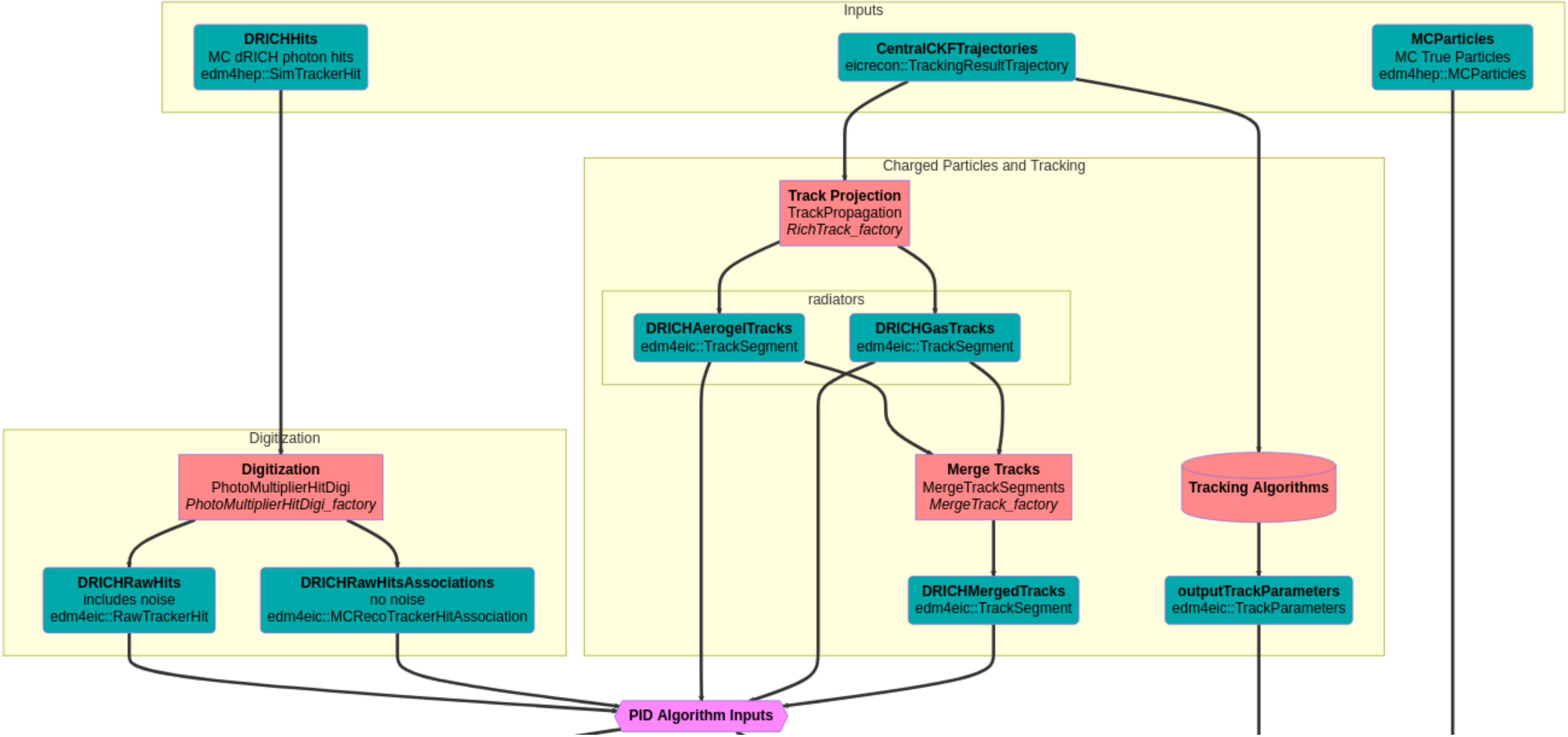


Synergy:

- EDM4* Datatypes are shared anywhere
- Algorithms can also be shared
- Let’s work together!

dRICH PID Plugin: Algorithm Flowchart – Part 1 of 3

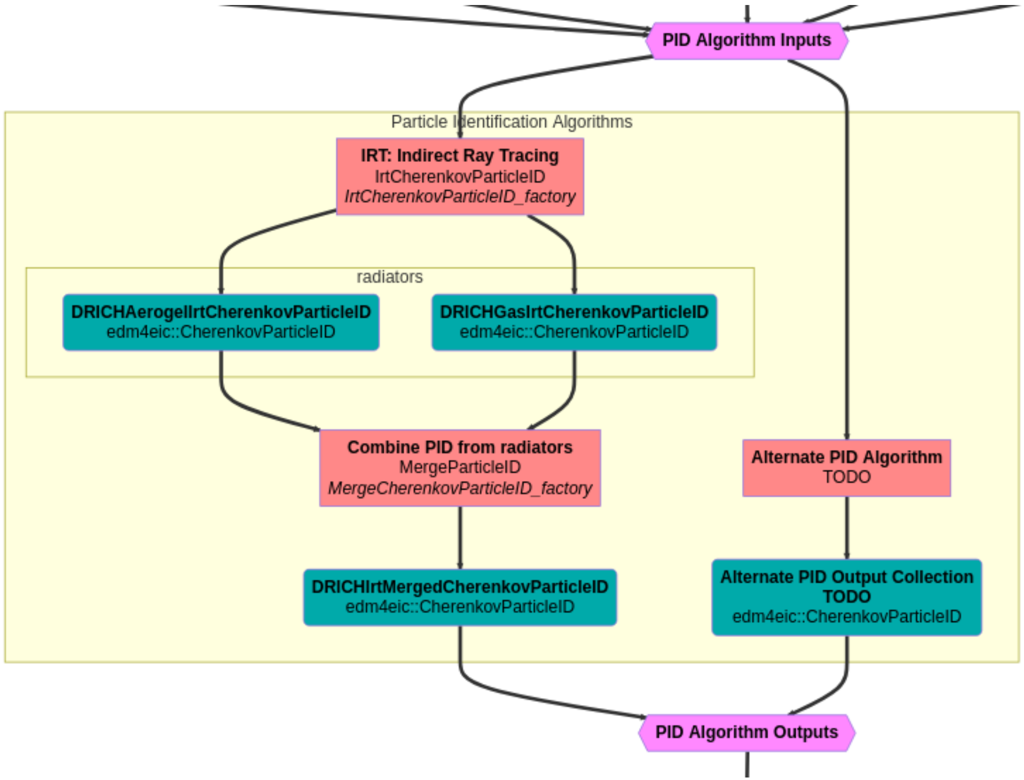
Click here for enlarged version and more



dRICH PID Plugin: Algorithm Flowchart – Part 2 of 3

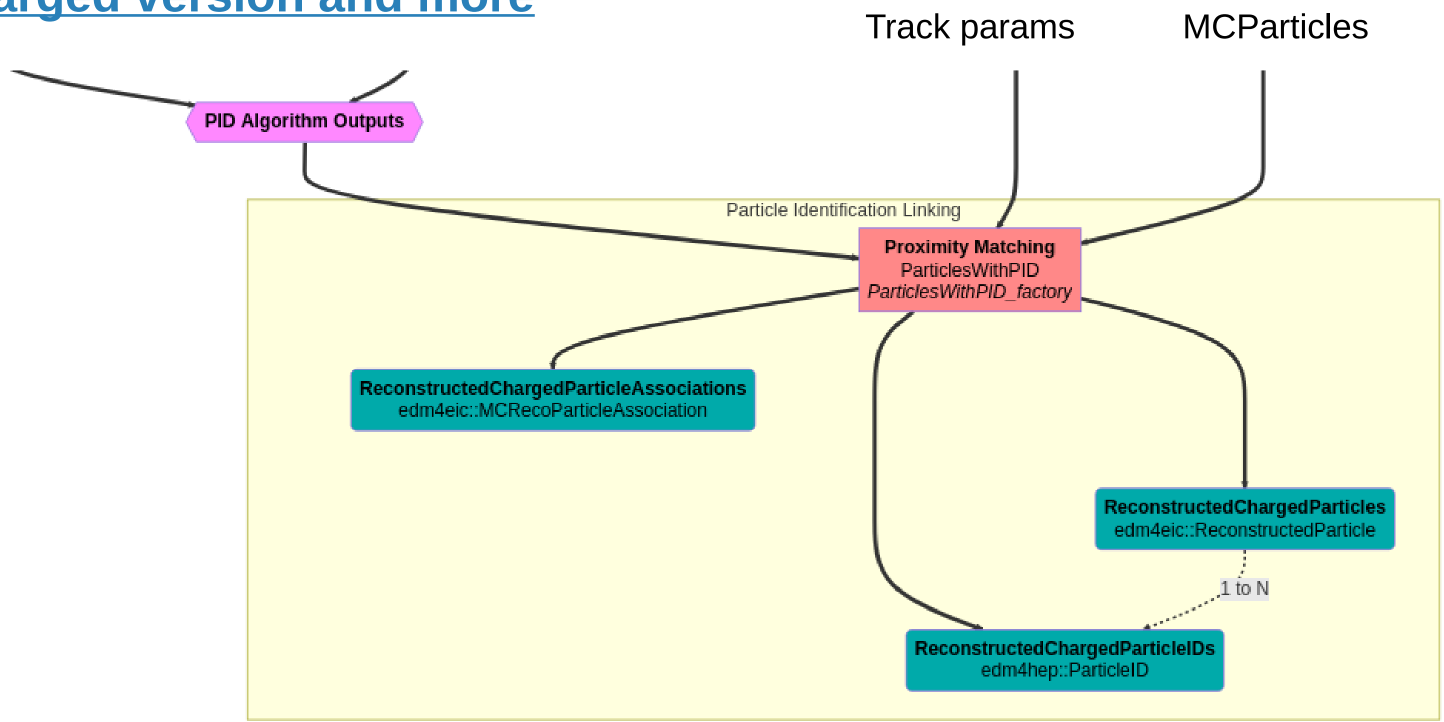
Click here for enlarged version and more

Track params MCParticles



dRICH PID Plugin: Algorithm Flowchart – Part 3 of 3

[Click here for enlarged version and more]



Algorithm: Digitization

Configuration Parameters: externally configurable

- **Common PMT Digitizer Algorithm**
  - Trigger parameters (gate, pedestal, etc.)
  - Quantum Efficiency
  - Empirical Safety Factor 70%
  - Sensor pixel gap mask (~88% survive)
  - Noise injection

  - **TODO:** Time over Threshold (ToT)
  - **TODO:** Refine configuration parameters

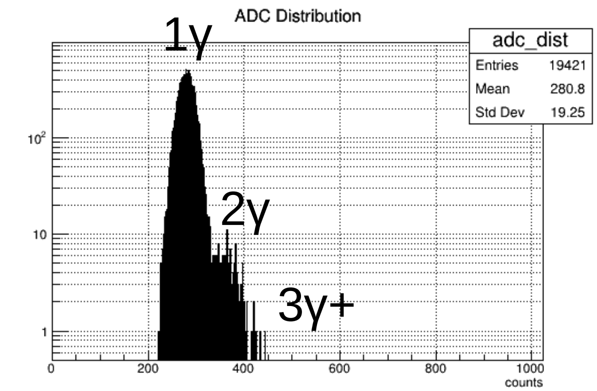
Synergy: this is a common algorithm anyone can use and improve

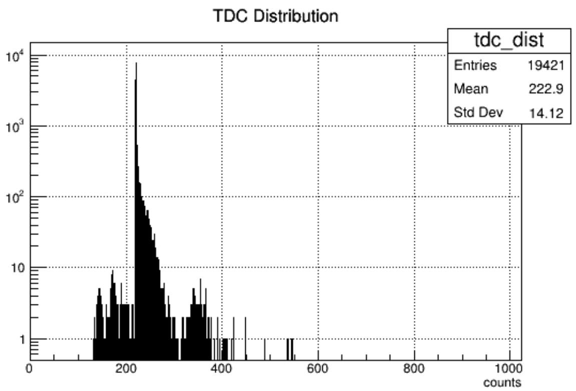
```
// RNG seed
unsigned long seed = 1;

// triggering
double hitTimeWindow  = 20.0;
double timeResolution = 1/16.0;
double speMean        = 80.0;
double speError       = 16.0;
double pedMean        = 200.0;
double pedError       = 3.0;

// noise
bool enableNoise        = true;
double noiseRate        = 20000; // [Hz]
double noiseTimeWindow = 20.0;   // [ns]

// SiPM pixels
bool   enablePixelGaps = true;
double pixelSize       = 3.0; // [mm]

// overall safety factor
double safetyFactor = 0.7;
```

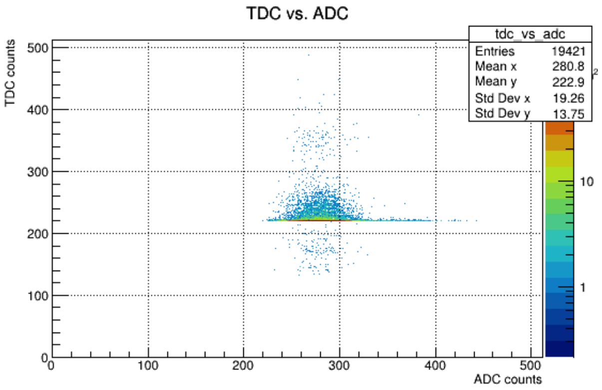
λ QE

```
{315,  0.00},
{325,  0.04},
{340,  0.10},
{350,  0.20},
{370,  0.30},
{400,  0.35},
{450,  0.40},
{500,  0.38},
{550,  0.35},
{600,  0.27},
{650,  0.20},
{700,  0.15},
{750,  0.12},
{800,  0.08},
{850,  0.06},
{900,  0.04},
{1000, 0.00}
```

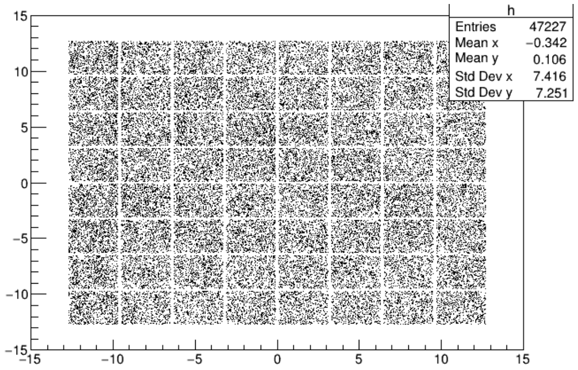
Algorithm: Digitization



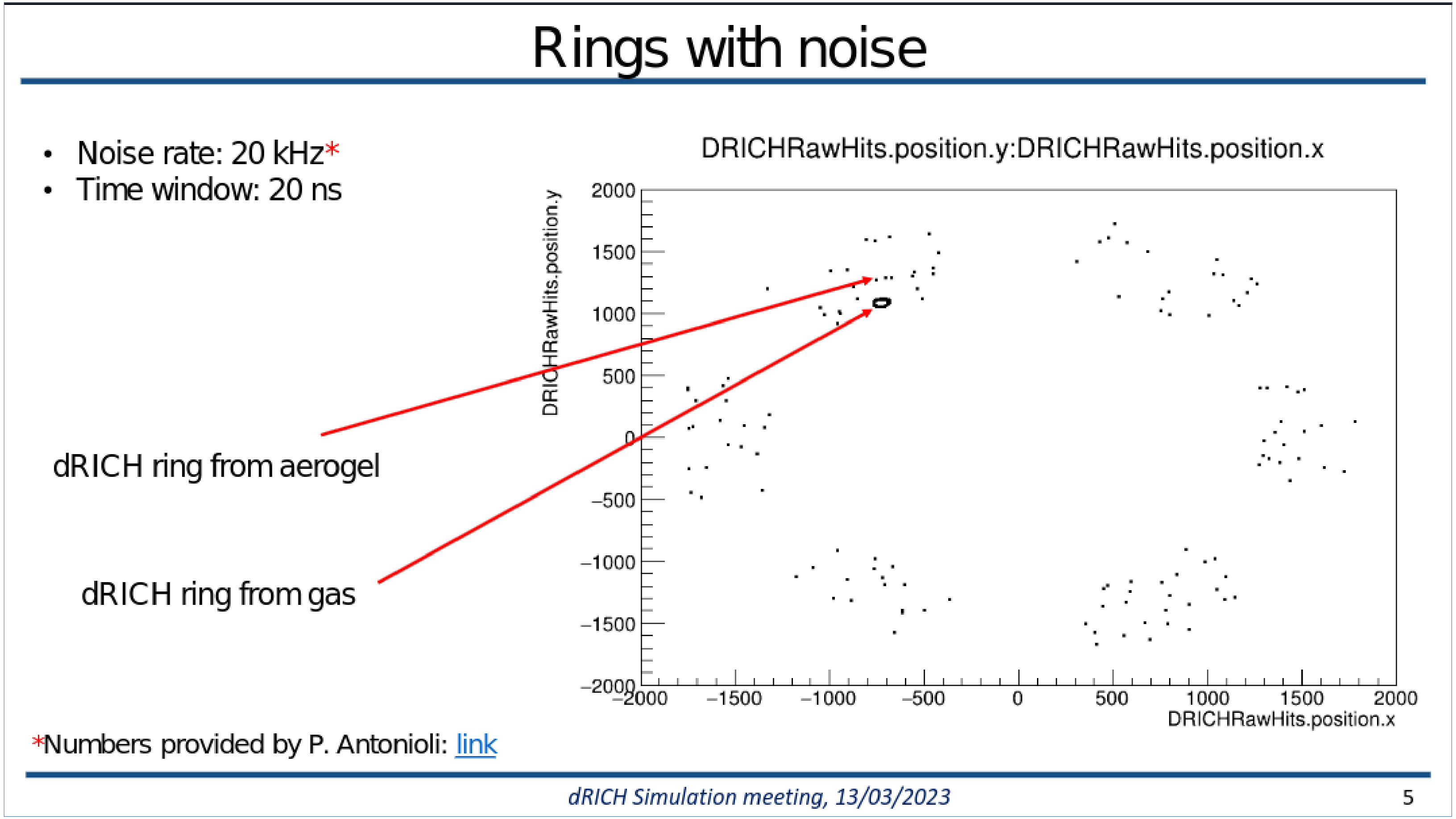




SiPM pixel gaps

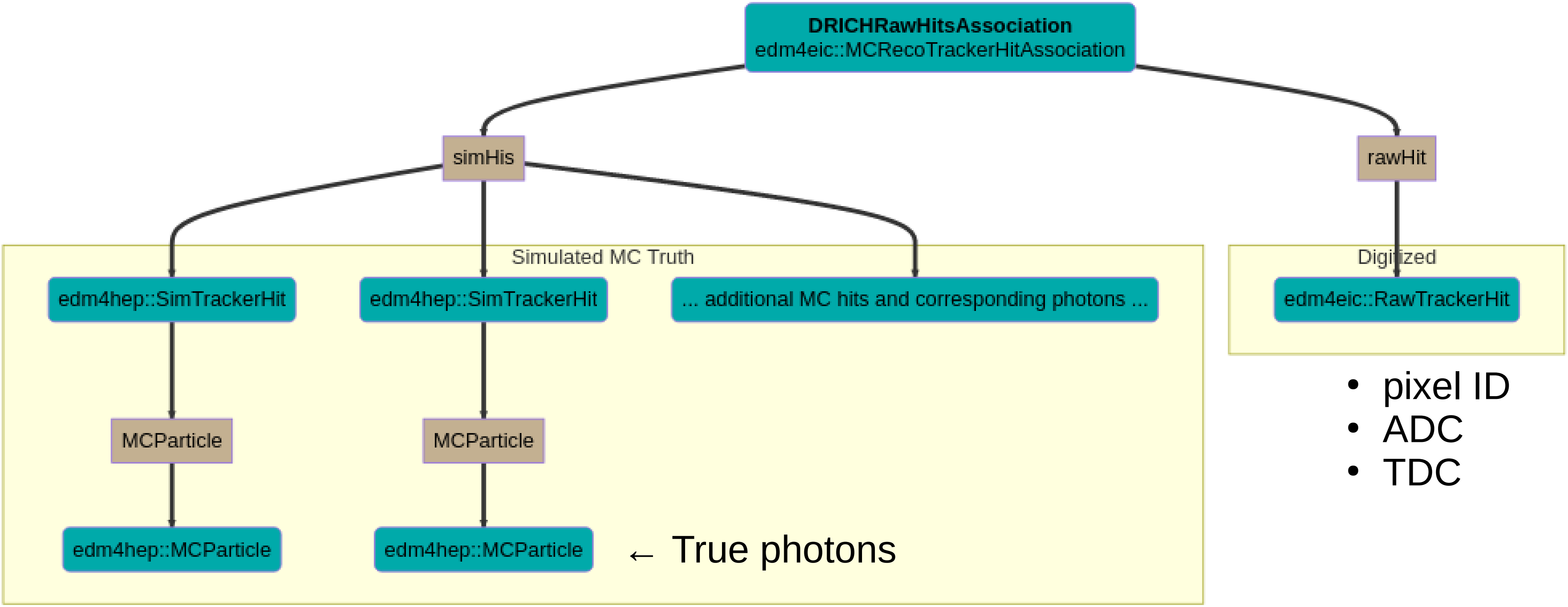


Algorithm: Digitization – Noise Injection



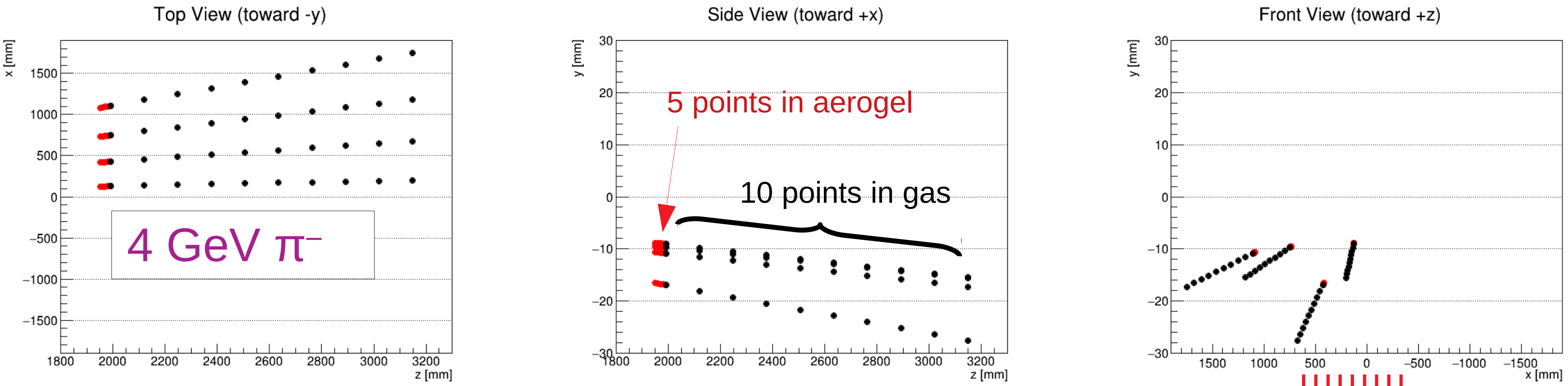
Slide from Luigi Dello Stritto

Data Model: Digitized Hits

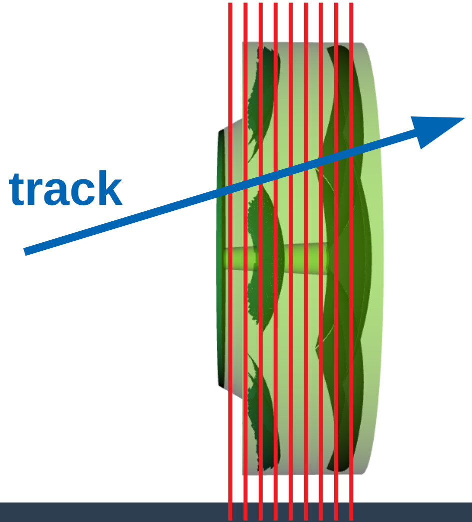


- pixel ID
- ADC
- TDC

Algorithm: Charged Particle Track Projection



- Example: 4 GeV pions in horizontal y=0 plane
- Propagate to xy-planes in the dRICH radiators
  - 5 planes in aerogel
  - 10 planes in gas
- **Reconstructed** track points in **Aerogel** and **Gas** (and the merged combination)
- Synergy: Can be used by the pfRICH and others



Data Model: Charged Particle Track Points

TrackSegment: a set of TrackPoints

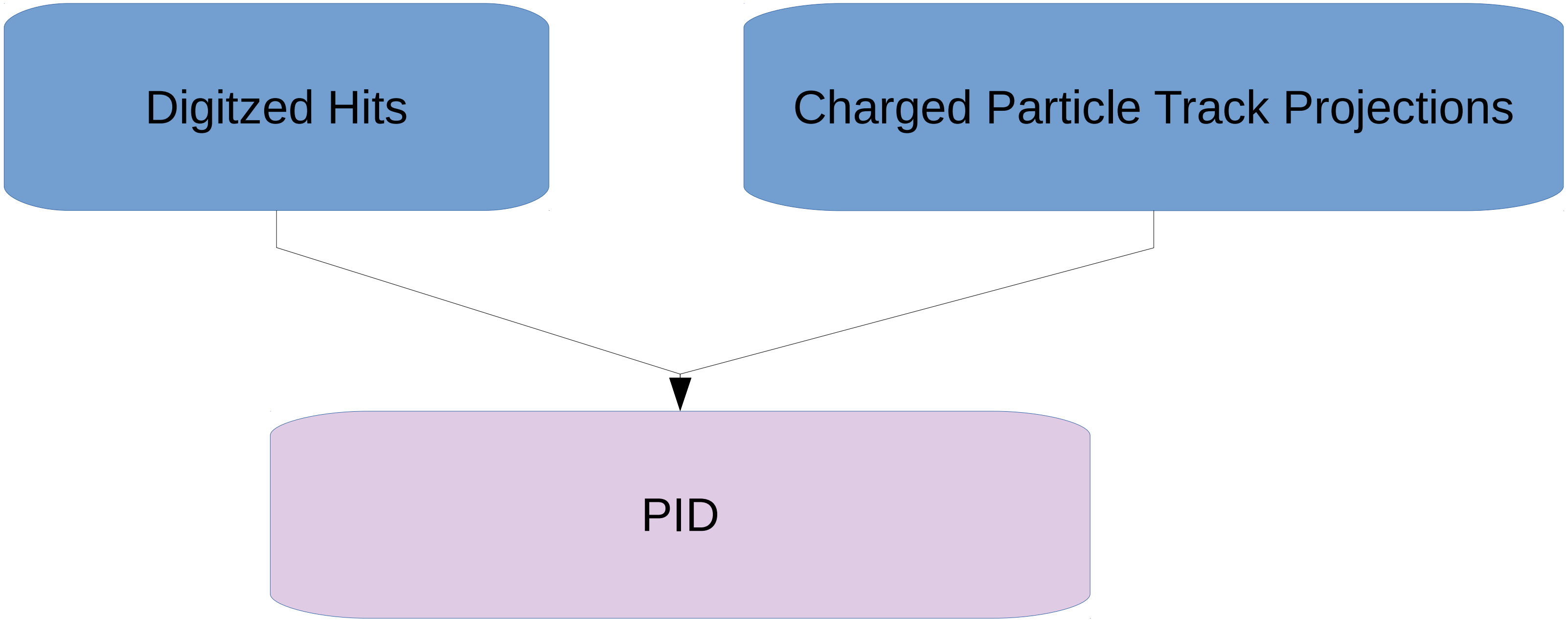
```
edm4eic::TrackSegment:
  Description: "A track segment defined by one or more points along a track."
  Author: "S. Joosten"
  Members:
    - float             length             // Pathlength from the first to the last point
    - float             lengthError        // Error on the segment length
  OneToOneRelations:
    - edm4eic::Track    track              // Track used for this projection
  VectorMembers:
    - edm4eic::TrackPoint points           // Points where the track parameters were evaluated
```

TrackPoints: the projected points:

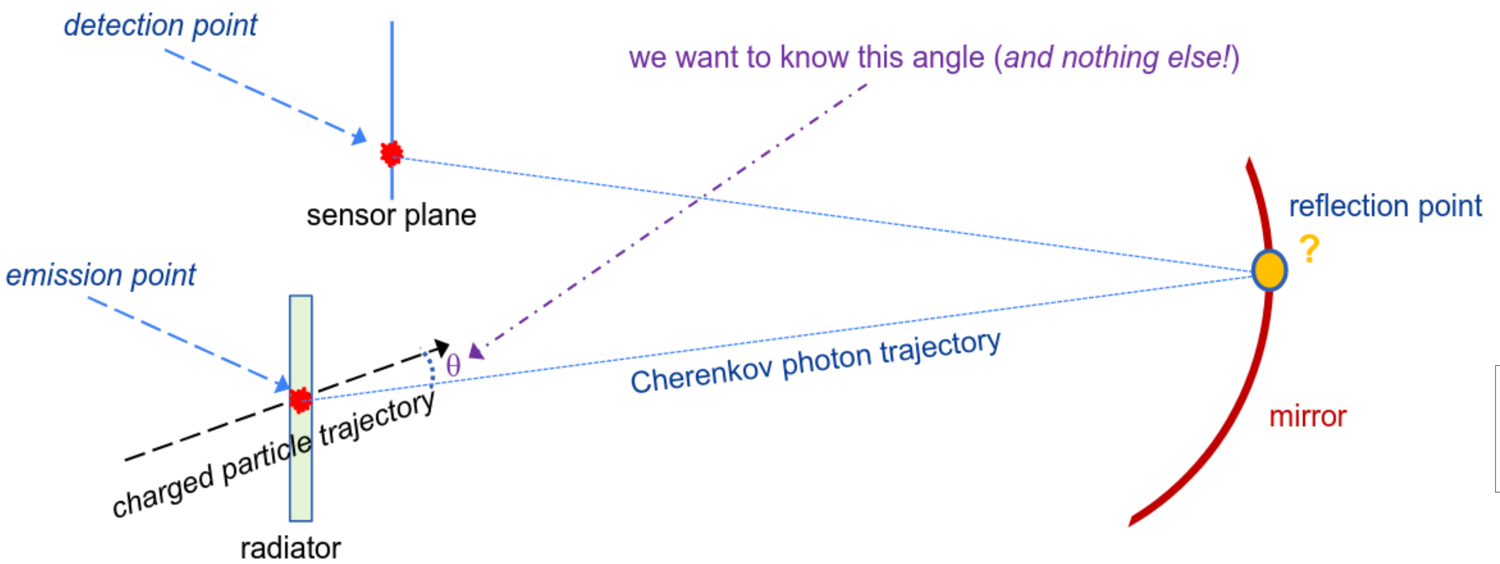
- position
- momentum
- time
- and more

```
## A point along a track
edm4eic::TrackPoint:
  Members:
    - edm4hep::Vector3f position        // Position of the trajectory point [mm]
    - edm4eic::Cov3f    positionError   // Error on the position
    - edm4hep::Vector3f momentum        // 3-momentum at the point [GeV]
    - edm4eic::Cov3f    momentumError   // Error on the 3-momentum
    - float             time            // Time at this point [ns]
    - float             timeError       // Error on the time at this point
    - float             theta           // polar direction of the track at the surface [rad]
    - float             phi             // azimuthal direction of the track at the surface [rad]
    - edm4eic::Cov2f    directionError  // Error on the polar and azimuthal angles
    - float             pathlength      // Pathlength from the origin to this point
    - float             pathlengthError // Error on the pathlength
```

PID Algorithm: Inputs



PID Algorithm: Indirect Ray Tracing (IRT)



Used by both dRICH and pfRICH

- Given sensor hits and optics, determine the photon emission angle, sampled along a charged particle trajectory
- Newton-Gauss iterative solver for optical path
- Compact, standalone library used for Geant4 and ATHENA
- Interfaced with EICrecon (and Juggler) for ePIC

https://github.com/eic/irt

Figures from Alexander Kiselev, From meeting on RICH Pattern Recognition Challenges
https://agenda.infn.it/event/30966/

PID Algorithm: Alternatives

- To be integrated with EICrecon
- **Synergy: The doors are open for development & integration!**
  - Inputs are available
  - Handling of outputs implemented
    - Working with Oskar Hartbrich – TOF & Cherenkov PID Synergy

Data Model: Cherenkov PID

TO BE IMPROVED
Opportunity for Synergy!

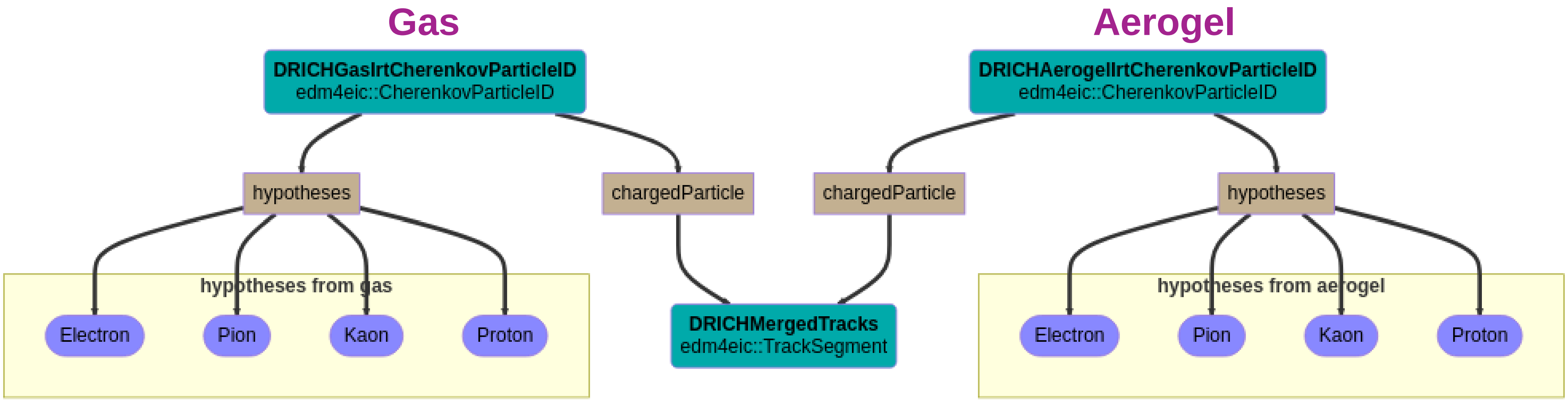
CherenkovParticleID datatype

```
edm4eic::CherenkovParticleID:
  Description: "Cherenkov detector PID"
  Author: "A. Kiselev, C. Chatterjee, C. Dilks"
  Members:
    - float             npe               // Overall photoelectron count
    - float             refractiveIndex   // Average refractive index at the Cherenkov photons' vertices
    - float             photonEnergy      // Average energy for these Cherenkov photons [GeV]
  VectorMembers:
    - edm4eic::CherenkovParticleIDHypothesis hypotheses        // Evaluated PDG hypotheses
    - edm4hep::Vector2f                      thetaPhiPhotons   // estimated (theta,phi) for each Cherenkov photon
  OneToOneRelations:
    - edm4eic::TrackSegment                  chargedParticle   // reconstructed charged particle
  OneToManyRelations:
    - edm4eic::MCRecoTrackerHitAssociation   rawHitAssociations // raw sensor hits, associated with MC hits
```

CherenkovPdgHypothesis component: one for each PDG (mass) hypothesis:

```
## PID hypothesis from Cherenkov detectors
edm4eic::CherenkovPdgHypothesis:
  Members:
    - int32_t           pdg              // PDG code
    - float             npe              // Overall p.e. count associated with this hypothesis for a given track
    - float             weight           // The weight associated with this hypothesis (the higher the more probable)
```

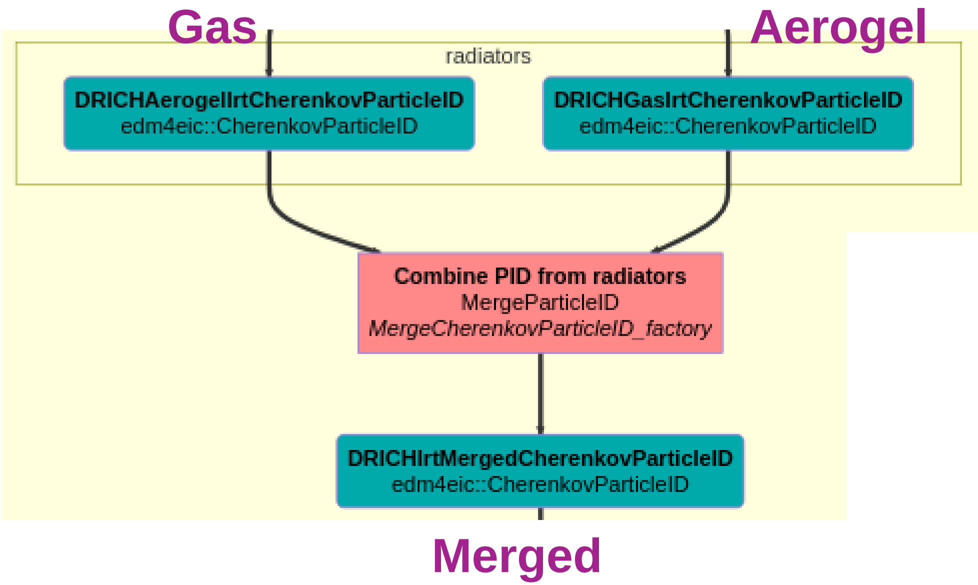
Data Model: Cherenkov PID



- One for each radiator (and one for the merged combination)
- All point to the same TrackSegment (as a unique ID)
- This is the “expert-level” PID object, specific for CherenkovPID

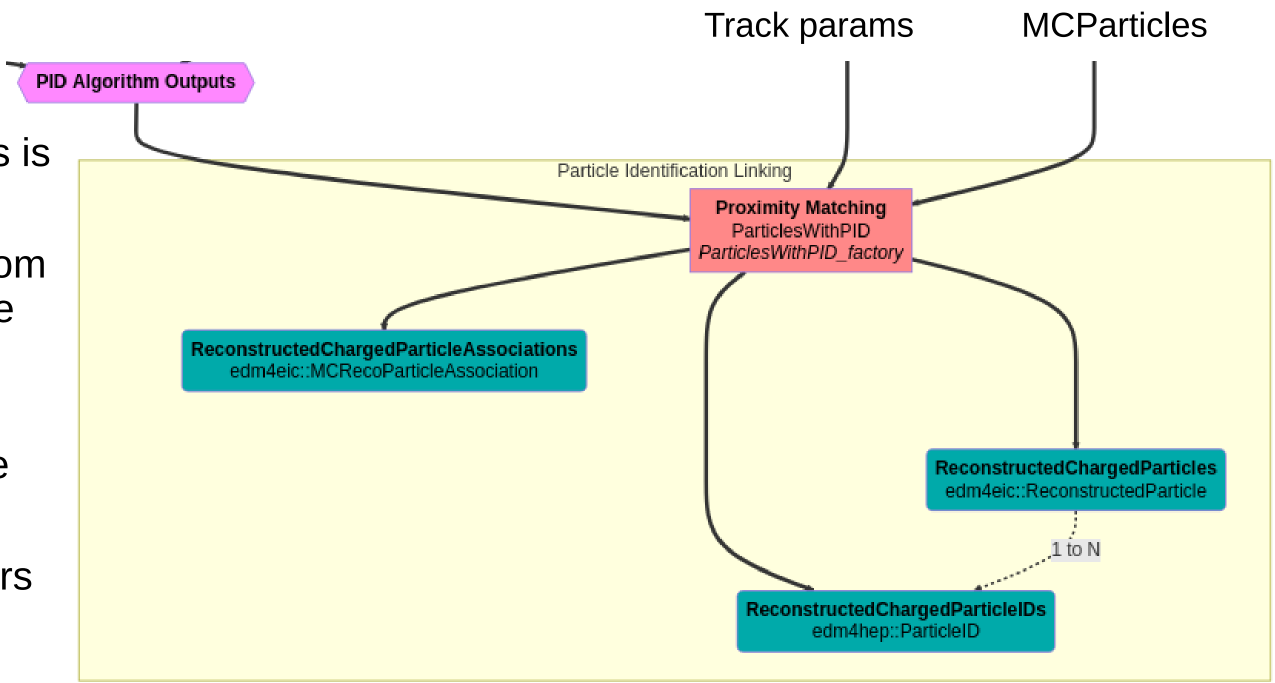
Algorithm: Merging Cherenkov PID Objects

- Simple Particle ID object merging implemented
- Currently handles merging dRICH gas + aerogel
- Could be generalized to merge PID objects from various subsystems



Algorithm: Linking to Reconstructed Particles

- Linking PID and reconstructed particles is (slightly) non-trivial….
- Track projections in dRICH originate from a non-EDM4hep/eic datatype, therefore cannot link back to it
- Workaround: proximity matching of the projected dRICH TrackSegments to the reconstructed particles (η,ϕ)
  - … which is also how track parameters are matched to MC Particles …
- At this stage, we also build the general-level PID objects, for non-experts
  - We could also merge PID results from other subsystems



Data Model: General PID

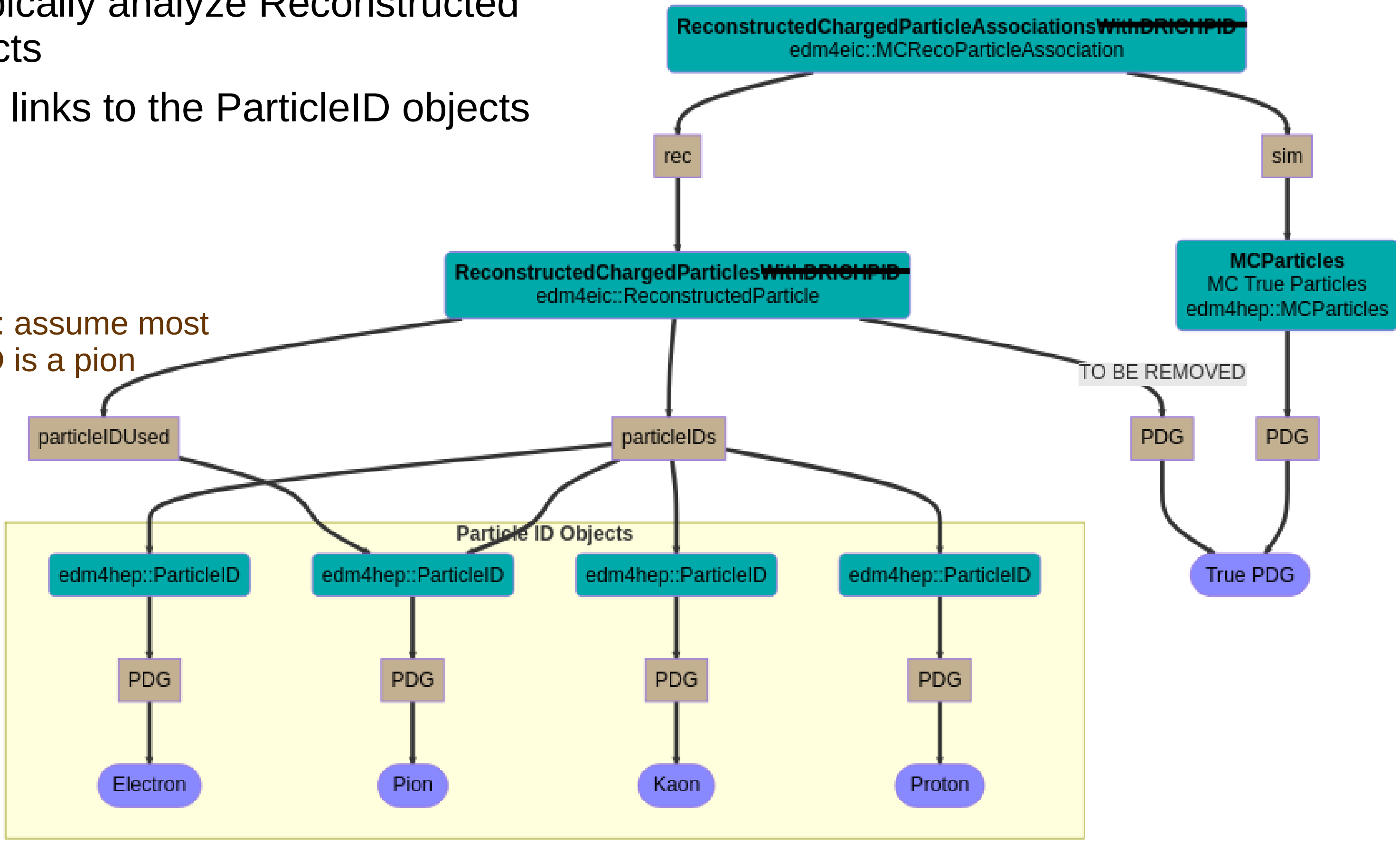
```
#---------- ParticleID
 edm4hep::ParticleID:
   Description:  "ParticleID"
   Author : "F.Gaede, DESY"
   Members:
     - int32_t   type             //userdefined type
     - int32_t   PDG              //PDG code of this id - ( 999999 ) if unknown.
     - int32_t   algorithmType  //type of the algorithm/module that created this hypothesis
     - float likelihood     //likelihood of this hypothesis - in a user defined normalization.
   VectorMembers:
     - float parameters     //parameters associated with this hypothesis.
```

- "**ParticleID**" is the main datatype for PID
- Used by many experiments, *<u>including</u>* ePIC
- This is the "user-level" PID object
- All PID subsystems should produce these objects as the final output

Data Model: General PID

- Users will typically analyze Reconstructed Particle objects
- They contain links to the ParticleID objects

Example: assume most likely PID is a pion



Outline

- Overview of ePIC Software
- Geometry
- Reconstruction
  - Data Model
  - Algorithms
- **Benchmarks**

Benchmarks

https://eicweb.phy.anl.gov/EIC/benchmarks

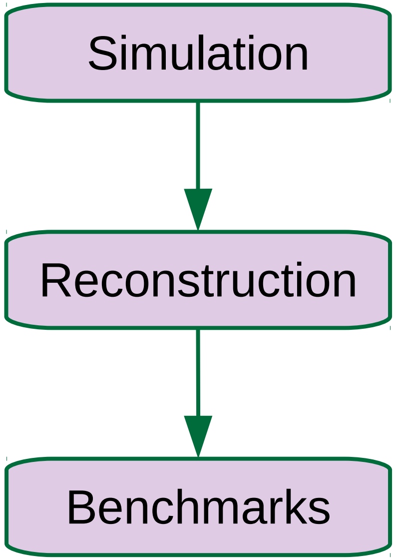
Continuous Integration

- Make a change in geometry or reconstruction, automatically see the impact everywhere on everything
- Benchmarks: validation, performance plots, anything that tells you that your subsystem is working as expected
- Critical for stability as we continue to improve detector and reconstruction design
  - And for stability against *any* change in the ePIC software stack or in any upstream software (dependencies)

dRICH Benchmarks

- Under development, working well but not yet triggered by upstream
- New paradigm proposal: “Analysis algorithms”: similar to reconstruction algorithms, these are also as independent as possible
- Synergy: Hopefully general enough to be used by the pfRICH

Continuous Integration (CI) Pipeline (simplified)



Summary

Geometry

Synergy

- Shared material and surface properties
- Shared common definitions
- Try to keep conventions the same between the detectors, where applicable (e.g., dRICH and pfRICH)

Data model
Reconstruction Algorithms
Benchmark Algorithms

Synergy:

- Datatypes are shared anywhere
- Algorithms can also be shared
- Let's work together!
- The doors are open for new algorithms, as well as improvements to current ones

- Share bug fixes
- Share improvements
- Share lessons learned
- Share tooling
- Avoid duplication of work
- Use collaborative tools on Github