The EIC Single Software Stack: Experiences from ePIC

Designing a Scientific Software Environment for the 2030s

Wouter Deconinck (University of Manitoba)

ePIC Computing & Software Working Group

Supported in part by NSERC SAPIN-2020-00049, SAPPJ-2021-00026.
With input from A. Bressan, M. Diefenthaler, C. Fanelli, T. Horn, S. Joosten, D. Lawrence, W. Li, J. Osborn, Z. Tu, T. Wenaus.

Milestones in the Electron-Ion Collider Development

- **2021:** Large detector proposal development:
  - **ATHENA**: 3T solenoid, Si+MM+GEM tracker, imaging barrel EM cal, proximity-focused RICH
  - **ECCE**: 1.5T BaBar solenoid, Si+muRWell trackers, projective SciGlass EM cal, modular RICH

ATHENA: A Totally Hermetic Electron-Nucleus Apparatus

ECCE *EIC Comprehensive Chromodynamics Experiment*

  - CORE: smaller effort focused on specific exclusive reaction channels at 2nd IR
- **2022:** Selection of ECCE proposal as reference for EIC project detector
  - DPAP advisory panel: ECCE design achieves physics goals with lowest risk and cost
  - Successful integration of ATHENA and ECCE communities within two months(!)
  - Resulting in the formation of the EPIC detector collaboration
- **2023:** Detector TDR for EIC Project CD-2/3a review (by January 2024)
  - 2022: technology selection for few areas where multiple options
  - 2023: finalization of design parametrization

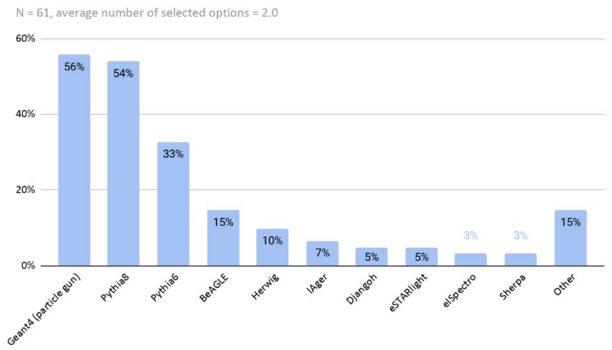
Milestones in the Electron-Ion Collider Software Development

- **2016**: EIC Software Consortium as part of EIC Generic R&D program
    - Activities included: interoperability and common interfaces between simulation components
    - Produce consensus-based community documents setting out vision for EIC software
- **2019**: EIC User Group Software Working Group
    - Community endorsement of software as a valuable endeavor for the EIC user group
    - Focus on preparing the community for detector (proto-)collaborations
        - Coordinate during Yellow Report preparation
        - Coordinate during proposal development process
- **2022**: EPIC Computing & Software and Simulation, Production & QA WGs
    - Single software stack decision process, together with EICUG SWG
    - Short term goals in organizing collaboration around single software stack: at best half of the collaboration will need to learn a lot of new things (at worst the whole collaboration).
- **Overall goals**: development of a community-supported full-lifecycle software stack for nuclear physics; prevent fragmentation; encourage modularity

User-Centered Design at the EIC

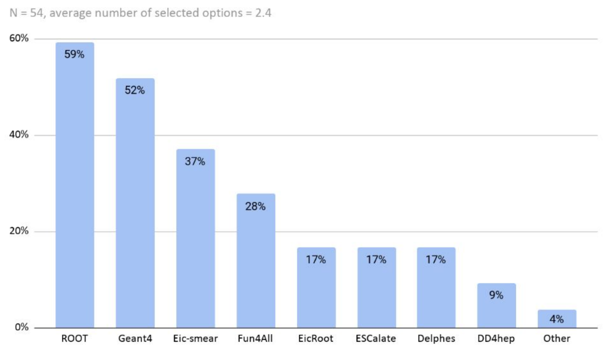
- Annual “State of EIC Software” exercise (2021, 2022)
  - Quantitative survey with question consistency from year to year
  - Qualitative focus groups (~5 users each) to drill into recurring themes
- Development of user personas to highlight diversity of experiences

Over the past year, which physics event generation tools did you use for EIC simulations?



Other (N = 9): personal computer codes (N = 2), ACT, CLASDIS, ComptonRad, GRAPE-DILEPTON, MADX, MILOU, OPERA, RAYTRACE, Sartre, Topeg, ZGOUBI

Over the past year, which detector simulation tools did you use for EIC simulations?



Other (N = 2): GEMC, RAYTRACE

User-Centered Design at the EIC

User-Centered Design:

- State of Software Survey
- Follow-up Focus Groups
- Develop Testing Community

Discoverable Software:

- Single Point of Entry (~ key4hep)
- Feasible Option for > 80% of EIC Simulations and Analyses

Workflows:

- Template Repositories for Key Analyses
- Template Repositories and Validation Workflows

Data and Analysis Preservation:

- User Analysis Code/Software Registry
- Tutorials on Reproducible Analyses

“Key4hep: a framework for future HEP experiments and its use in FCC”, arXiv:2111.09874 [hep-ex]

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:

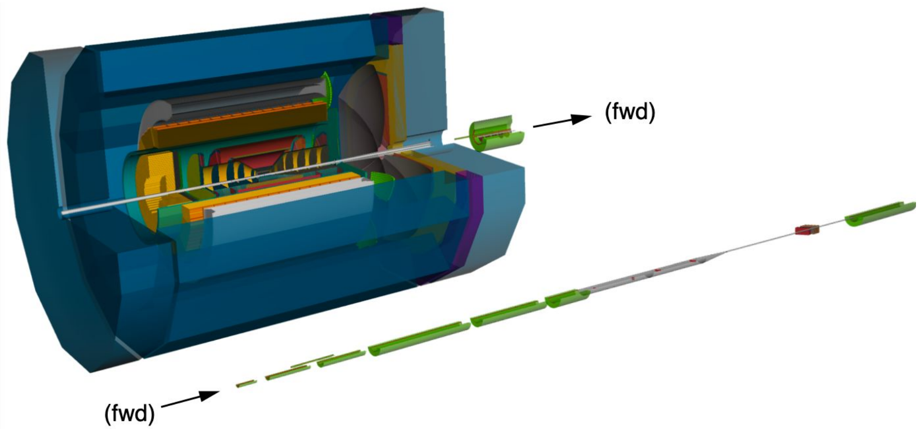
- We will not separate software development from software use and support.
- We are committed to providing a software stack for EIC science that continuously evolves and can be used to achieve all EIC milestones.
- We will deploy metrics to evaluate and improve the quality of our software.
- We aim to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

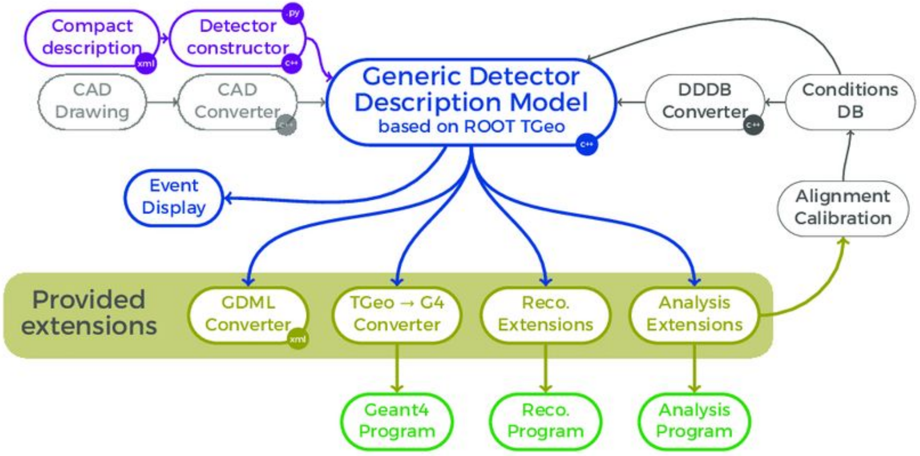
The "Statement of Principles" represent guiding principles for EIC Software. They have been endorsed by the international EIC community. For a list of endorses, see LINK.

Geometry Description: DD4hep

Requirement: **consistent geometry** for simulations, reconstruction, data taking

- DD4hep: Abstraction layer for Geant4, TGeo, and other geometry consumers
- Geometry service from simulation through reconstruction and analysis
- Community-managed external project with large experimental user base

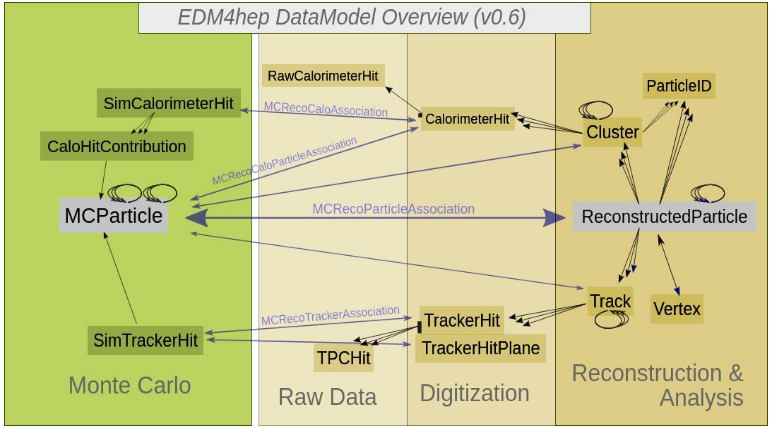


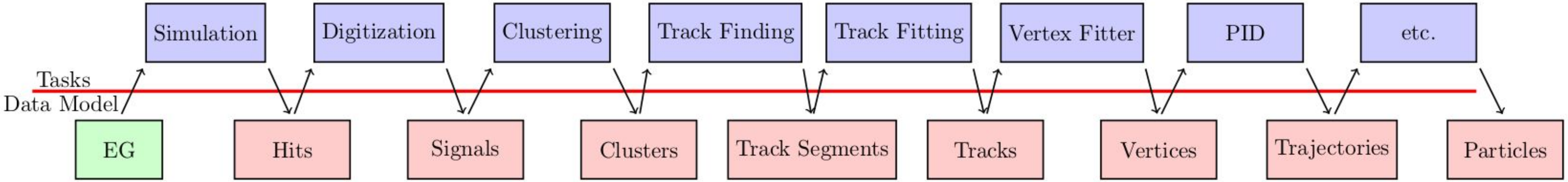


Data Model: podio, EDM4hep, EDM4eic

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

- podio (github.com/AIDASoft/podio)
  - Text-based definition of flat data models
  - Automatic C++ and Python interfaces
  - Stored inside ROOT files or other formats
- EDM4hep (github.com/key4hep/EDM4hep)
  - Defined using podio, based on LCIO and others
  - Designed as a standard for current/future HEP
  - Extensible with EIC-specific data types (EDM4eic)

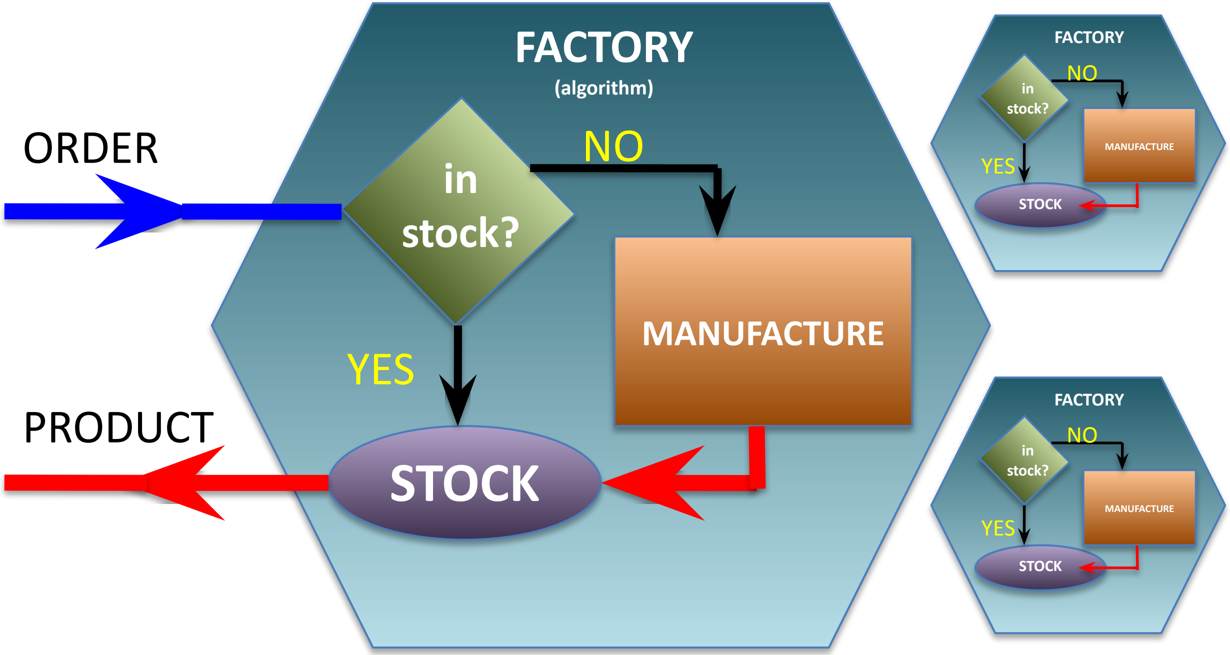




Reconstruction Framework: JANA2

Requirement: **Streaming-readout reconstruction** of simulated or real data, enabling use of heterogeneous computing resources and of machine learning

- Factory model: only reconstruct when not in stock
- JANA2 used by GlueX and streaming readout development projects
- Algorithms depend on data model only, but JANA2 schedules and orchestrates



Code Repositories and Continuous Integration

Code Repository:

Centralized collaborative development of all software components, for preservation of a full record of the development activity.

Several widely used options based on `git`:

- GitHub (github.com or enterprise instance)
- GitLab (gitlab.com or self-hosted instance)
- Others…

We use a GitHub organization, github.com/eic

Milestones and versioning, reproducibility, preservation, collaboration, code review

Continuous Integration/Deployment (CI/CD):

A strategy of **automatic evaluation** of software components, and of **automatic deployment** into testing and production environments.

Tightly integrated with repositories:

- GitHub
- GitLab

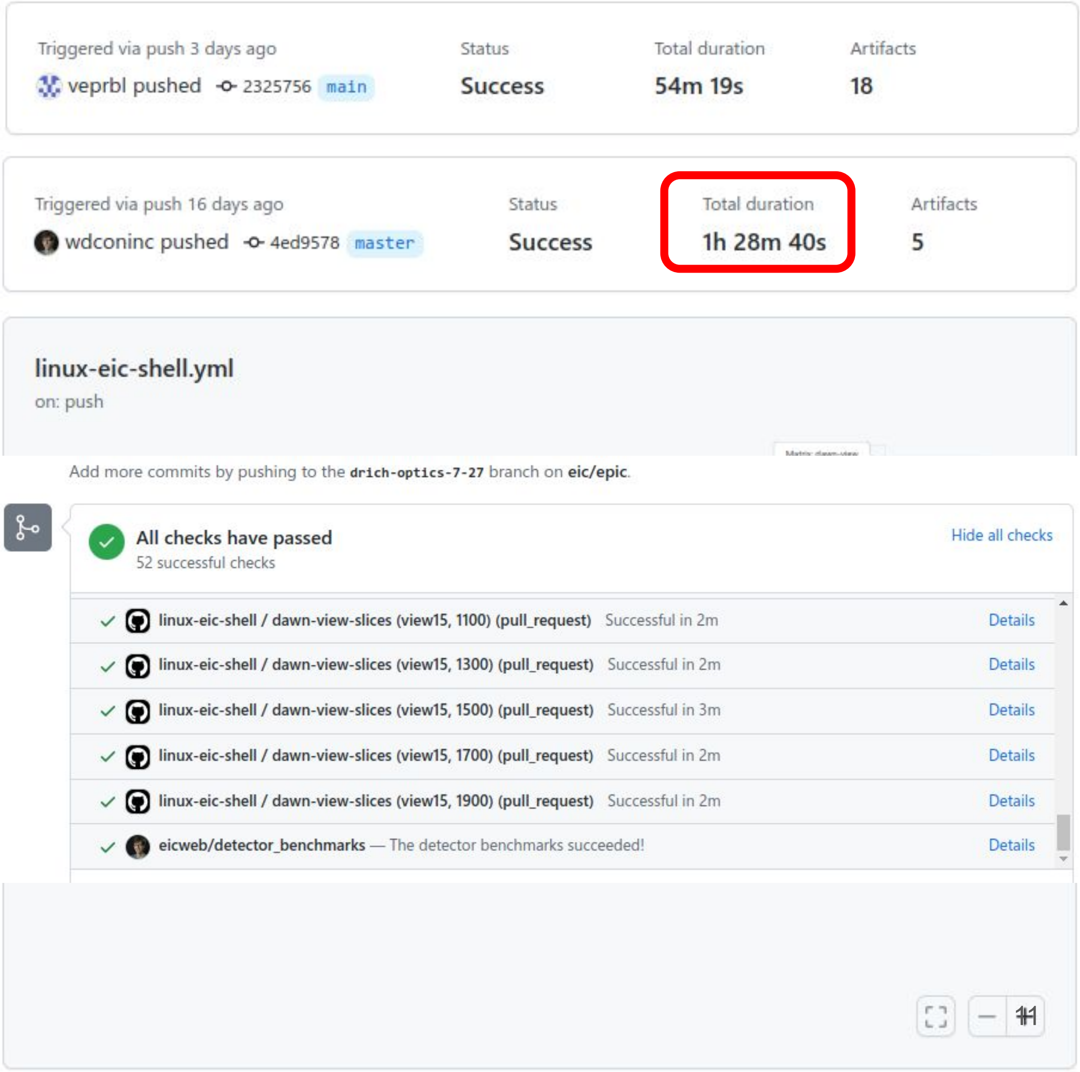
Or as a separate service.

We use GitLab instances triggered from GitHub.

Automation, quality control, workflows, deployment into production environments

GitHub Pipelines

- ePIC: ~50+ jobs on GitHub
- Community-supported GitHub Actions infrastructure:
  - eic/trigger-gitlab-ci
  - eic/run-cvmfs-osg-eic-shell
  - AIDAsoft/run-lcg-view
  - cvmfs-contrib/github-action-cvmfs
- Inside these pipelines, all read-only interactions happen with CernVM global filesystem



Software for *all* EIC experiments - and beyond

- Nothing about the modular software toolkit design is unique to Detector 1
- We explicitly expect the toolkit to be used as a starting point for the Detector 2 software toolkit
- Many design decisions were taken to explicitly allow collaboration and even algorithm sharing with other NP and HEP experiments
- The EPIC software stack could be used for future NP experiments, e.g. SoLID at Jefferson Lab (a fixed-target experiment!)

Component	**Modification for detector 2?**
Geometry	New configuration, can reuse many detector components
Data model	Identical
Framework	Can reuse/add to algorithms, only need different configuration
Code repository and CI	Same resources could be used
Data analysis and preservation	Same strategy can apply
AI and ML	Same strategy can apply

Bottom line: detector 2 can hit the ground running!

Summary: EIC Software Stack

Geometry description with **DD4hep** enables consistent geometry between simulated data, real data, and their reconstruction.

A defined data model with **podio**, based on **EDM4hep**, allows seamless exchange of data and innovation in algorithm development thanks to modularity.

Reconstruction of data streams using **JANA2** with modular algorithms enables running on heterogeneous computing systems to take advantage of parallelism.

These software components are applicable to all experiments that use Geant4 simulations and event reconstruction. We aim for modularity and separation of function from code to allow other new experiments to join in these efforts.

By reusing existing tools, we can focus on what is unique to our EIC needs.