EIC Software Tutorial

Designing a Scientific Software Environment for the 2030s, for all of EIC

Wouter Deconinck (University of Manitoba)

ePIC Computing & Software Working Group

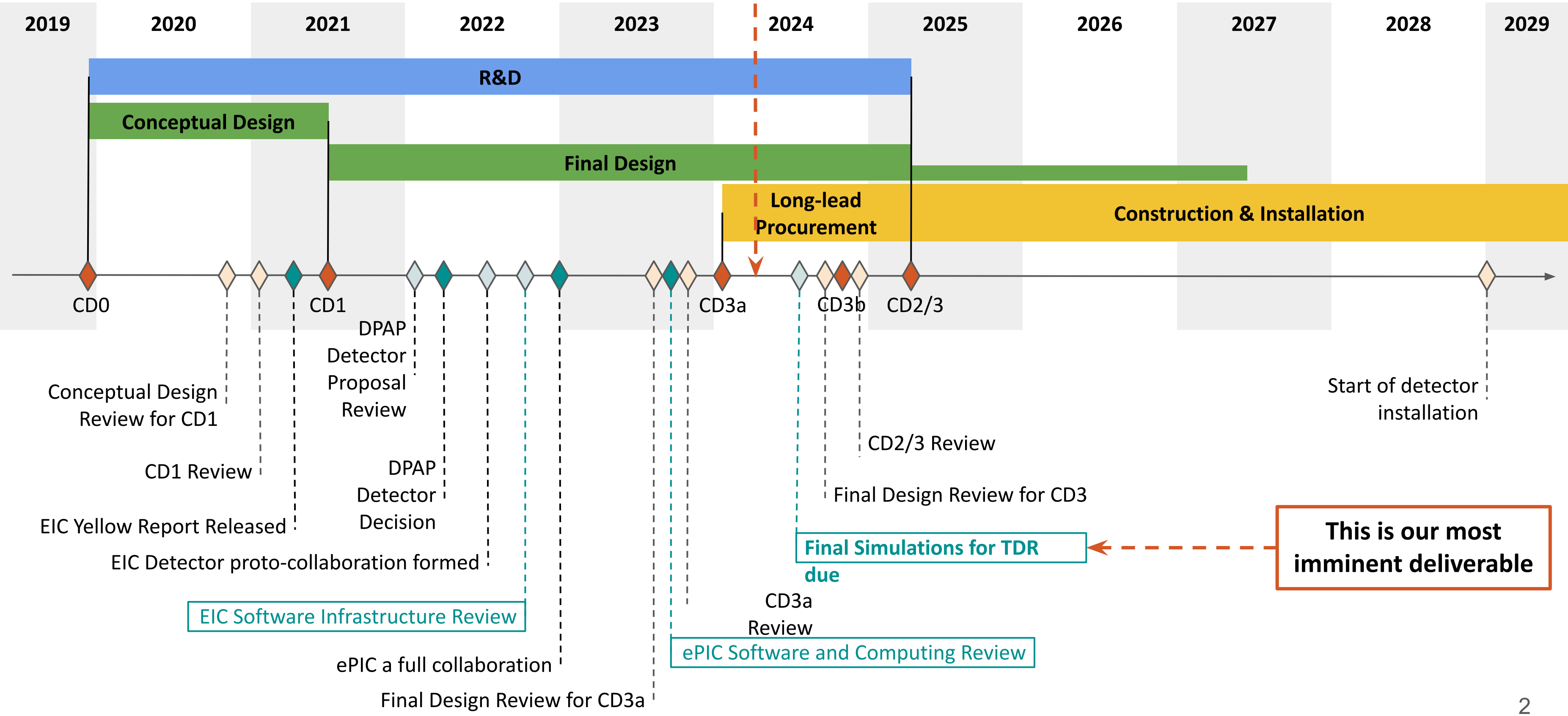
Supported in part by NSERC SAPIN-2020-00049, SAPPJ-2021-00026.
With input from A. Bressan, M. Diefenthaler, C. Fanelli, T. Horn, S. Joosten, D. Lawrence, W. Li, J. Osborn, Z. Tu, T. Wenaus.

Introduction

EIC Timeline

We are here

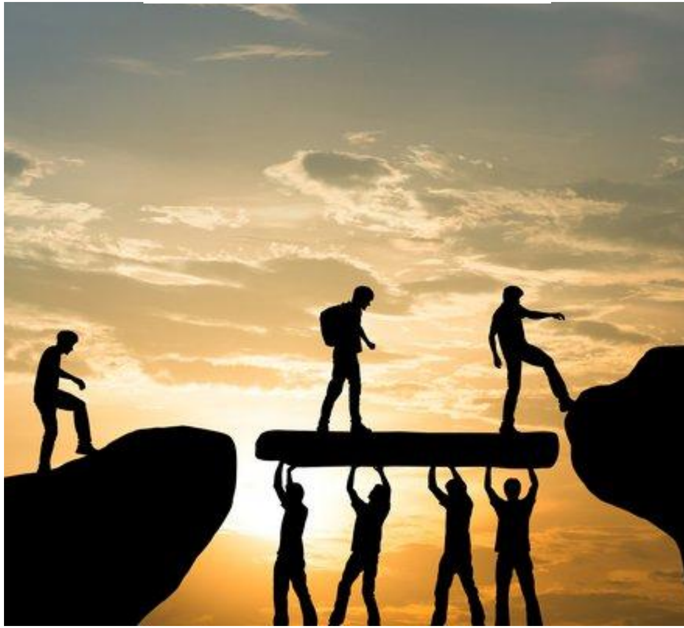
Not shown:
Early CD4 (Oct 2032)
CD4 (Oct 2034)



Introduction

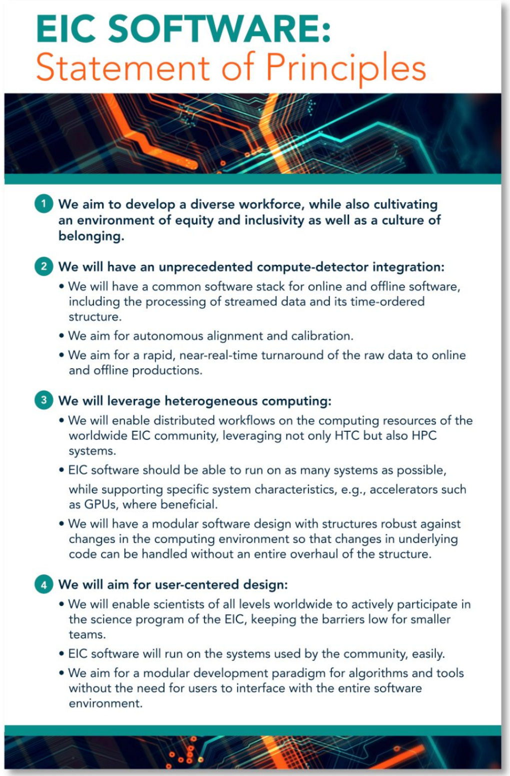
Our Philosophy

- We focus on **modern scientific software & computing practices** to ensure the long-term success of the EIC scientific program throughout all CD milestones
  - Strong emphasis on modular, orthogonal tools
  - Integration with HTC/HPC, CI workflows, and enable use of standard data science toolkits
- We leverage **cutting edge sustainable community software** where appropriate, avoiding the “not invented here” syndrome
  - Can build our software on top of a mature, well-supported, and actively developed software stack by using modern community tools, e.g. from CERN, the HPC community, and the data science community
  - Actively collaborate with external software projects, while externalizing some support burden to external projects
- We embrace these practices today to avoid starting our journey to EIC with technical debt.
- **We are writing software for the future, not the lowest common denominator of the past!**



Introduction

EIC Software: Statement of Principles



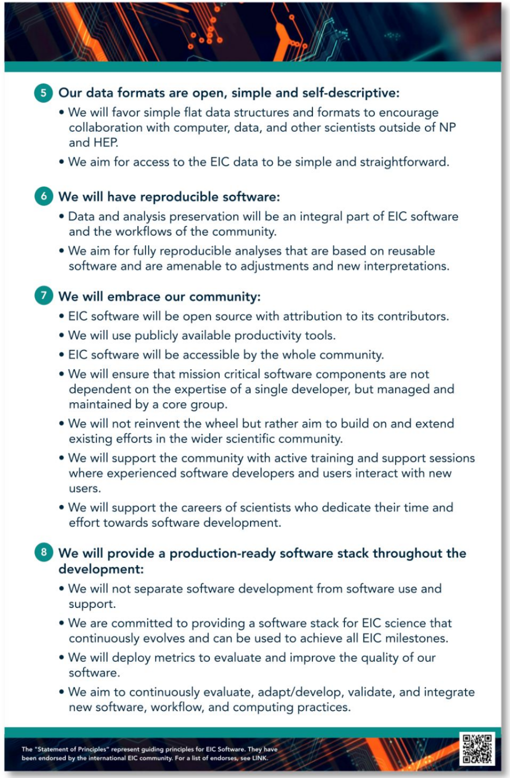
EIC SOFTWARE: Statement of Principles

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.

- Community document that encodes our aspirations (technical and cultural) for software and computing at the EIC
- The foundation of the ePIC Software Stack
- Co-written and endorsed by a large group representing the international EIC community

EIC Software: Statement of Principles

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.

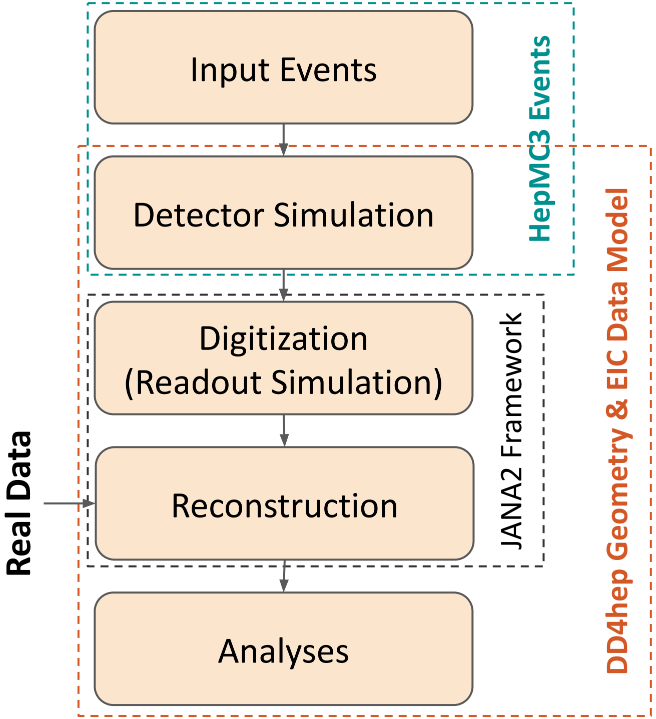
EIC Software is:

1. Diverse
2. Integrative
3. Heterogeneous
4. User-centered
5. Accessible
6. Reproducible
7. Collaborative
8. Agile

Software Stack Overview

A modular simulation, reconstruction, and analysis toolkit

The ePIC Software Stack



Input events from MC event generators or particle guns, with optional physics background merging

Full detector simulations driven by Geant4 and DD4hep, into EIC Data Model output (EDM4hep/EDM4eic, defined in Podio)

Digitization algorithms to mimic real detector readout from Geant4 hits, including background events, “pileup”, DAQ frames

Realistic reconstruction algorithms starting from raw detector output (from digitization or real data)

User analyses in plain C++/ROOT or Python/uproot, facilitated by using a flat data model, enabling use by anyone anywhere

Continuous integration for detector and physics benchmarks and regular **monthly production campaigns** ensure a production-ready software stack

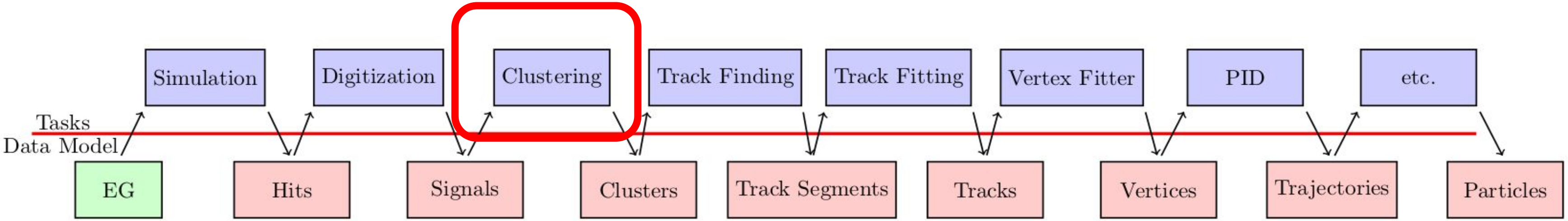
What is a Data Model?

- **Standardized data structures that we collectively agree to use to pass information between simulation, reconstruction, and analysis algorithms**
  - Example: The information we talk about when we say ‘a hit in a tracking detector,’ such as channel number, energy deposition, time, position, etc…
  - The data model is the “protocol” that the components in our software stack use to talk to each other
- This does **not** include: Decisions about the input/output file format, memory layout, or the physical data storage medium
  - Example: Our choice of data model does not require storage in ROOT files (but can be written to ROOT files, HDF5 files, and many others), does not require C++ (or Python), does not require row-oriented memory layouts (but allows for GPU processing), etc…
- **We aim for flexibility through our choice of data model.**

Standardized interfaces between components ensure modularity

Data-Driven API Design and the EIC Data Model

- Use of **standard interfaces** between individual simulation, reconstruction, and analysis tasks **creates modularity** that enables **easy exchange of components**
- Example: multiple clustering algorithms can be swapped out, as long as they adhere to the data model interfaces
- We standardized on **HepMC3 for Monte Carlo input** and **EDM4eic** within our software stack (an extended version of EDM4hep from Key4HEP project at CERN).
- This modularity extends beyond the EIC community, since many data structures are common across NP and HEP experiments worldwide; reuse of CERN methodologies



Open source community data model toolkit for NP & HEP

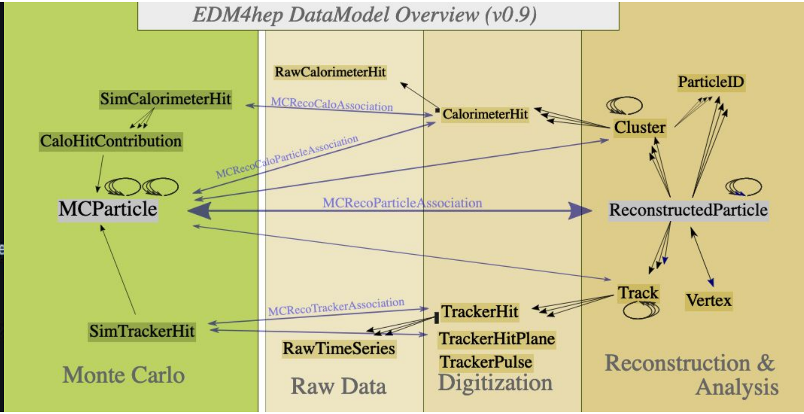
Podio, EDM4hep, and EDM4eic

- Podio is a community tool to define data models in a human-readable format
- EDM4hep implements a standard data model for HEP using Podio
- EDM4eic is a set of EIC-specific extensions to EDM4hep

} Key4HEP stack

- All components of the data model are **open source** and supported by **multiple institutions and collaborations** with goals aligned with ours
- **ePIC is closely aligned and actively involved with Key4HEP at CERN**

```
#----------- SimTrackerHit
edm4hep::SimTrackerHit:
  Description: "Simulated tracker hit"
  Author: "F.Gaede, DESY"
  Members:
    - uint64_t cellID      //ID of the sensor that created this hit
    - float EDep                     //energy deposited in the hit [GeV].
    - float time                     //proper time of the hit in the lab frame in [ns].
    - float pathLength               //path length of the particle in the sensitive material that re
    - int32_t   quality                  //quality bit flag.
    - edm4hep::Vector3d position     //the hit position in [mm].
    - edm4hep::Vector3f momentum     //the 3-momentum of the particle at the hits position in [GeV]
  OneToOneRelations:
    - edm4hep::MCParticle MCParticle //MCParticle that caused the hit.
```



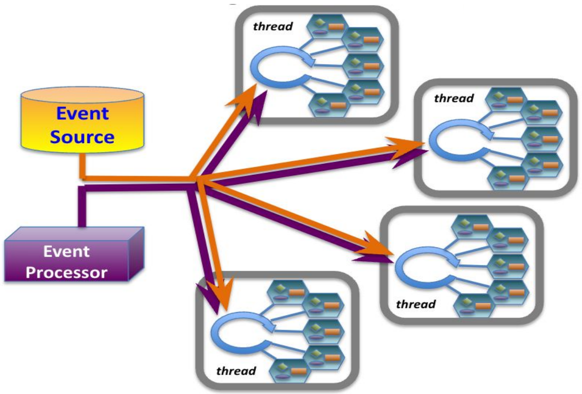
Single source of geometry for simulation and reconstruction

Geometry Description and Detector Implementation

- Simulation and reconstruction both need a single source of geometry
- To ensure modularity this should be provided by an orthogonal toolkit (e.g. not directly connected to the event processing framework)
- DD4hep provides a complete solution for a full detector description (geometry, materials, visualization, readout, alignment, …)
  - Parametrized geometries are a powerful tool for detector optimization
  - A full implementation of the ePIC detector in DD4hep has been used in production for well over a year; second detector effort is using DD4hep as well
  - Our experience has been very positive, in terms of new user onboarding, using DD4hep to drive Geant4 simulations , accessing DD4hep geometry information in reconstruction algorithms, and collaborating with the DD4hep developers
- EIC-specific npsim library configures EIC physics, optical photon settings specific to our Cherenkov PID detectors,…
- **ePIC has become a major contributor to the DD4hep project** (bugfixes and new features)

Reconstruction Framework

- We selected JANA2 as the reconstruction framework based on a carefully formed set of requirements reviewed by the EIC software community
  - Natively multithreaded from the start
  - Supports streaming DAQ and heterogeneous hardware
  - Active development and support (1 dedicated FTE from Jefferson Lab to support ePIC)
  - Developed by Jefferson Lab, one of the EIC host labs
  - Over 20 years of production experience between JANA and JANA2 (used in GlueX)
- We implemented EICrecon with the tooling and algorithms needed for ePIC
  - Podio frames support
  - DD4hep geometry support
  - Quasi-framework independent algorithms
  - Algorithm configuration and wiring
- Current mid-term framework developments:
  - Fully framework independent (modular) algorithms
  - Flexible external algorithm wiring and configurability
  - Metadata and conditions handling
  - Better integration of ML in the reconstruction



Community software in reconstruction algorithms

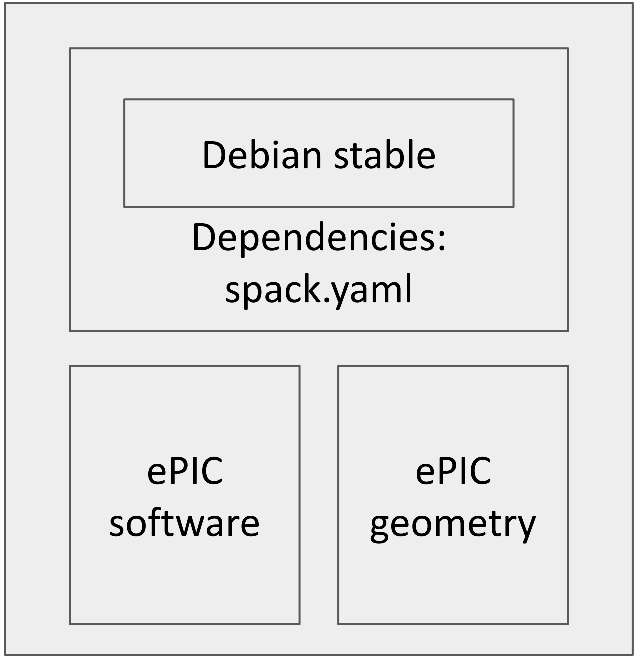
Example: ACTS for Tracking Algorithms

- ACTS is the main toolkit to express our tracking algorithms, we have used ACTS for all our ePIC simulation campaigns
- ACTS integrates almost seamlessly with our DD4hep geometry
- ACTS developers frequently attend out weekly Tracking Reconstruction meetings, which is invaluable
- Solving EIC-unique problems together with the ACTS team
- Many EIC-related commits are now part of the main ACTS codebase
- **Highly positive experience:** the ACTS team has treated us as first-rate “clients” of their project, and we are contributing back code

Enabling User Workflows

Enabling a consistent working environment

Deployment with Containers



- Provide a single curated software build "eic-shell" for local development, CI, and production campaigns
    - Multiple architecture-specific versions of images where needed (e.g. amd64 and aarch64)
    - Build docker image and converted singularity image
- Different flavors:
    - **nightly:** all master branches, built every night
    - **stable/tagged:** release versions
    - **unstable**: temporary containers for Pull Requests
- Distribution:
    - DockerHub & Github Registry: all docker images
    - eicweb: Internal docker images, all singularity images
    - CVMFS: OSG ~6 hour synchronizations, to `/cvmfs/singularity.opensciencegrid.org`

Easy to get started locally… in only 1 line!

Local EIC Software Deployment with `eic-shell`

Step 1: `curl -L get.epic-eic.org | bash`

Step 2: ???

Step 3: Profit

- Uses deployed images on /cvmfs when available, downloads singularity sifs otherwise
- Rolling out seamless container updates to end users
- At the same time basis of scalable computing on OSG: same containers are used everywhere.
- Note: In principle not even needed to look at data (flat format!)

Approach has worked robustly for multiple years now. Biggest challenge was making people believe it can really be this simple!

Main software repository for development

GitHub: EIC organization and managed runners

- Recommended standard interface for all source code projects in ePIC
- Modest computational resources:
  - 20 quad-core job slots for all projects under github.com/eic
- User management and workflow:
  - Everyone can get a GitHub account and every EIC user can get EIC organization membership
  - All new contributions happen through a pull request (PR)
  - Code can only be merged if it passess all CI checks and passes expert review
  - All PRs are squash-merged into the main branch to maintain a clear history
  - We strongly encourage users to make small incremental changes to most effectively develop software in a collaborative context
- Additional features:
  - GitHub actions model of development: easily shared across all of GitHub
  - GitHub pages for presentation (e.g. https://eic.github.io/epic/craterlake_views)

Open, collaborative software development

Workflow Philosophy

Encourage Upstream Contributions

- Requirements of well-formed HepMC as input has resulted in real improvements to multiple MCEGs used by EIC community.
- Various upstream contributions to DD4hep, ACTS, Spack, uproot,...

Encourage Social Coding

- CI platform provides the incentive for developers to commit code frequently: achieving data management and analysis preservation goals.
- Pull request reviews to ensure higher quality code and build developer skills.

Enable Access Without Restrictions

- ePIC collaboration members include over 170 institutions worldwide
- Data ‘publicly’ available through BNL S3 and publicly available through JLab xrootd.
- Flat data structures (i.e. could be a csv), stored as ubiquitous ROOT trees without need for data structure libraries.
- Support for uproot using numpy library (not awkward).

Data Analysis Preservation Approaches

- Rucio for data management
- Reproducible analysis workflow tools

Tutorial: Get Out Your Laptops!

(or keep using your laptops but work along with me)

EIC Software Stack: User Tutorials (Current Snapshot)

GitHub: http://github.com/eic and search any repository with 'tutorial' in the name:

- **Setting up your environment**: https://eic.github.io/tutorial-setting-up-environment
  - How to get started with `eic-shell` at your favorite analysis cluster or your laptop
- **Geometry Development with DD4hep**: https://eic.github.io/tutorial-geometry-development-using-dd4hep
  - How to implement new detectors (or modify the description of existing detectors)
- **Simulations using ddsim and Geant4**: https://eic.github.io/tutorial-simulations-using-ddsim-and-geant4
  - How to run full simulations starting from single particles or from HepMC3 events
- **Reconstruction with JANA2**: https://eic.github.io/tutorial-jana2/
  - How to write algorithm factories in JANA2
- **Analysis**: https://eic.github.io/tutorial-analysis/
  - How to analyze reconstruction output from EICrecon
- **Reconstruction algorithms**: https://eic.github.io/tutorial-reconstruction-algorithms/
  - How to write new event reconstruction algorithms

EIC Software Stack: Accessing Through `eic-shell`

On a Linux system (with `bash`, `curl` and possibly `cvmfs`, `singularity` or `docker`)

```
$ curl -L get.epic-eic.org | bash
```

This will install the script `eic-shell` to access the EIC software stack (continuously updated).

```
$ ./eic-shell
jug_xl> wdconinc@menelaos:~ $
```

Accessing the EIC Software on GitHub

- **GitHub Codespaces**: cloud-enabled compute environment, free of charge for any GitHub user (up to a finite number of hours per month, ~4 hours per day)
  - Navigate to https://github.com/eic/python-analysis-bootcamp and click the top right “Code” button
  - Direct link to Codespaces: https://codespaces.new/eic/eic-shell?quickstart=1 (ignore warnings about Copilot)
  - If prompted which Python Environment, choose “analysis, python 3.12”
- **Determining x and Q^2 from the scattered electron**
  - Naive(!) calculation of x and Q^2 based on reconstructed electron kinematics (entirely from tracking)
  - Refer to Tyler’s talk on Monday for why this is not ideal for all kinematic regions
  - Not strictly necessary to do this calculation by hand: of course it is done in our reconstruction too
- **Comparing measured and true x and Q^2 for unfolding**
  - First step in determining bin-to-bin migration (purities, unfolding)
  - Refer to Tyler’s talk on Monday again…