Future of the EIC Argonne Software Toolkit

Whitney Armstrong, David Blyth, Sergei Chekanov, Ian Cloët, Adam Freese, Sereres Johnston, Mohammad Hattawy, José Repond and the Argonne EIC Collaboration

Argonne National Laboratory
Funded by ANL LDRD

October 16, 2017
Outline

1. Quick Overview
   - Guiding Philosophy

2. Argonne Software Toolkit
   - Critical Software Tools

3. Simulation and Reconstruction Data-Flows

4. Future Vision

5. Summary
Full Simulation and Reconstruction Tasks

Basic tasks:

1. **Event Generation** - Produce the simulation input events
2. **Detector Simulation** - Particle transport through detectors (Geant4)
3. **Digitization** - Turn *Sim Hits* into realistic hits
4. **Reconstruction** - Track, vertex, PID, PFA, and primary reconstruction
5. **Performance Analysis** - Collection of benchmark analyses used to tune the overall design
Question: What is the best software framework?

frame·work
/fræmˌwɜːrk/ (4)
noun
- an essential supporting structure of a building, vehicle, or object.
  "a conservatory in a delicate framework of iron"
synonyms: frame, substructure, infrastructure, structure, skeleton, chassis, shell, body, bodywork. More
- a basic structure underlying a system, concept, or text.
  "the theoretical framework of political sociology"
synonyms: structure, shape, fabric, order, scheme, system, organization, construction, configuration, composition, warp and woof, informal makeup
  "the framework of society"
Question: What is the best software framework?

frame·work  
/frəmˈwɜrk/  (dʒ)  
noun  
an essential supporting structure of a building, vehicle, or object.  
"a conservatory in a delicate framework of iron"  
synonyms: frame, substructure, infrastructure, structure, skeleton, chassis, shell, body, bodywork. More  
• a basic structure underlying a system, concept, or text.  
"the theoretical framework of political sociology"  
synonyms: structure, shape, fabric, order, scheme, system, organization, construction, configuration, composition, warp and woof, informal makeup  
"the framework of society"

Trick question. The best framework is having no framework!

The Argonne software toolkit follows the idea that large frameworks are bad and should be avoided.
No Frameworks
Classic case of “less is more”

Why is this better?

- Fewer dependencies (always good)
- More freedom of choice
- Nested frameworks result in more rigid larger frameworks
- Precludes certain ideas/uses (often addressed by refactoring)

Frameworks are unavoidable. Examples:
- operating system + compiler
- language
- ROOT, GEANT4

Many frameworks we implicitly accept or take for granted, e.g., x86_64
Some software tools: (in no particular order)

- HepSim
- GEANT4
- ROOT
- DD4hep (S.Chekanov’s talk)
- LCIO, ProMC, proio, podio, fcc-edm, ...
- SLHC + lesim → evochain (D.Blyth’s talk) → This talk
  - NPdet + a collection of tools
- parameic : light weight parameter passing tool (under development)
Outline

1 Quick Overview
   - Guiding Philosophy

2 Argonne Software Toolkit
   - Critical Software Tools

3 Simulation and Reconstruction Data-Flows

4 Future Vision

5 Summary
DD4hep

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

Structure and packages

- Thoughtfully designed
- Interface to Geant4
- Single source of geometry
- Simple geometry hook → better algorithm development
- Full concept detector described in human readable text file
- Easily used in a root/python scripts
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.
Add a new detector

```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)
```

```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>
```
GEANT + DD4hep
The largest framework in the toolkit

DD4hep provides a **single geometry source** used in both simulation and reconstruction

Geometry hooks allow for development of **flexible and unified** reconstruction
- Allows easier development of algorithms
- Generic algorithms become the focus of development
- **No large framework to battle** and integrate with
Outline

- Quick Overview
  - Guiding Philosophy

- Argonne Software Toolkit
  - Critical Software Tools

3 Simulation and Reconstruction Data-Flows

- Future Vision

- Summary
Data-flow Map

**EG (physics)**
- Contains:
  - EG metadata (all configuration and running details)
  - Generated Events

**Legend:**
- Configuration (meta)data:
  These blocks represent a tool configuration and instance with associated input and output data.

**Configuration (meta)data**
- = HepSim data entry

**TOPSiDE**
- Detector Geometry (DD4hep)
- Contains:
  - Generated Events Metadata

**Digitization**
- Turn “Sim Hit” into more realistic hits

**Reconstruction**
- Turn Hits into tracks, showers, trajectories, etc...

**Analysis**
- Detector performance benchmarks
- Physics benchmarks

**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...

**JLEIC**
- Detector Geometry (DD4hep)

**TOPSiDE**
- Detector Geometry (DD4hep)

**JLEIC**
- Detector Geometry (DD4hep)

**Legend:**
- detector = HepSim detector image (docker/singularity)
Data-flow Map

EG Data Model

- Detector Simulation
  - Turn truth level EG event into “Sim Hits”

Sim Hits Data Model

- Digitization
  - Turn “Sim Hit” into more realistic Hits

Digi Hits Data Model

- Reconstruction
  - Turn Hits into tracks, showers, trajectories, etc...

Reconstructed Particle Data Model

Analysis

- Detector performance benchmarks
- Physics benchmarks

Legend:

Configuration (meta)data

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

= HepSim data entry

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

EG (physics)

Contains:

- EG metadata (all configuration and running details)
- Generated Events

EOS

TOPSiDE Detector Geometry (DD4hep)

...
Data-flow Map

**EG** (physics)

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

**TOPSiDE**
Detector Geometry (DD4hep)

**EG Data Model**

**Sim Hits Data Model**

**Digi Hits Data Model**

**Reconstructed Particle Data Model**

**Legend:**
- Configuration (meta)data

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

- = HepSim data entry

**Legend:**
- detector = HepSim detector image (docker/singularity)

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

- Turn truth level EG event into "Sim Hits"
- Turn "Sim Hit" into more realistic Hits
- Turn Hits into tracks, showers, trajectories, etc...
- Detector performance benchmarks
- Physics benchmarks

October 16, 2017
DD4hep Geometry Hooks
C++ in a ROOT script

```cpp
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 position
    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).
Where are we going with this?

Use geometry hooks to **develop generic** digitization and reconstruction **algorithms**

Not detector concept specific...

```
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 pos = converter.position(cell); // The segmentation hit position
    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;
```

- Digitization Algorithms
- Tracking Finding Algorithms
- Track Fitting Algorithms
- Algorithms, Algorithms, Algorithms

**Focus on the algorithm development**

- The product of effort is high quality algorithm (not a bigger framework)
- Many existing algorithms are embedded in tightly coupled frameworks

Can easily collaborate and get contributions from other R&D Consortia!
Outline

1 Quick Overview
   ■ Guiding Philosophy

2 Argonne Software Toolkit
   ■ Critical Software Tools

3 Simulation and Reconstruction Data-Flows

4 Future Vision

5 Summary
Tracking Example

Developing good algorithms is the goal!

A note about recent ROOT developments

TDataFrame is awesome! Check it out.

```cpp
[](const std::vector<lcio::TrackerHitData>& hits) {
    HoughTransform ht(captured_params);
    std::vector<std::vector<lcio::TrackerHitData>> res = ht(vec_hits);
    return res;
};

[](const std::vector<std::vector<lcio::TrackerHitData>>& possible_tracks) {
    std::vector<lcio::TrackData> result_tracks;
    for(const auto& track_seed : possible_tracks) {
        [GenFit...]
    }
    return result_tracks;
};
```

Frameworks come and go, algorithms are forever...
Moving Forward
Extracting algorithms from existing frameworks

- We want to build a collection of **generic algorithms**
- Currently there are many excellent algorithms embedded in tightly coupled frameworks will extracted
- These can be made more general with DD4hep hooks
- Individual detector experts are most familiar their operation → best people to characterize its digitization
Outline

1 Quick Overview
   - Guiding Philosophy

2 Argonne Software Toolkit
   - Critical Software Tools

3 Simulation and Reconstruction Data-Flows

4 Future Vision

5 Summary
Links and References

- **HepSim**
- **EICweb** ([eicweb.phy.anl.gov](http://eicweb.phy.anl.gov)) - EIC dedicated gitlab server (publicly available to EIC UG)
- **Singularity**
- **DD4hep**
- **lcgeo**

DD4hep Presentations

- Detector Simulations with DD4hep - Marko Petric
- DD4hep Based Event Reconstruction - Andre Sailer
- The FCC software: how to keep SW experiment independent - A. Zaborowska
Summary
EIC Argonne Software Toolchain (EAST)

• We are shedding tightly coupled frameworks for a flexible toolkit
• Focusing on algorithm development – not framework development

• **Collaboration tools** for the **EIC User Group** are available now.
• **We want to invite the entire EIC User Group to collaborate.**
  • Contribute new EG data (physics) – Let’s see what detectors work best
  • Add detectors to NPDet detector library – Make your detector technology available
  • Add reconstruction data for a new concept detector
  • Write benchmarks (detector and physics) – Optimise your concept detector to physics
  • Suggest ideas for improvement! – **We want EIC UG feedback**
Backup Slides
JLEIC
Reconstructed Tracks
Reconstructed Tracks
Reconstructed Tracks
SiEIC
SiD style detector
SiEIC
Reconstructed Tracks
SiEIC

Reconstructed Tracks
Reconstructed Tracks
Why a 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
- EAST initial data model: LCIO
- 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)