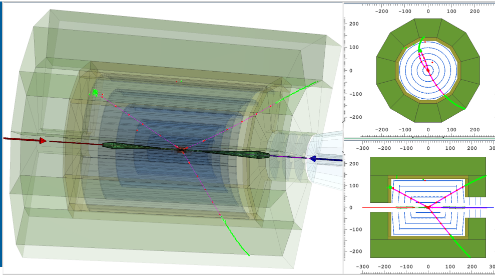
Simulation and Reconstruction Software

Tools for the next decade



Whitney R. Armstrong
Argonne National Laboratory

March 31, 2020

Argonne NATIONAL LABORATORY

Introduction

Trends in software:

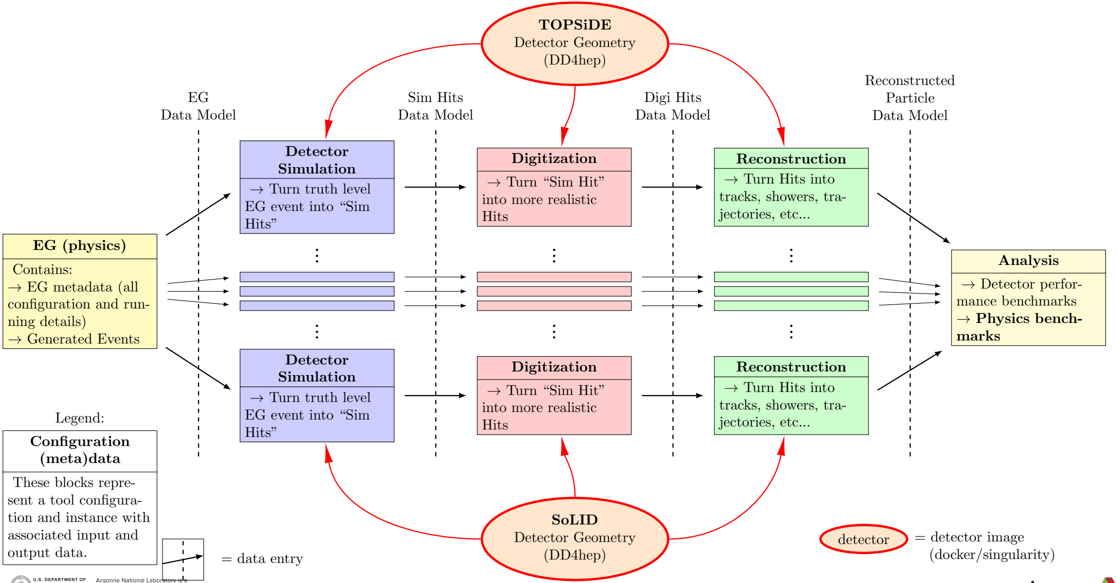
- Tools mostly written in C++ and python.
- python provides a nice configuration/scripting
- Thread-safe code – C++ `const` correctness[1]
- Data model for algorithm interfacing

Toolkit

- **DD4hep** – Detector description
- **A Common Tracking Software (Acts)**
- **PODIO** – Data model tool
- Genfit – track fitting

[1] const changed meaning with c++11 Herb Sutter talk

Data-flow Map



TOPSiDE
Detector Geometry
(DD4hep)

EG
Data Model

Sim Hits
Data Model

Digi Hits
Data Model

Reconstructed
Particle
Data Model

EG (physics)

Contains:
→ EG metadata (all configuration and running details)
→ Generated Events

Detector Simulation

→ Turn truth level EG event into “Sim Hits”

Digitization

→ Turn “Sim Hit” into more realistic Hits

Reconstruction

→ Turn Hits into tracks, showers, trajectories, etc...

Analysis

→ Detector performance benchmarks
→ **Physics benchmarks**

SoLID
Detector Geometry
(DD4hep)

Legend:

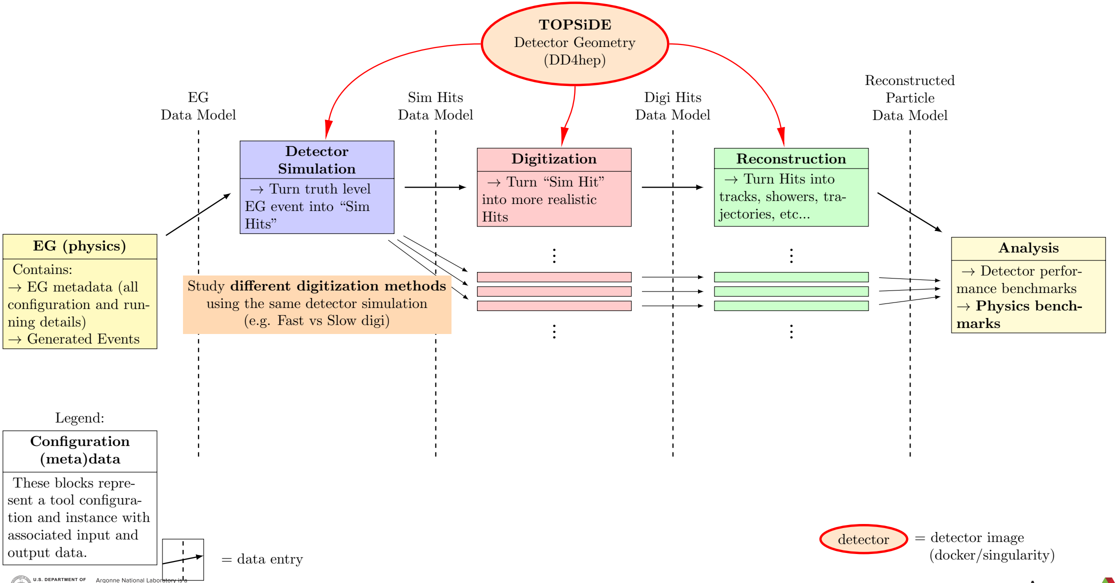
Configuration (meta)data

These blocks represent a tool configuration and instance with associated input and output data.

= data entry

detector = detector image (docker/singularity)

Data-flow Map



TOPSiDE
Detector Geometry
(DD4hep)

EG Data Model

Sim Hits Data Model

Digi Hits Data Model

Reconstructed Particle Data Model

EG (physics)

Contains:
→ EG metadata (all configuration and running details)
→ Generated Events

Detector Simulation

→ Turn truth level EG event into "Sim Hits"

Digitization

→ Turn "Sim Hit" into more realistic Hits

Reconstruction

→ Turn Hits into tracks, showers, trajectories, etc...

Analysis

→ Detector performance benchmarks
→ **Physics benchmarks**

Study **different digitization methods** using the same detector simulation (e.g. Fast vs Slow digi)

Legend:

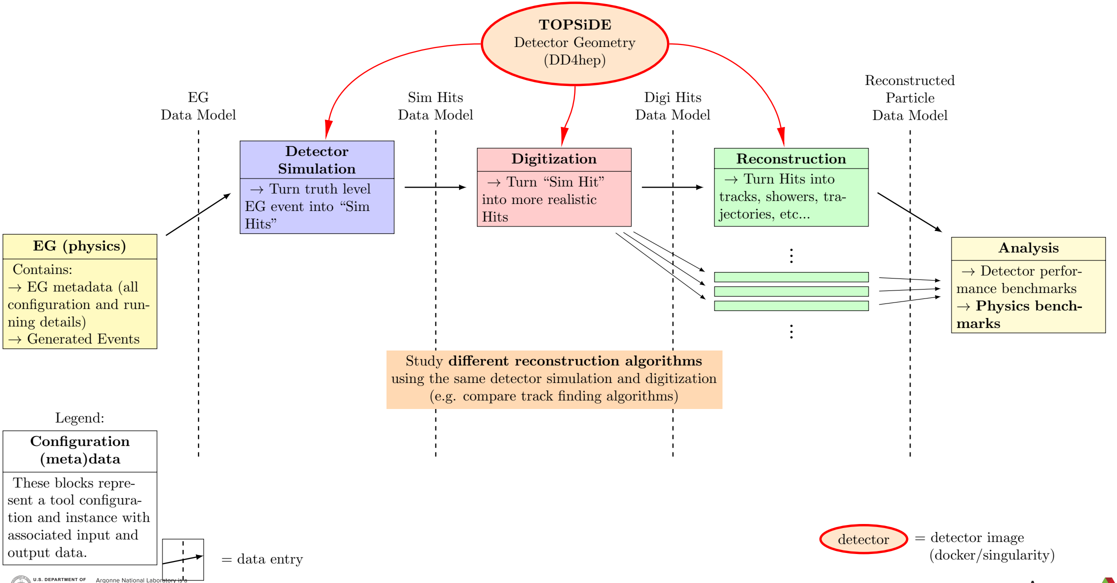
Configuration (meta)data

These blocks represent a tool configuration and instance with associated input and output data.

= data entry

detector = detector image (docker/singularity)

Data-flow Map



TOPSiDE
Detector Geometry
(DD4hep)

EG
Data Model

Sim Hits
Data Model

Digi Hits
Data Model

Reconstructed
Particle
Data Model

EG (physics)

Contains:
→ EG metadata (all configuration and running details)
→ Generated Events

Detector Simulation

→ Turn truth level EG event into "Sim Hits"

Digitization

→ Turn "Sim Hit" into more realistic Hits

Reconstruction

→ Turn Hits into tracks, showers, trajectories, etc...

Analysis

→ Detector performance benchmarks
→ **Physics benchmarks**

Study **different reconstruction algorithms** using the same detector simulation and digitization (e.g. compare track finding algorithms)

Legend:

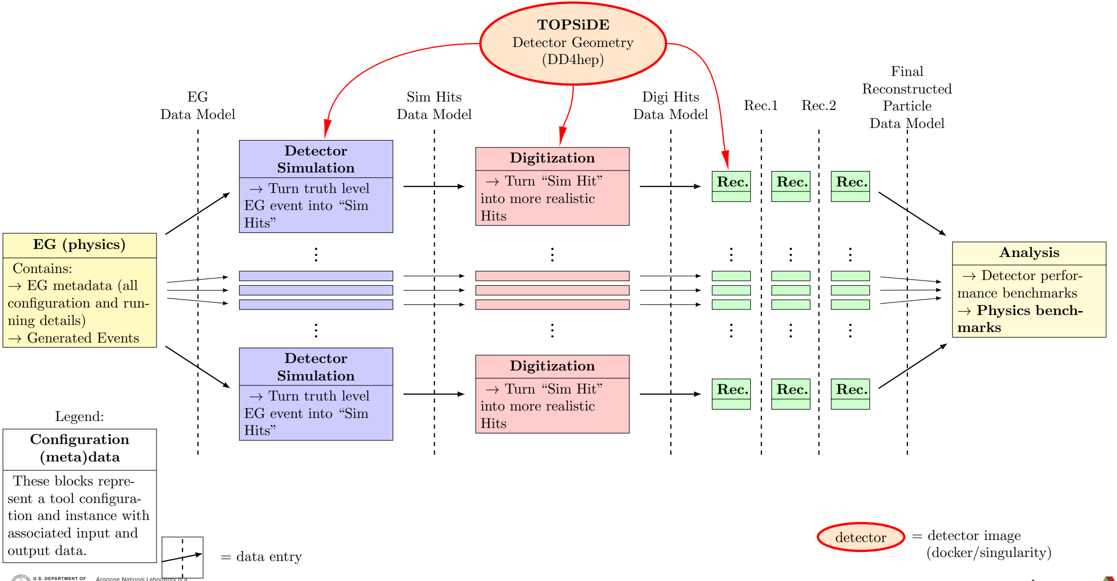
Configuration (meta)data

These blocks represent a tool configuration and instance with associated input and output data.

= data entry

detector = detector image (docker/singularity)

Data-flow Map



TOPSiDE
Detector Geometry
(DD4hep)

EG Data Model

Sim Hits Data Model

Digi Hits Data Model

Rec.1

Rec.2

Final Reconstructed Particle Data Model

EG (physics)

Contains:
→ EG metadata (all configuration and running details)
→ Generated Events

Detector Simulation

→ Turn truth level EG event into "Sim Hits"

Digitization

→ Turn "Sim Hit" into more realistic Hits

Rec.

Analysis

→ Detector performance benchmarks
→ **Physics benchmarks**

Legend:

Configuration (meta)data

These blocks represent a tool configuration and instance with associated input and output data.

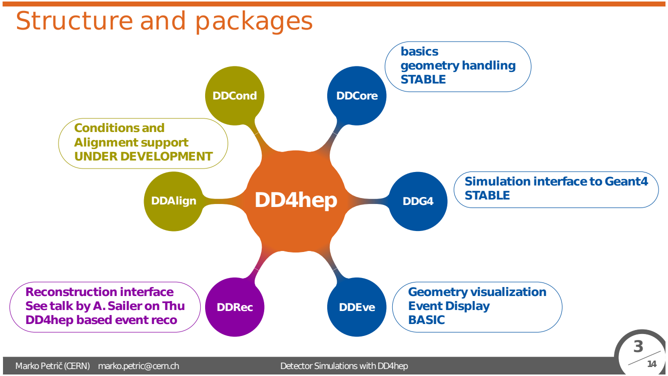
= data entry

detector = detector image (docker/singularity)

DD4hep

Detector Description

The result of a study from the *Advanced European Infrastructures for Detectors at Accelerators* (EU AIDA 2020) initiative.



Structure and packages

DD4hep

DDCond

Conditions and
Alignment support
UNDER DEVELOPMENT

DDCore

basics
geometry handling
STABLE

DDAlign

DDG4

Simulation interface to Geant4
STABLE

DDRec

Reconstruction interface
See talk by A. Sailer on Thu
DD4hep based event reco

DDEve

Geometry visualization
Event Display
BASIC

Marko Petrič (CERN) marko.petric@cern.ch Detector Simulations with DD4hep 3 / 14

- Thoughtfully designed for future (thread-safe)
- Interface provides full access to Geant4
- **Single source of geometry**
- Simple geometry hook → **better algorithm development**
- Full geometry definition defined in human readable compact detector description file
- Easily used in a ROOT/python scripts and works well with external tools.

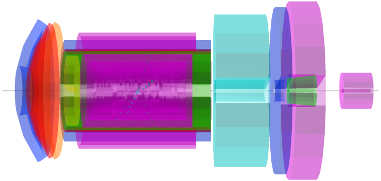
DD4hep solves the ”geometry problem” for end-to-end simulation and reconstruction.

Nuclear Physics Detector Library (NPDet)

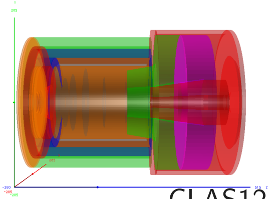
NPDet is a **collection of parameterized detectors** (using DD4hep) which can be used to construct full concept detectors in a single text file.

NPdet/src/
- GenericDetectors
- SiEIC
- JLEIC
- CLAS12
- SOLID

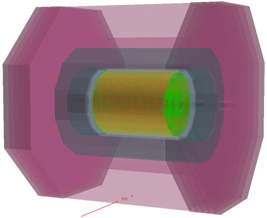
JLEIC



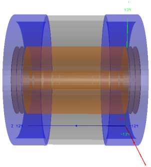
SOLID



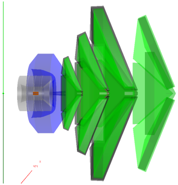
SiEIC



eRHIC



CLAS12



DD4hep Geometry Hooks

C++ in a ROOT script

```
dd4hep::Detector& detector = dd4hep::Detector::getInstance();  // Get the DD4hep instance
detector.fromCompact("my_awesome_detector.xml");               // Load the compact XML file
dd4hep::rec::CellIDPositionConverter converter(detector);      // Position/cellid converter tool
[...]
    for(const auto& h: hits) {
      auto cell     = h->cellID;                        // Unique segment/volume identifier
      auto pos1     = converter.position(cell);         // The segmentation hit postion
      auto cell_dim = converter.celldimensions(cell);   // Dimensions of segment/volume
      [...]
      }
[...]
auto    bField = detector.field().magneticField(pos);   // Get the magnetic field
double  Bz     = bField.z()/dd4hep::tesla;
```

That's it.

See NPDet examples for a tutorial (work in progress).

Add a new detector

Detector construction (.cpp)

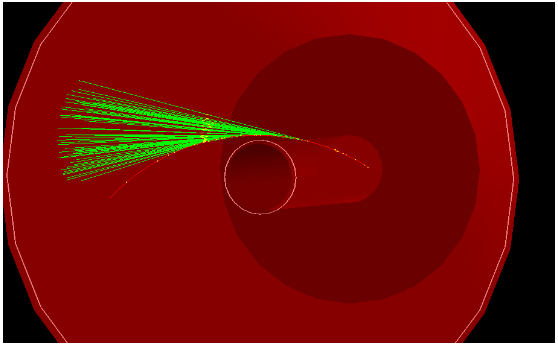
```cpp
static Ref_t build_detector(Detector& det, xml_h e, SensitiveDetector sens)
{
  xml_det_t   x_det     = e;
  Material    air       = det.air();
  double z_offset = dd4hep::getAttrOrDefault(x_det, _Unicode(zoffset), 10.0*dd4hep::cm);
... [ Build geometry ]
}
DECLARE_DETELEMENT(SimpleRomanPot, build_detector)
```

Compact detector description (.xml)

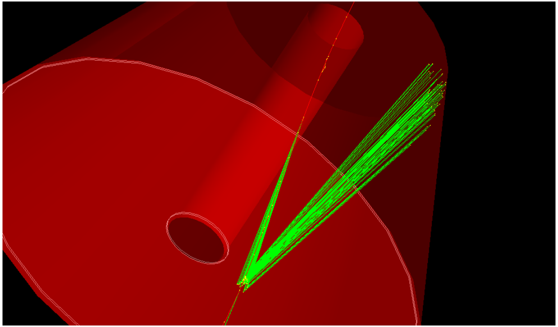
```xml
<detector id="1" name="MyRomanPot" type="SimpleRomanPot"
          vis="RedVis" readout="RomanPotHits" zoffset="1.0*m">
</detector>
[...]
<readouts>
  <readout name="RomanPotHits">
    <segmentation type="CartesianGridXY" grid_size_x="1.0*mm" grid_size_y="1.0*cm" />
    <id>system:5,layer:9,module:14,x:32:-16,y:-16</id>
  </readout>
</readouts>
```

Recent improvements from DD4hep developers

Optical photon process:



optical surfaces:



All material properties defined in compact detector description file. [a]

[a] Requires root > 6.18

Sensitive Detectors and Data Model

Built-in SD types

- calorimeter
- tracker
- *maybe a photon detector in the future...*

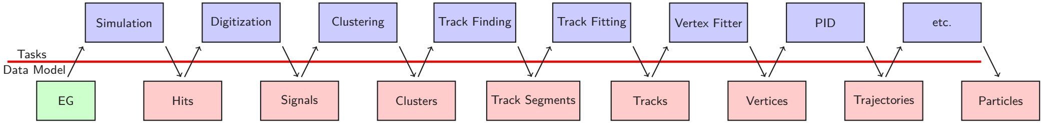
Uses built-in data model
`dd4hep::Geant4Tracker::Hit`
`dd4hep::Geant4Calorimeter::Hit`

Custom SD implementation

Full access to Geant4 information possible through implementation of `G4VSenstiveDetector`.

Can use built-in data model or define your own

A good Event Data Model (EDM) defines task boundaries and decouples algorithms/frameworks.



Tasks
Data Model

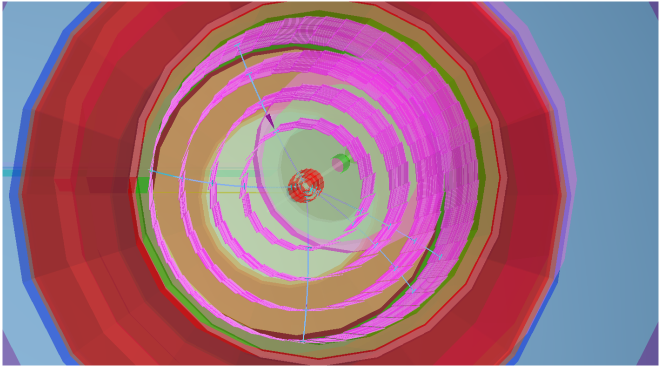
Simulation
Digitization
Clustering
Track Finding
Track Fitting
Vertex Fitter
PID
etc.

EG
Hits
Signals
Clusters
Track Segments
Tracks
Vertices
Trajectories
Particles

Track Reconstruction

- Track Finding – Hough, Conformal finding algos, Hopfield network, etc...
- Track Fitting – Kalman Filters, DAFs, GBL, etc...
- Line between finding/fitting not always clear.
- Acts aims to provide a performant, future proof toolkit for tracking
- **Acts is thread-safe**
- Acts is being **actively developed** by a team at Cern

DD4hep and Genfit



Both use TGeo – all material accounted for with no effort. Full Kalman Filter track fitting in a root script.

Event Processing Framework

Short vs. Long term

Short Term:

- Not critically important in the near term
- Better to not make a decision than to make the wrong one
- Currently a few good options, but none really stands out as solving a problem.

Long Term:

- An event processing framework will be important for handling increasingly complex processing chains
- Will be difficult to change after the fact

My Opinion

When picking this framework, it is best to wait until it is absolutely needed to solve a problem.

Pick an Event Data Model instead

The Data model[a] is more important now: pick an existing model (eg LCIO/FCC) and extend as needed

[a] EDM is not picking a language implementation, serialization, or IO library

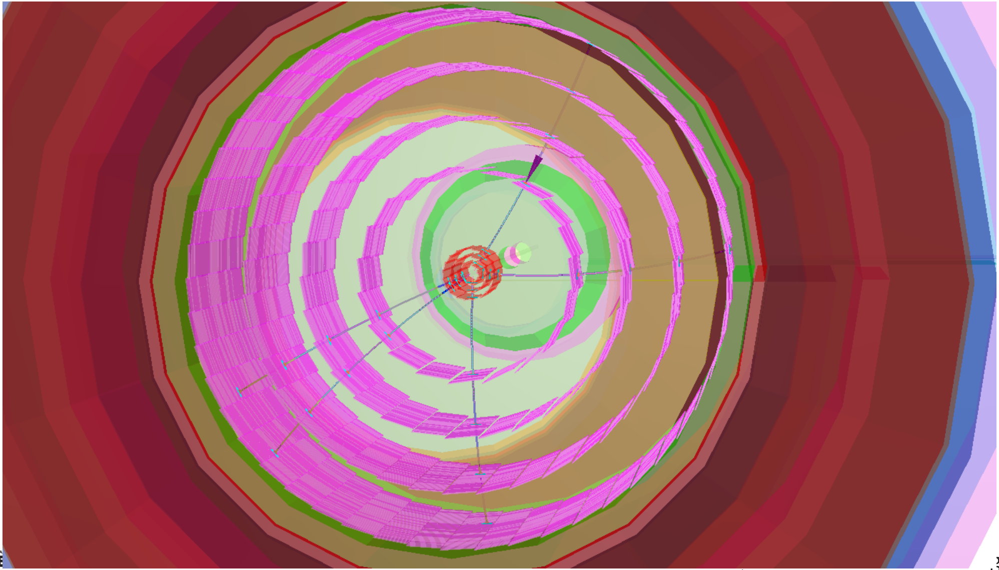
Summary

- There are a lot of good simulation and reconstruction tools that did not exist 5 years ago.
- DD4hep uniquely solves the ”geometry problem”
- Many tools play nice with DD4hep.
- Acts will provide an excellent platform for tracking and reconstruction algorithm and library development.
- Event Data Model consistency is more important than Event Processing Frameworks.

backup

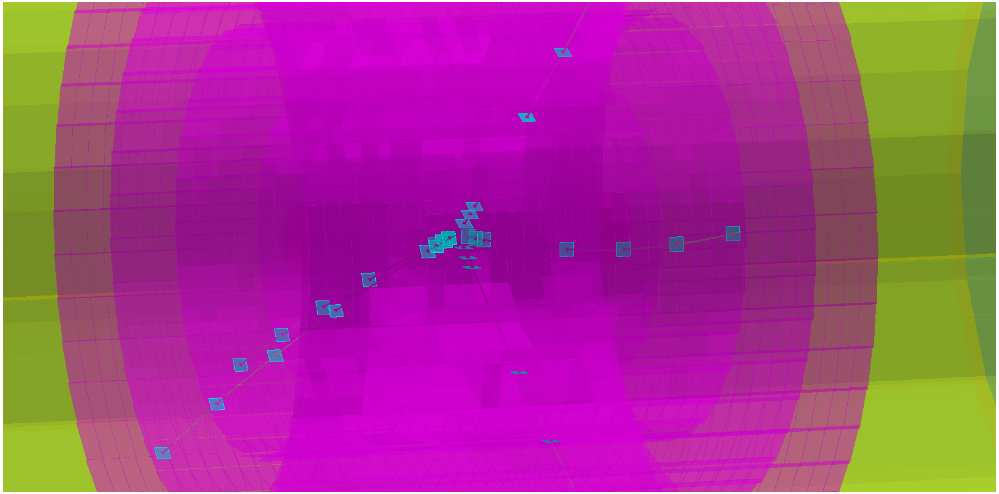
DD4hep and Genfit

Reconstructed Tracks

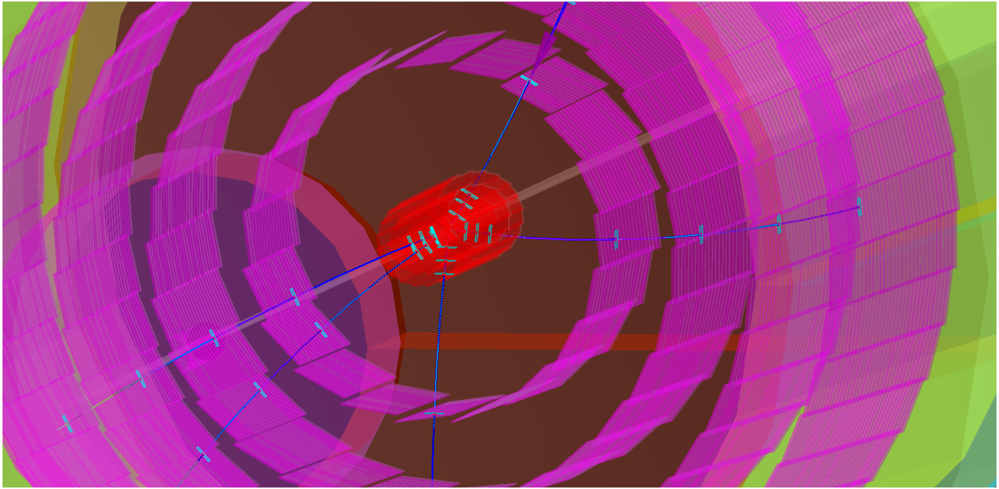


DD4hep and Genfit

Reconstructed Tracks

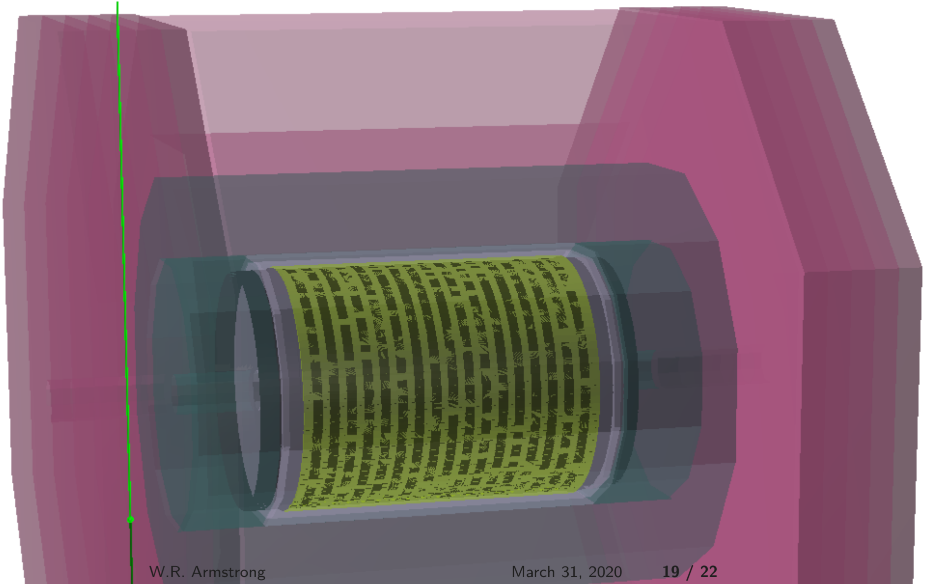


Reconstructed Tracks



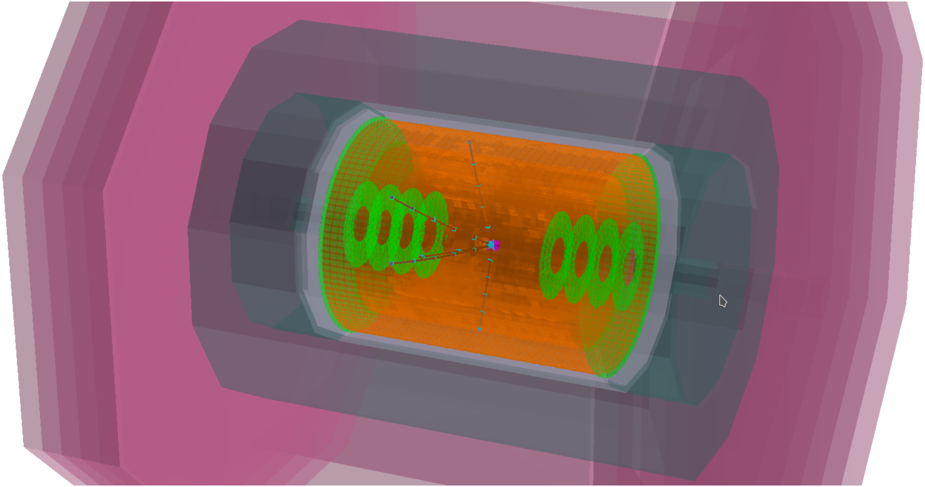
SiEIC

SiD style detector



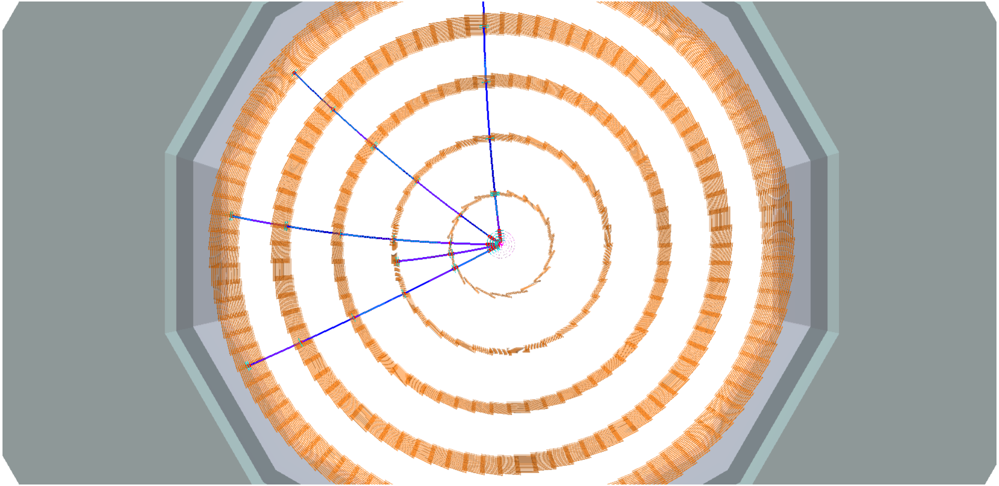
SiEIC

Reconstructed Tracks

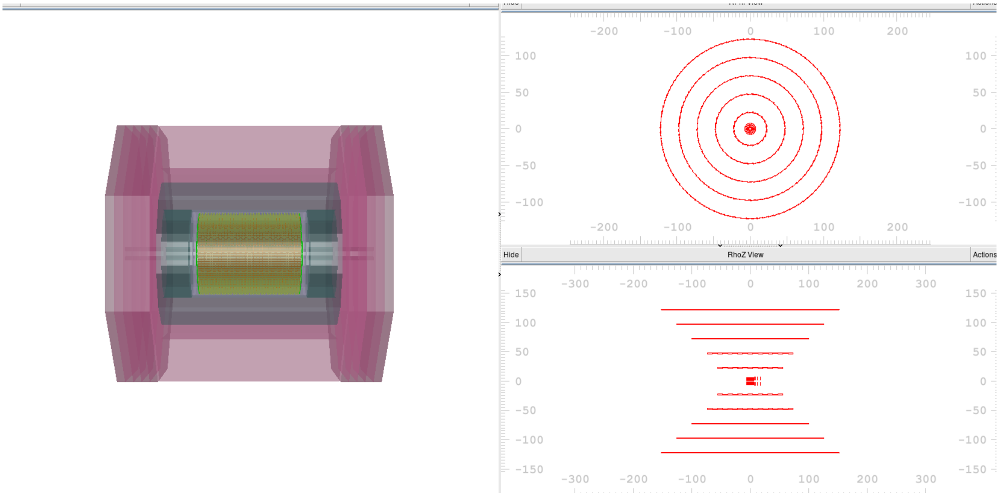


SiEIC

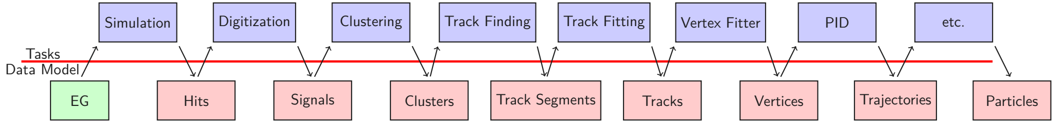
Reconstructed Tracks



Reconstructed Tracks



Why a Data Model?



Simulation

Digitization

Clustering

Track Finding

Track Fitting

Vertex Fitter

PID

etc.

Tasks

Data Model

EG

Hits

Signals

Clusters

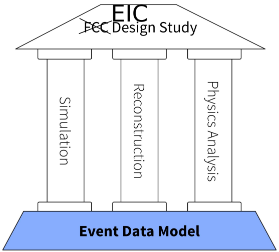
Track Segments

Tracks

Vertices

Trajectories

Particles



EIC ~~FCC~~ Design Study

Simulation

Reconstruction

Physics Analysis

Event Data Model

The FCC software: how to keep SW experiment independent - A. Zaborowska

- The **Data Model** is the boundaries of every task.
- A **Common** data model is the first step towards generic algorithms and tasks
- Challenge: Getting everyone to agree
- Initial data model: LCIO (not the library)
- Note: *Data Model* does not mean *serialization tool*! It is just the data structures
- podio is a new tool which by default uses ROOT for serialization (new serialization libraries can be easily added)