

EIC Data Model (EDM)

A proposal for collaborative development through a standard data model

Whitney R. Armstrong

Argonne National Laboratory

October 17, 2016

1 Introduction



Overview

Goals:

- 1 Convince the developer community that a standardized data model is a great idea (short-term: this week)
- 2 Have the developer community (ESC) select and develop the specific data model (mid-term: few weeks to one month)
- 3 Begin adopting said EDM by integrating it with out tools (long term)



Goal 1

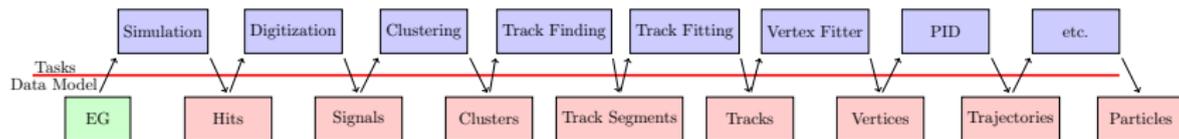
Is worth adopting a data model?

Yes!

Motivation

Development of unified geometry, detector, tracking, and reconstruction tools for an EIC.

- The data model exists at the boundaries of each software task.



Goal 1

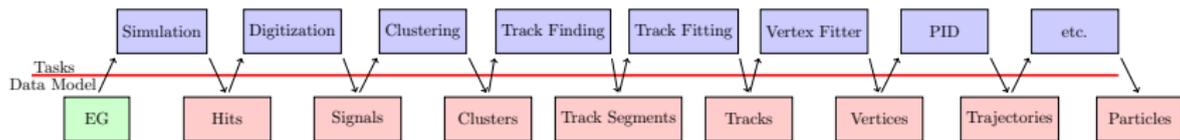
Is worth adopting a data model?

Yes!

Motivation

Development of unified geometry, detector, tracking, and reconstruction tools for an EIC.

- The data model exists at the boundaries of each software task.



Example: I want to make an Event Display

- The Data Model are the **red boxes**
- I know exactly which kind of hit/cluster/tracks and how to read them
- Now I can just worry about the actual challenge of building an elegant and useful event display.

This principle holds for all tasks manipulating (input and output) data model objects.

Hopefully I have convinced you that a standardized data model is a
good idea.
OK, but what tool?



Goal 2

What data model should we use?

What are the requirements for a good data model?

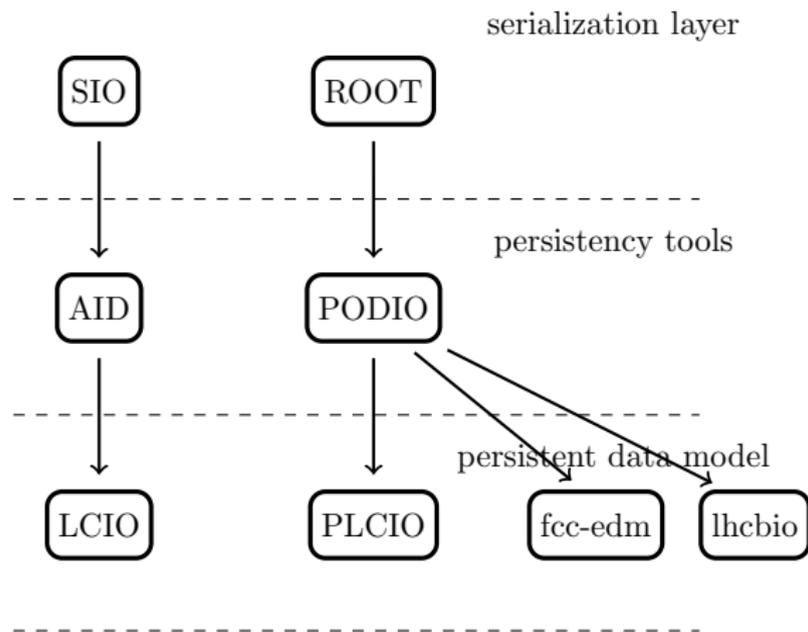
- Should be relatively static (ie, we don't want to create a new "Hit" class for every new detector/algorithm).
- ROOT compatible
- Maximum compatibility with existing libraries (frameworks?)
- What else?

Examples of data models from other projects

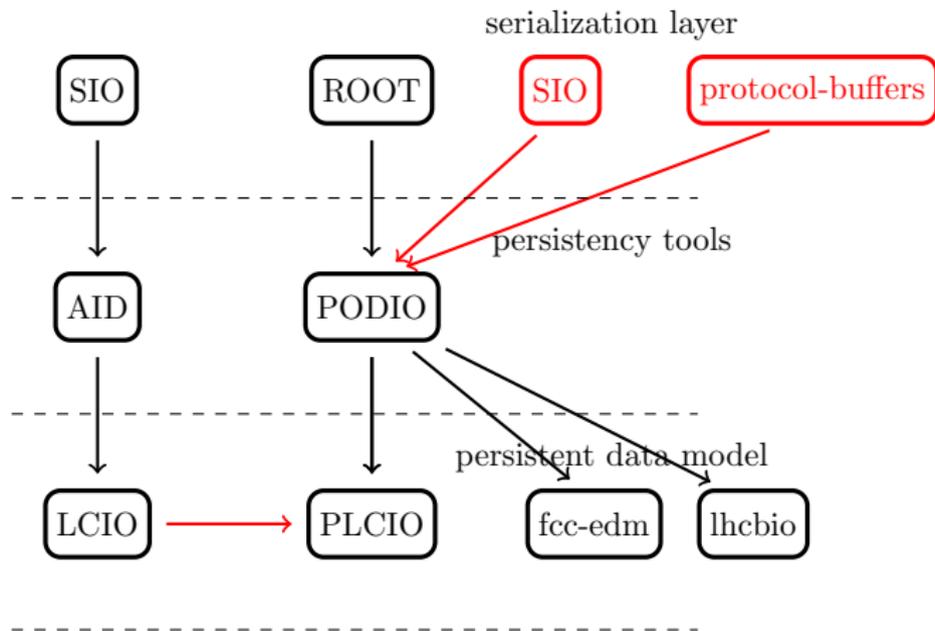
- LCIO (old)
- PLCIO
- lhcbio
- fcc-edm

What about the actual data serialization IO (file) format?

Data Persistence for the near future



Data Persistence for the near future



Goal 3: Adopting the EDM



Conclusion

- Can we do it?
- This decision has direct impact on how the tracking/geometry/detector objectives will proceed.

Some useful links

- <https://github.com/hegner/podio>
- <https://stash.desy.de/projects/IL/repos/plcio/browse>
- <https://github.com/iLCSoft>
- <https://github.com/HEP-FCC>
- <http://ilcsoft.desy.de/v01-17-09/DD4hep/v00-15/doc/html/index.html>

Backup Slides



Introduction to the data persistency problem

What is **persistent data**?

Persistent data denotes information that is infrequently accessed and not likely to be modified.

The opposite of this is **transactional data**.

– google



Introduction to the data persistency problem

What is **persistent data**?

Persistent data denotes information that is infrequently accessed and not likely to be modified.

The opposite of this is **transactional data**.

– google

What is a **persistent data structure**?

Persistent data structure is a data structure that always preserves the previous version of itself when it is modified

– wikipedia

Introduction to the data persistency problem

What is **persistent data**?

Persistent data denotes information that is infrequently accessed and not likely to be modified.

The opposite of this is **transactional data**.

– google

What is a **persistent data structure**?

Persistent data structure is a data structure that always preserves the previous version of itself when it is modified

– wikipedia

What is data **serialization**?

Serialization is the process of translating data structures or object state into a format that can be stored (for example, in a file or memory buffer, or transmitted across a network connection link) and reconstructed later in the same or another computer environment

– google

Quick note on context

Here persistent data means the data that is used **between each** step of simulation/tracking/reconstruction. Thus it facilitates the development of simple or complex single purpose libraries.

There is a larger type of data persistence, of the data archiving type, which we are not talking about there.

We want a **quasi-persistent** data model, which from the view of the entire software chain, seems to be used in a transient way.

Let's look at some HEP projects

- **SIO** (1999) - serial IO library
- **AID** (1999-2003?) - tool that generates code based on data model
- **LCIO** (2003) - A fixed but flexible persistent data model. Uses **AID** to define data structures and **SIO** for serialization.

Let's look at some HEP projects

- SIO (1999) - serial IO library
- AID (1999-2003?) - tool that generates code based on data model
- LCIO (2003) - A fixed but flexible persistent data model. Uses AID to define data structures and SIO for serialization.

LCIO is still in heavy use. Why has it successfully lasted this long?

- Is it really fast? ... Not really
- Does it have the best compression? ... No
- Are the data structures optimized for speed? ... No
- Has it been modernized with changing/improving language features? ... No

Let's look at some HEP projects

- SIO (1999) - serial IO library
- AID (1999-2003?) - tool that generates code based on data model
- LCIO (2003) - A fixed but flexible persistent data model. Uses AID to define data structures and SIO for serialization.

LCIO is still in heavy use. Why has it successfully lasted this long?

- Is it really fast? ... Not really
- Does it have the best compression? ... No
- Are the data structures optimized for speed? ... No
- Has it been modernized with changing/improving language features? ... No
- Has it been well maintained? ... **Yes.**
- Was it adopted by the community? ... **Yes.**
- Was it accessible with a variety of languages? ... **Yes.**

Looking at LCIO's Success

- LCIO was successful because it was **flexible, maintained, and adopted by the developer community** (not the users).
- User community overwhelmingly adopted ROOT for everything (but no persistent data model).
- ROOT and LCIO do not work together!
- ROOT is clearly the tool of choice and will remain so.



Looking at LCIO's Success

- LCIO was successful because it was **flexible, maintained, and adopted by the developer community** (not the users).
- User community overwhelmingly adopted ROOT for everything (but no persistent data model).
- ROOT and LCIO do not work together!
- ROOT is clearly the tool of choice and will remain so.

We need a library for the future!

It needs to

- Use ROOT IO for serialization layer.
- Develop tools for creating persistent data, making maximal use of ROOT tools as well.
- Read and write LCIO files to provide backward compatibility so we can use all the tools developed over the past 15 years.



This is the most immediate software problem

We need to solve this problem ASAP!

- Use ROOT IO for serialization layer.
- Develop tools for creating persistent data, making maximal use ROOT tools as well.
- Read and write LCIO files to provide backward compatibility so we can use all the tools developed over the past 15 years.

This is the most immediate software problem

We need to solve this problem ASAP!

- Use ROOT IO for serialization layer.
- Develop tools for creating persistent data, making maximal use ROOT tools as well.
- Read and write LCIO files to provide backward compatibility so we can use all the tools developed over the past 15 years.

Fortunately, the FCC community is already working on it:

- PODIO - Plain-old-data IO (analog of AID) but uses ROOT and treats python as first class language.
- PLCIO - LCIO data model implementation with PODIO

Both of these projects are at an early stage but can be easily completed with more support from the community.



This is the most immediate software problem

We need to solve this problem ASAP!

- Use ROOT IO for serialization layer.
- Develop tools for creating persistent data, making maximal use ROOT tools as well.
- Read and write LCIO files to provide backward compatibility so we can use all the tools developed over the past 15 years.

Fortunately, the FCC community is already working on it:

- PODIO - Plain-old-data IO (analog of AID) but uses ROOT and treats python as first class language.
- PLCIO - LCIO data model implementation with PODIO

Both of these projects are at an early stage but can be easily completed with more support from the community.

The real challenge...

Getting library/toolkit/framework developers to agree to using the same event data model.