

EIC Software Infrastructure Review

Data Model

Wouter Deconinck

*On behalf of the EPIC
Collaboration*

What is a Data Model?

- **The set of standardized data structures that we collectively agree to use to pass information between reconstruction algorithms**
- Example: The information we talk about when we say ‘a hit in a tracking detector,’ such as channel number, energy deposition, time, position, etc...

What is **not** included in this discussion of the data model?

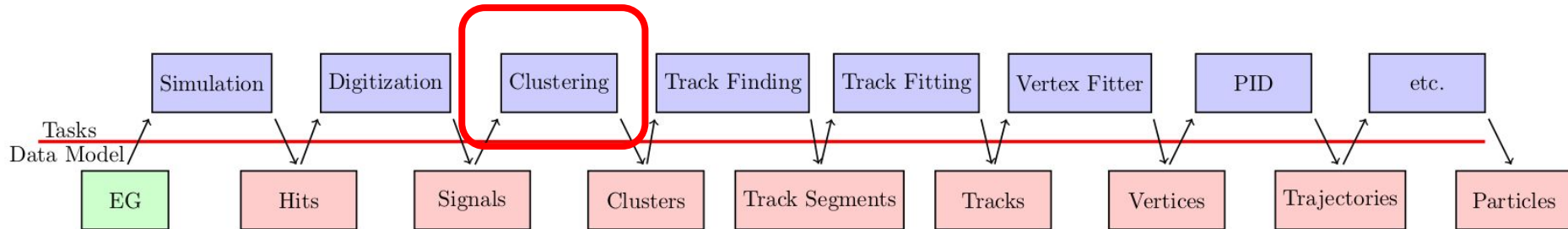
- Decisions about input/output file/memory formats, physical data storage medium: **we aim for flexibility through our choice of data model.**
- Example: Our choice of data model does not require storage in ROOT files (but can be written to ROOT files, HDF5 files, and many others), does not require C++ (or Python), does not require row-oriented memory layouts (but allows for GPU processing), etc...

The Motivation Behind a Standardized Data Model

Use of **standard interfaces** between individual simulation, reconstruction, and analysis tasks **creates modularity** that allows **easy exchange of components**.

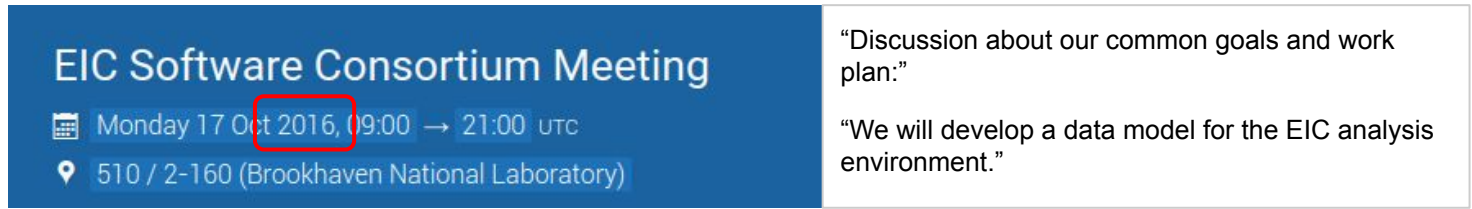
Example: Multiple clustering algorithms can be swapped out, as long as they adhere to the data model interfaces.

This modularity extends beyond the EIC, since many data structures are common across collider experiments worldwide.



Lessons Learned from EIC Software Working Group

For more than 5 years, the EIC Software Working Group has attempted to define a data model, but defining this from scratch has not resulted in a minimum viable product due to the many conflicting opinions (perfect is the enemy of the good).



EIC Software Consortium Meeting

Monday 17 Oct 2016, 09:00 → 21:00 UTC

510 / 2-160 (Brookhaven National Laboratory)

“Discussion about our common goals and work plan:”

“We will develop a data model for the EIC analysis environment.”

An **externally defined** data model based on ‘good enough practices’ with a **process for modification and extension** provides an alternative starting point.

EIC Software Statement of Principles: Open Formats

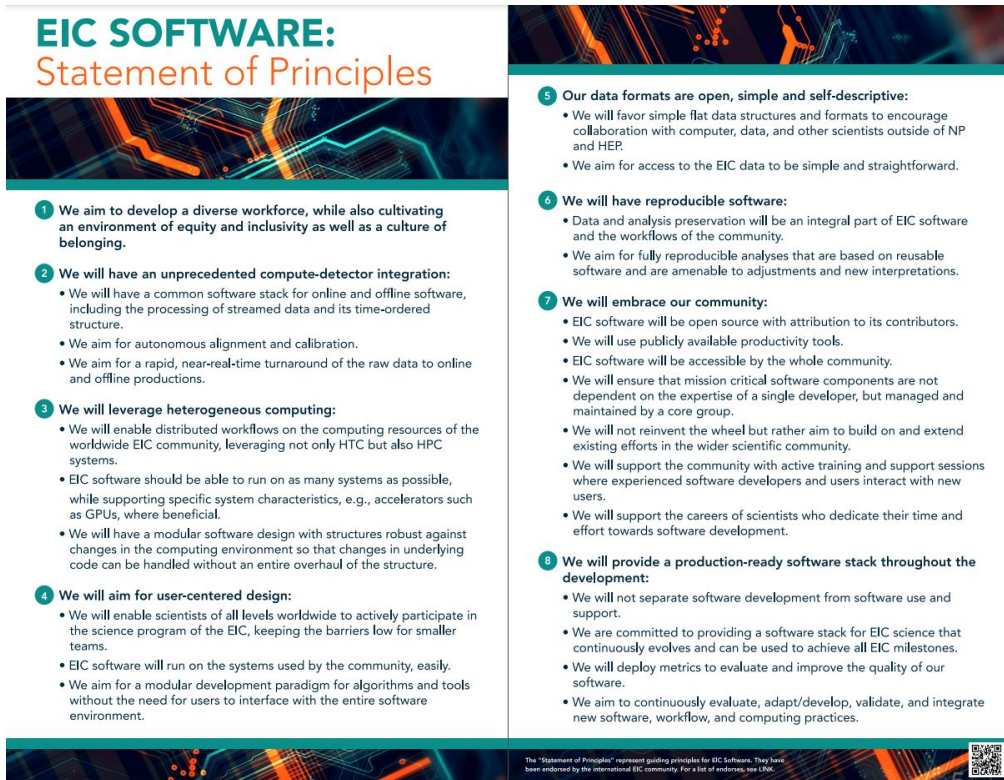
5. Our data formats are open, simple and self-descriptive.

- “We will favor simple **flat data structures** and format to encourage collaboration with computer, data, and other scientists outside nuclear and high energy physics.”

Flat (“plain-old data”, POD) vs. structured data:

- Flat: could be stored in a spreadsheet
- Structured: requires infrastructure to describe class hierarchies, etc...

Most artificial intelligence/machine learning tools are built for flat data. Heterogeneous computing works best on flat data.



EIC SOFTWARE: Statement of Principles

- 1 We aim to develop a diverse workforce, while also cultivating an environment of equity and inclusivity as well as a culture of belonging.**
- 2 We will have an unprecedented compute-detector integration:**
 - We will have a common software stack for online and offline software, including the processing of streamed data and its time-ordered structure.
 - We aim for autonomous alignment and calibration.
 - We aim for a rapid, near-real-time turnaround of the raw data to online and offline productions.
- 3 We will leverage heterogeneous computing:**
 - We will enable distributed workflows on the computing resources of the worldwide EIC community, leveraging not only HTC but also HPC systems.
 - EIC software should be able to run on as many systems as possible, while supporting specific system characteristics, e.g., accelerators such as GPUs, where beneficial.
 - We will have a modular software design with structures robust against changes in the computing environment so that changes in underlying code can be handled without an entire overhaul of the structure.
- 4 We will aim for user-centered design:**
 - We will enable scientists of all levels worldwide to actively participate in the science program of the EIC, keeping the barriers low for smaller teams.
 - EIC software will run on the systems used by the community, easily.
 - We aim for a modular development paradigm for algorithms and tools without the need for users to interface with the entire software environment.
- 5 Our data formats are open, simple and self-descriptive:**
 - We will favor simple flat data structures and formats to encourage collaboration with computer, data, and other scientists outside of NP and HEP.
 - We aim for access to the EIC data to be simple and straightforward.
- 6 We will have reproducible software:**
 - Data and analysis preservation will be an integral part of EIC software and the workflows of the community.
 - We aim for fully reproducible analyses that are based on reusable software and are amenable to adjustments and new interpretations.
- 7 We will embrace our community:**
 - EIC software will be open source with attribution to its contributors.
 - We will use publicly available productivity tools.
 - EIC software will be accessible by the whole community.
 - We will ensure that mission critical software components are not dependent on the expertise of a single developer, but managed and maintained by a core group.
 - We will not reinvent the wheel but rather aim to build on and extend existing efforts in the wider scientific community.
 - We will support the community with active training and support sessions where experienced software developers and users interact with new users.
 - We will support the careers of scientists who dedicate their time and effort towards software development.
- 8 We will provide a production-ready software stack throughout the development:**
 - We will not separate software development from software use and support.
 - We are committed to providing a software stack for EIC science that continuously evolves and can be used to achieve all EIC milestones.
 - We will deploy metrics to evaluate and improve the quality of our software.
 - We aim to continuously evaluate, adapt/develop, validate, and integrate new software, workflow, and computing practices.

The "Statement of Principles" represent guiding principles for EIC Software. They have been endorsed by the international EIC community. For a list of endorsees, see ENW.

Our Requirements for the Data Model

- A **simple method to define** the data model, accessible to anyone
- Ability to **easily modify or extend** the data model and include metadata
- Ability to use with **multiple programming languages** (e.g. C++ and Python)
- Adaptable to the changing landscape of heterogeneous computing hardware
- Does not constrain future IO implementations, file formats, or data stores
- Does not constrain memory layout: array-of-structs or struct-of-arrays
- Independent of the event processing framework for the EIC software
- Supports **streaming readout** data, e.g. frames with hits from multiple events
- Supports **truth propagation** when used in Monte Carlo simulations

Other considerations: support for schema evolution over lifetime of EIC program, allow data structure meta-programming approaches for generic algorithms

Several Solutions Evaluated, One Solution Proposed

Proposed solution consists of **two components, satisfies all requirement:**

- **podio** (<https://github.com/AIDAsoft/podio>): “plain-old-data input/output”
 - Define flat data models in a text-based format, using components and datatypes
 - Supports collections, reference collections, relations (1-to-1, 1-to-many), vectors
 - Automatic generation of C++ classes and Python bindings
 - File formats supported: ROOT and SIO files; supported by RNTuple in recent ROOT versions
 - In-memory event store implemented in podio, but also integrates in other event stores
 - Collections on disk are read-only, explicitly need to declare new collection entries as mutable
- **EDM4hep** (<https://github.com/key4hep/EDM4hep>): “event data model for HEP”
 - Data model defined using podio, based on LCIO and other past data models
 - Designed as a standard for current and future HEP experiments
 - Adopted by future HEP collider as standard event data model (FCC, CEPC)
 - Extensible with additional data types that may be site-specific
 - Integrated in DD4hep, Acts, and other adopted components of the EIC software stack

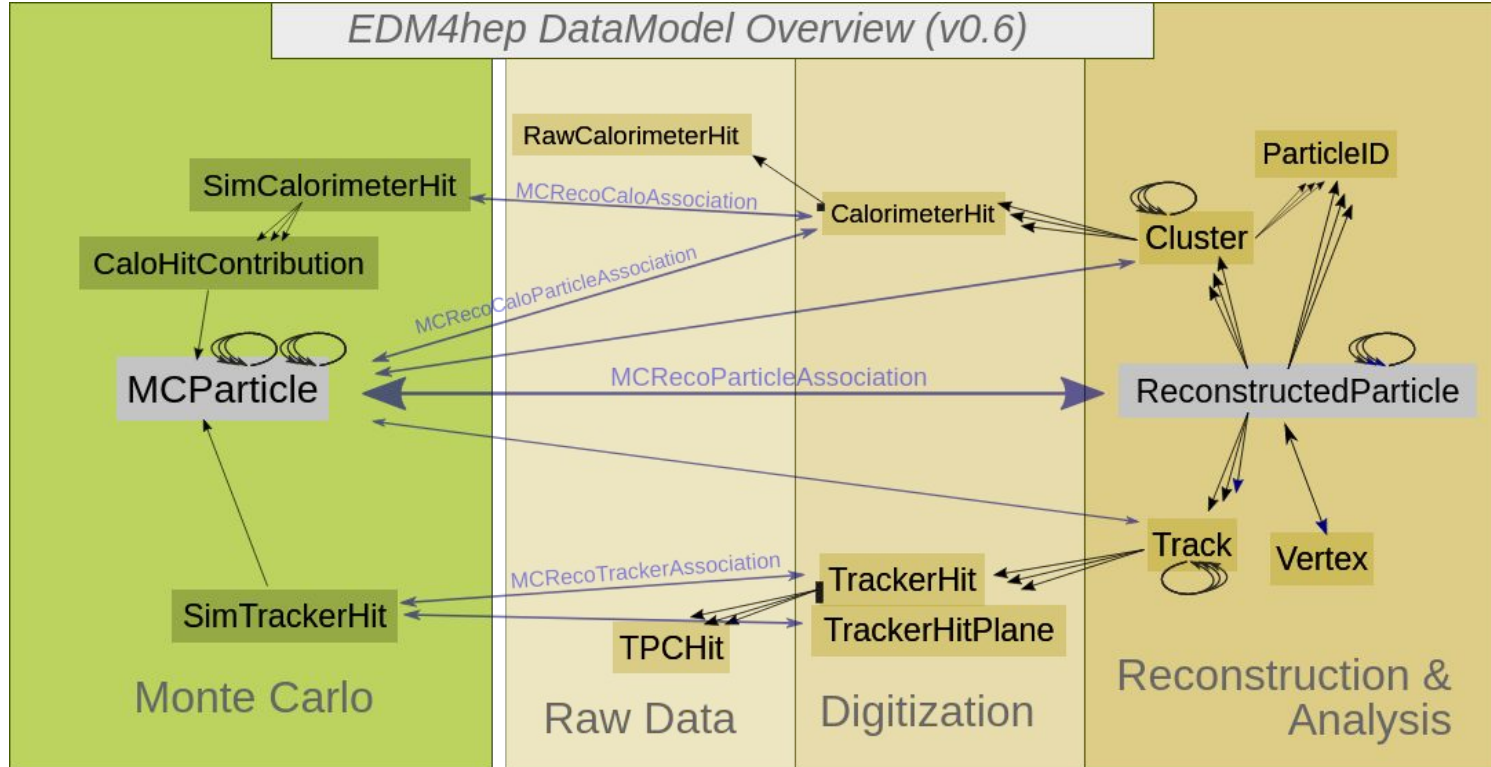
Podio: Plain-Old-Data I/O

Example: human-readable data model definition

```
edm4hep::SimTrackerHit:
  Description: "Simulated tracker hit"
  Author : "F.Gaede, DESY"
  Members:
    - uint64_t cellID      // ID of the sensor that created this hit
    - float EDep           // energy deposited in the hit [GeV].
    - float time           // proper time of the hit in the lab frame in [ns].
    - float pathLength     // path length of the particle in the sensitive material.
    - int32_t  quality     // quality bit flag.
    - edm4hep::Vector3d position // the hit position in [mm].
    - edm4hep::Vector3f momentum // the 3-momentum of the particle at the hits position in [GeV]
  OneToOneRelations:
    - edm4hep::MCParticle MCParticle // MCParticle that caused the hit.

#etc
```


EDM4hep: Event Data Model for HEP



Extensibility of EDM4hep: Adding EIC Physics

By request of the EIC community, podio supports extensions of data models.

We have been using this to define data types on top of EDM4hep since the early days of the proposal process.

```
edm4eic::InclusiveKinematics:
```

```
Description: "Kinematic variables for DIS events"
```

```
Author: "S. Joosten, W. Deconinck"
```

```
Members:
```

```
- float          x           // Bjorken x (Q2/2P.q)
- float          Q2          // Four-momentum transfer squared [GeV^2]
- float          W           // Invariant mass of final state [GeV]
- float          y           // Inelasticity (P.q/P.k)
- float          nu          // Energy transfer P.q/M [GeV]
```

```
OneToOneRelations:
```

```
- edm4hep::ReconstructedParticle e // Associated scattered electron (if identified)
```

Sustainability and Community

- All components of the data model are **open source** and **supported by multiple institutions and collaborations** with goals aligned with ours.
- **EIC collaborators are involved** in both the podio and EDM4hep projects, including through the key4HEP project as well as through code contributions.
 - Monthly EIC-oriented meetings by the key4HEP project
 - Inclusion of features requested by EIC into podio and EDM4hep (e.g. C++20 ranges support)
- **Extensions of EDM4hep** with EIC-specific data model components and data types is possible and anticipated, while standardization will benefit from inclusion into EDM4hep where appropriate.
- Worst case scenario: in the unlikely scenario that we were forced to take over sole ownership of podio/EDM4hep, we estimate ~0.25 FTE for maintenance throughout the project's duration

Summary

The existing projects podio and EDM4hep are leveraged by the EIC software stack to provide a standardized flat data model, accessible to researchers with modern AI/ML tools, on a variety of hardware and software systems.

For those aspects that are not in EDM4hep due to scope considerations, we can extend the data model with our own data type. We have experience with this from the proposal stage.

The standard data model for EIC will allow modularity and experimentation with new methodologies for data analysis.