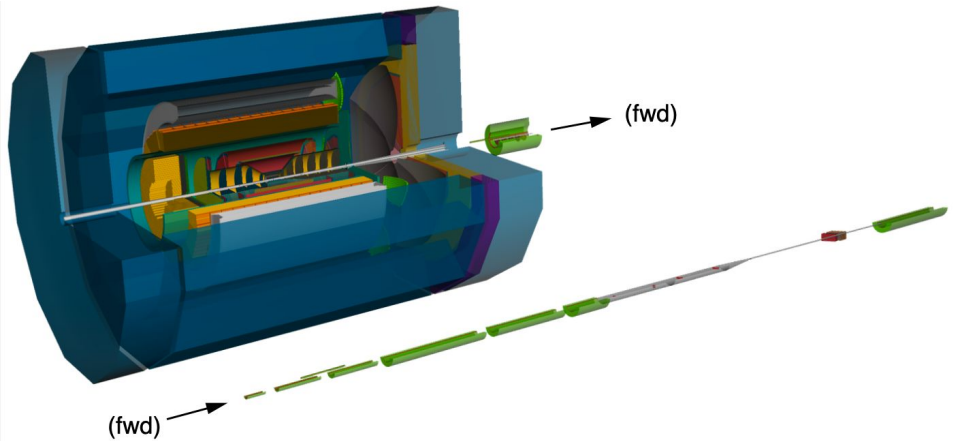
Experience using DD4hep for EIC Detector Design



Sylvester Joosten
CompSW Meeting, June 8 2022

Why did we pick DD4hep for the ATHENA proposal?

Philosophy: Let's prepare for our future at the EIC!

ATHENA

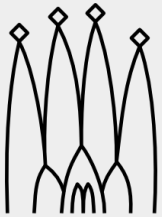
- **Build forward-looking team of developers to ensure the long-term success of the EIC scientific program in software & computing.**
- Focus on modern scientific computing practices
  - Strong emphasis on modular orthogonal tools.
  - Integration with HTC/HPC, CI workflows, and enable use of data-science toolkits.
- Avoid "not-invented-here" syndrome, and instead leverage cutting-edge CERN-supported software components where possible.
  - Build on top of mature, well-supported, and actively developed software stack.
  - Externalize support burden where possible.
- Actively work with the EICUG SWG to help develop and integrate community tools for all collaborations.

1. It fit well within our philosophy
2. We knew from experience it would be feasible to train a large group of people to become productive quickly.

How did DD4hep fit into the software stack?

DD4hep (CLiC, SLD): define active and passive geometry, readout segmentation

Simulation in npsim (DDG4) provides hits for Gaudi



Gaudi (originally LHCb): connect geometry and data to algorithms

Flat output files (podio) enable flexible workflows (ROOT, python,…) without additional dependencies, available worldwide (S3, xrootd)

DD4hep geometry service in Juggler

ACTS has built-in support for DD4hep geometries

G4 backend (can be swapped out, e.g., celeritas, opticks,…)

Juggler: I/O, digitization, reconstruction, clustering and tracking algorithms

Call ACTS for tracking & vertexing, providing hits and detector geometries

ACTS (ATLAS): provide general track reconstruction algorithms

Experiences with DD4hep

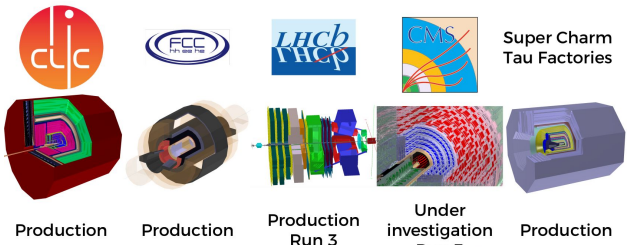
Parametrized geometries

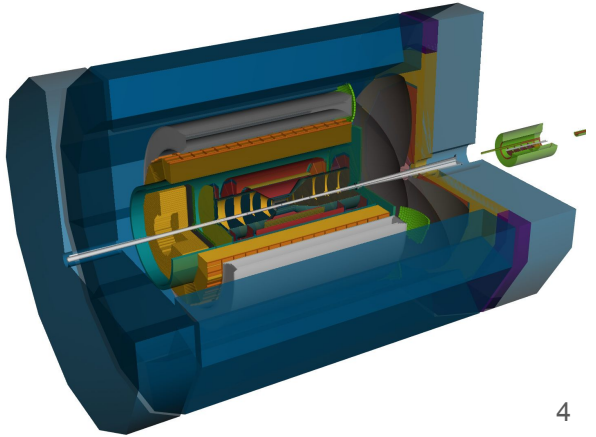
- Geometry configuration happens through XML files: easy to edit for beginners!
- Learning curve to design a proper detector parametrization a bit steeper, but easy to overcome under expert guidance.
- Worth it:
  - Well designed parametrization easy to maintain.
  - Parametrized geometries show their power when designing and optimizing a detector: were able to manage four major design iterations for ATHENA (*Acadia, BigBend, Canyonlands, DeathValley*), and many smaller optimizations.
- jsROOT geometry browser invaluable for both beginners and experts.

Development with DD4hep

- Very positive interactions with DD4hep developers:
  - Joined some of their regular meetings
  - ATHENA had multiple pull requests merged into main DD4hep GitHub repository, very fast turnaround!
- Long-term pathway for custom DD4hep plugins: can minimize amount of custom code we need to maintain ourselves by pushing our developments upstream to DD4hep and share with the entire particle physics community.

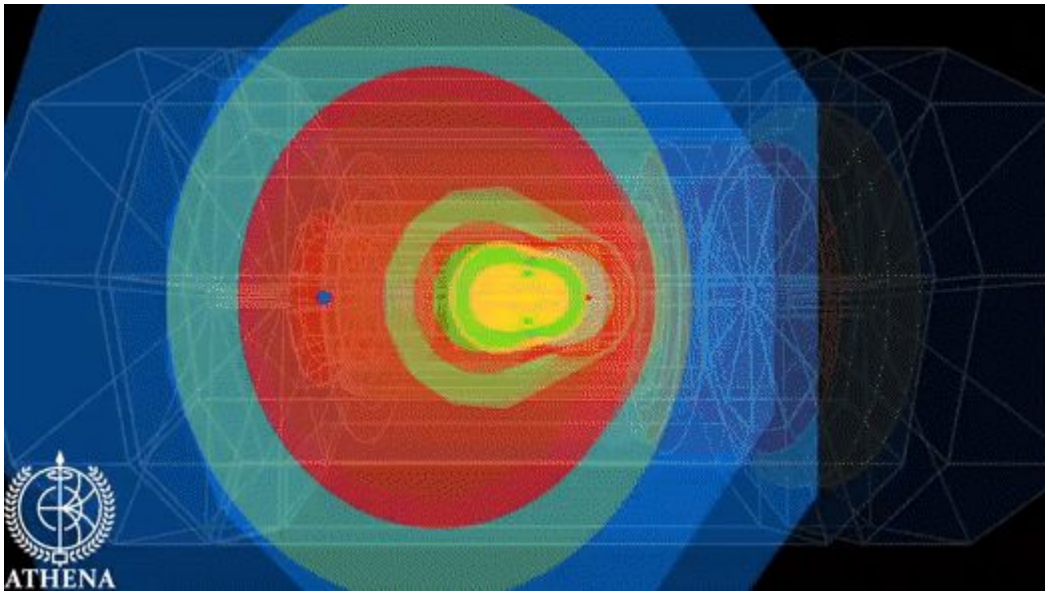
The DD4hep Community





Showcase: Modern toolkit enables creativity

ATHENA 3D VR display created by PhD student Sean Preins (UCR)



Structuring the toolkit as a set of modular, orthogonal tools (independent geometry description, independent detector simulation, independent data model, …) enabled our colleagues to make use of our environment in creative new ways … a big win in our book!

DD4hep has an API for both C++ and Python that can be used for a wide range of future applications that we cannot currently envision (e.g., streaming readout, future event builders, …)

How does DD4hep address the geometry requirements?

Requirements

1. ✅ The geometry information **should be the same in both simulation and reconstruction.**
2. ✅ Fast simulation systems should, as much as possible, **be able to use the common exchange format**.
3. ✅ The geometry system **should allow to include misalignment** and more general condition data.
4. ✅ Geometry description format should be **independent of a specific software technology**.
5. ✅ Geometry description **should be modular**. It should be possible to specify different geometry components in isolation with ideally zero dependency between different modules (detectors). Each detector component should have the ability to change the level of detail independent of other parts of the detector system.
6. ✅ Geometry description **should allow to specify logical information** (sensitivity, B-Fields) in addition to the solids, material and placements. In particular, sensitivity is recognized as a critical issue.

… checks all the boxes!

✅ 7. It **should be possible to make the geometry description persistent**. Different equivalent output formats should be supported (e.g. ROOT files, GDML files) and it should always be possible to translate one format into another in a simple manner.

✅ 8. Hits output files produced in a simulation job should be as much as possible self-describing, in particular **it should be possible to locate hits in space** without the need to run the simulation job. A *self-describing* format for the hits would be ideal, but in case this is not possible, the additional libraries to manipulate hits should not depend on the simulation stack used to produce the hits.

✅ 9. It should be possible to **change sensitivity attributes without changing other static aspects of the geometry**.

✅ 10. Geometry exchange format should **allow clients to use a subset of the features clearly stating which are the optional ones.** We should support existing interesting frameworks without discouraging other R&D activities. Since it is difficult to support all use-cases, the minimal set of mandatory elements to support should be clearly specified and what to do with non-supported ones should be stated (e.g. ignore visualization attributes if not needed).

✅ 11. **Support for import from CAD** should be included. Simplified CAD files will be provided via the Detector Menagerie.

✅ 12. Geometry information **should have support for versioning,** also including the Detector Menagerie.

…and many more (e.g. CAD export for integration in project workflows).

In line with the EIC Software Statement of Principles

3. **We will leverage heterogeneous computing:**
   a. We will enable distributed workflows on the computing resources of the worldwide EIC community, leveraging not only HTC but also HPC systems.
   b. 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.
   c. 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**:
   a. 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.**
   b. EIC software will run on the systems used by the community, easily.
   c. **We aim for a modular development paradigm for algorithms and tools without the need for users to interface with the entire software environment.**

- ✓ DD4hep works with a modular approach, meant from the ground up to interface with various simulation and reconstruction/framework components.
- ✓ DD4hep itself is modular. Example: Using the geometry component and reconstruction components does not lock us in to the conditions or calibrations module.
- ✓ Barrier of entry very low. Example: Changing geometries only requires knowledge of DD4hep, which in many cases means knowledge of editing XML files.
- ✓ Track-record of user-friendliness: able to onboard large detector proposal effort (ATHENA) with very good results.
- ✓ User guide, documentation, tutorials, active worldwide community.

In line with the EIC Software Statement of Principles

6. **We will have reproducible software:**
   a. Data and analysis preservation will be an integral part of EIC software and the workflows of the community.
   b. We aim for reproducible, re-usable, and re-interpretable analyses.

7. **We will embrace our community:**
   a. EIC software will be open source with attribution to its contributors.
   b. We will use publicly available productivity tools.
   c. EIC software will be accessible by the whole community.
   d. 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.
   e. We will not reinvent the wheel but rather aim to build on and extend existing efforts in the wider scientific community.
   ~~f. We aim for a modular development paradigm for algorithms and tools without the need for users to interface with the entire software environment.~~
   g. We will support the community with active training and support sessions where experienced software developers and users interact with new users.
   h. We will support the careers of scientists who dedicate their time and effort towards software development.

- ✓ Methodical approach to geometry conducive of reproducibility.
- ✓ Large community effort in Particle Physics, used at major LHC experiments, with proven track record of user support.
- ✓ Avoid unnecessary use of small home-cooked software solutions.

In line with the EIC Software Statement of Principles

8. **We will provide a production-ready software stack throughout the development:**
   a. We will not separate software development from software use and support.
   b. We are committed to providing a software stack for EIC science that continuously evolves and can be used to achieve all EIC milestones.
   c. **We will deploy metrics to evaluate and improve the quality of our software.**
   d. We aim to continuously evaluate, adapt/develop, validate, and integrate new software, workflow, and computing practices.

- ✓ DD4hep well-integrated in one of the major software stacks used for EIC detector design.
- ✓ Can easily be with any of the frameworks we will consider (fun4all, Gaudi, …)
- ✓ Project maturity and widespread support in HEP community make it a solid choice to use for all stages of EIC development, from TDR to run-time.

Path forward

- Detector 1 Reference Design implementation almost ready for validation, not much work needed to make October deadline (simulations starting in weeks!).
- Half of the collaboration already trained, and track-record of onboarding new people quickly.
- HEP converging on a supported modular stack of modern tools (Key4hep project), should seriously consider Key4hep in the next steps of our decision-making process. E.g. strong compatibility between DD4hep, ACTS, and EDM4hep.
- I strongly believe DD4hep is the only modern solution that really hits all the requirements.