IRT Geometry Integration

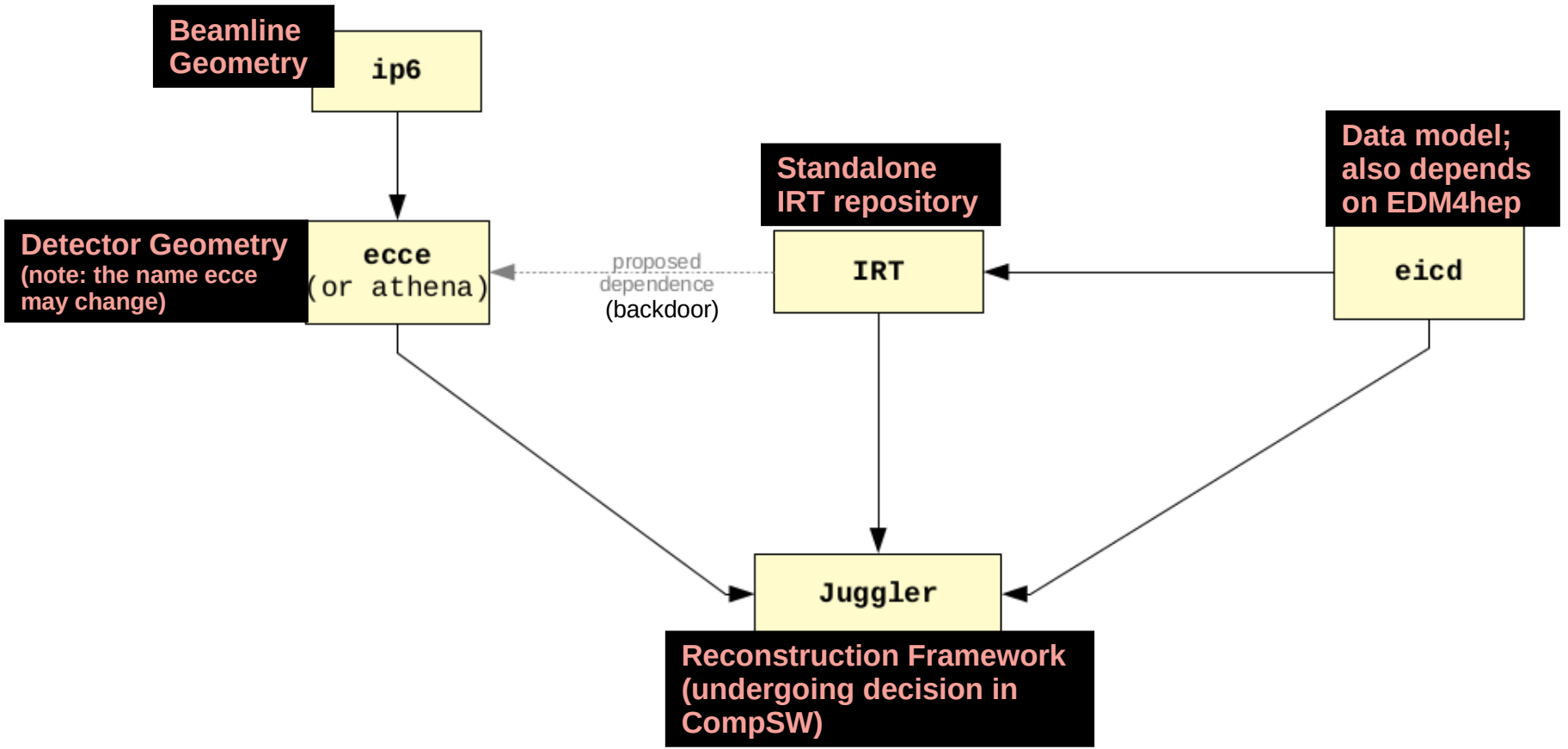
Christopher Dilks
dRICH Meeting
6 July 2022

Motivation

- The IRT code needs to know the geometry, including:
  - Envelope entrance end exit windows
  - Aerogel and filter z-positions and thicknesses
  - Mirror centers and radii
  - Sensor position and normals
  - Radiator refractive indices

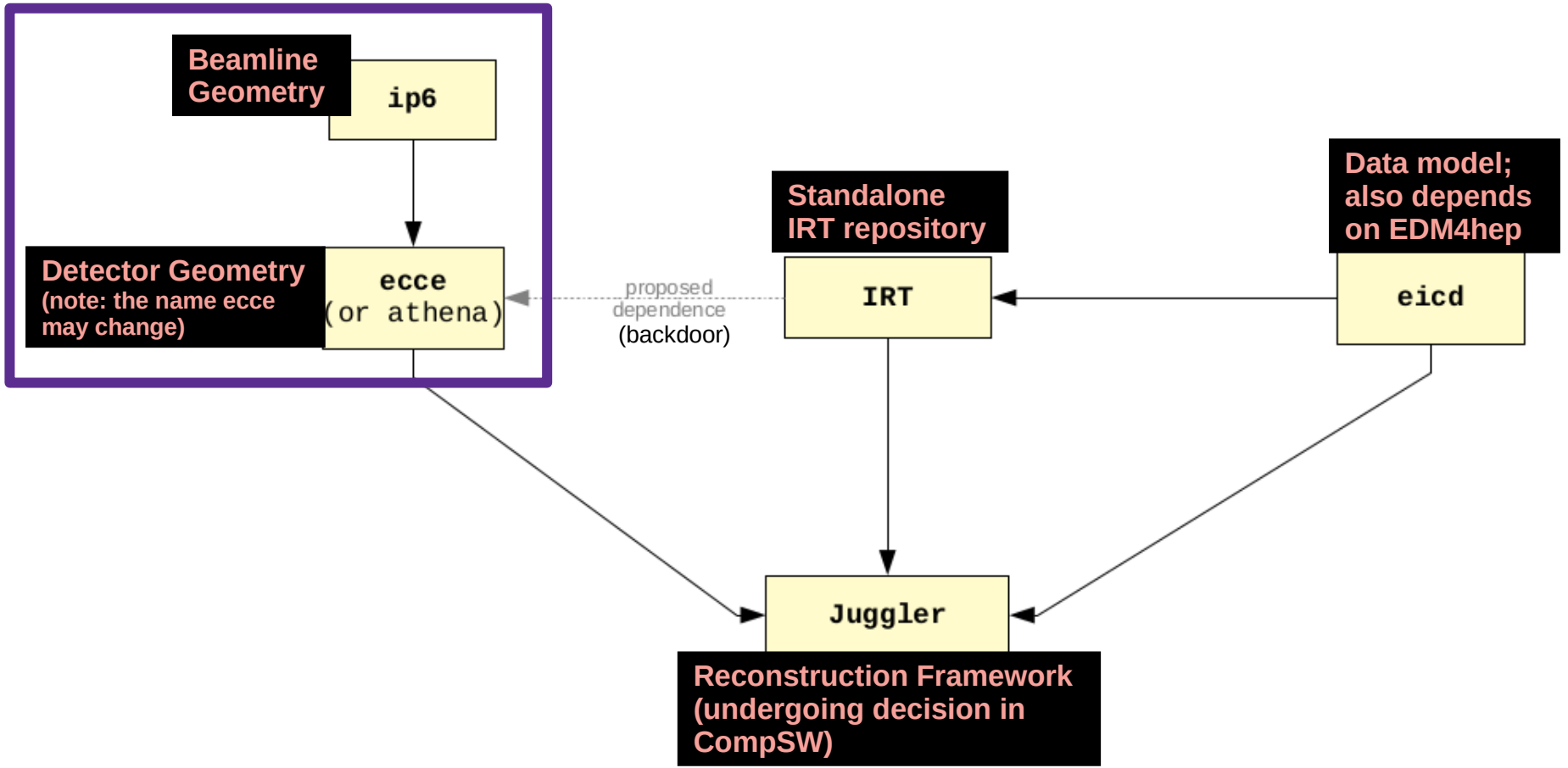
- We have 2 approaches for connecting IRT and geometry:
  - Frontdoor Approach
    - Access the geometry tree from the reconstruction (IRT) algorithm
    - Preferred by CompSW
  - Backdoor Approach
    - Embed IRT geometry objects with detector description
    - Produces auxiliary ROOT file, read by the reconstruction
    - Was our primary approach during the ATHENA proposal

Modules and Dependency



from https://github.com/c-dilks/drich-dev/blob/main/doc/docDiagram.pdf

Modules and Dependency



from https://github.com/c-dilks/drich-dev/blob/main/doc/docDiagram.pdf

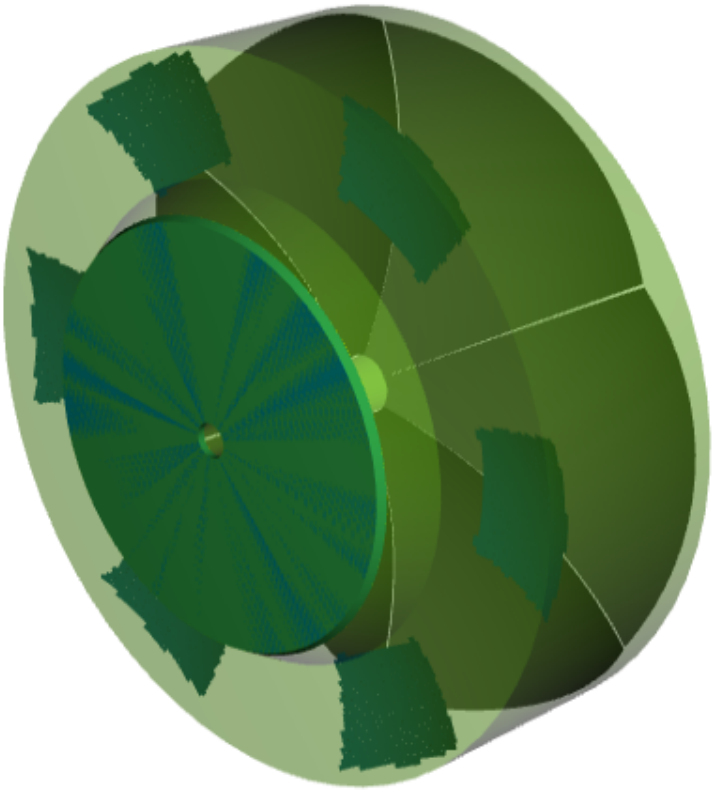
Geometry: ip6 and ecce

https://eicweb.phy.anl.gov/EIC/detectors/ip6

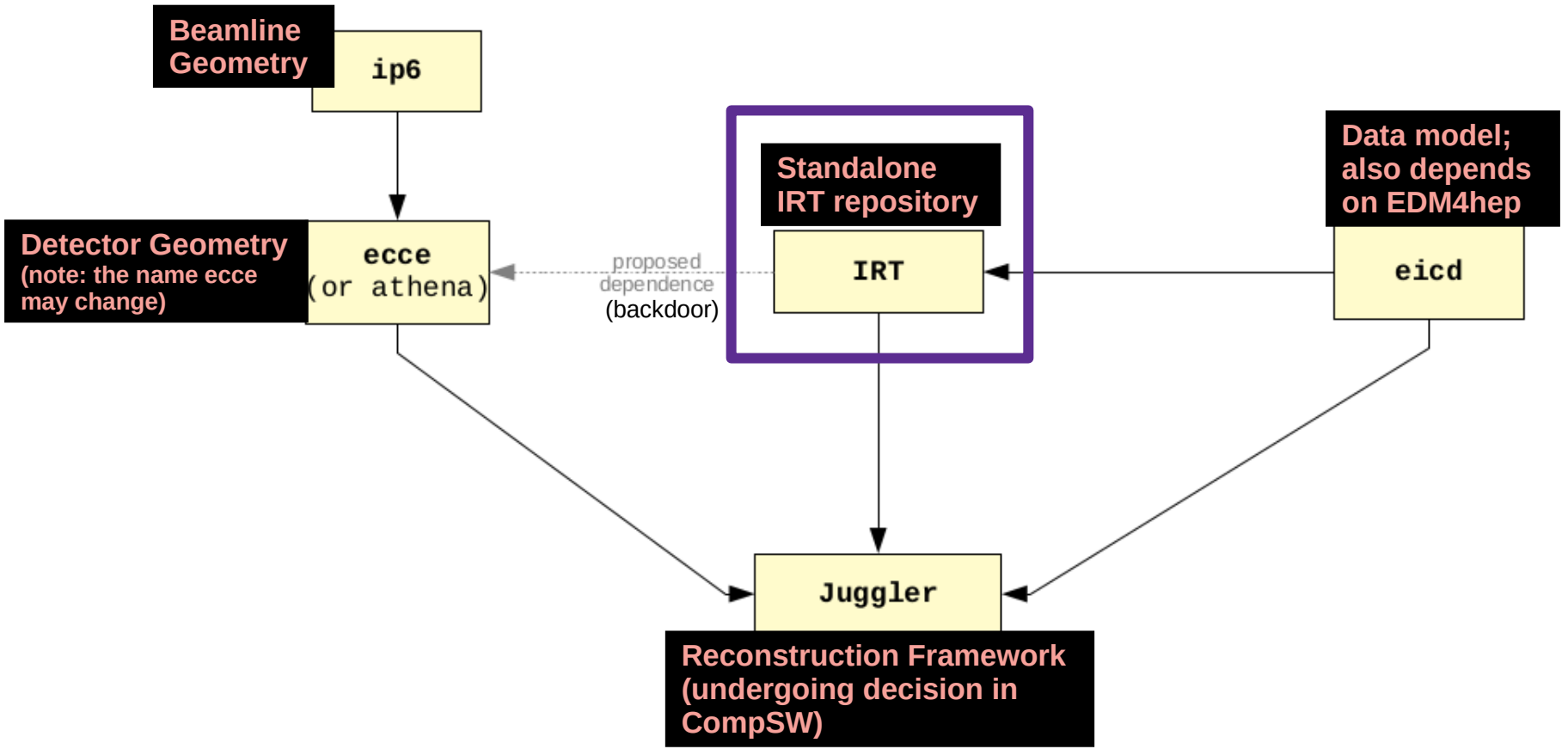
https://eicweb.phy.anl.gov/EIC/detectors/ecce

- dRICH Source file and Compact file
- Material property tables
- All other geometry descriptions

```
<mirror
  material="Acrylic_DRICH"
  surface="MirrorSurface_DRICH"
  vis="DRICH_mirror_vis"
  backplane="DRICH_window_thickness + 0.71*cm"
  rmin="DRICH_rmin1 + DRICH_wall_thickness - 1.0*cm"
  rmax="DRICH_rmax2 - DRICH_wall_thickness - 3.0*cm"
  phiw="59.5*degree"
  thickness="0.2*cm"
  focus_tune_x="69.78*cm"
  focus_tune_z="51.45*cm"
  />
```



Modules and Dependency



from https://github.com/c-dilks/drich-dev/blob/main/doc/docDiagram.pdf

Indirect Ray Tracing: irt

https://eicweb.phy.anl.gov/EIC/irt

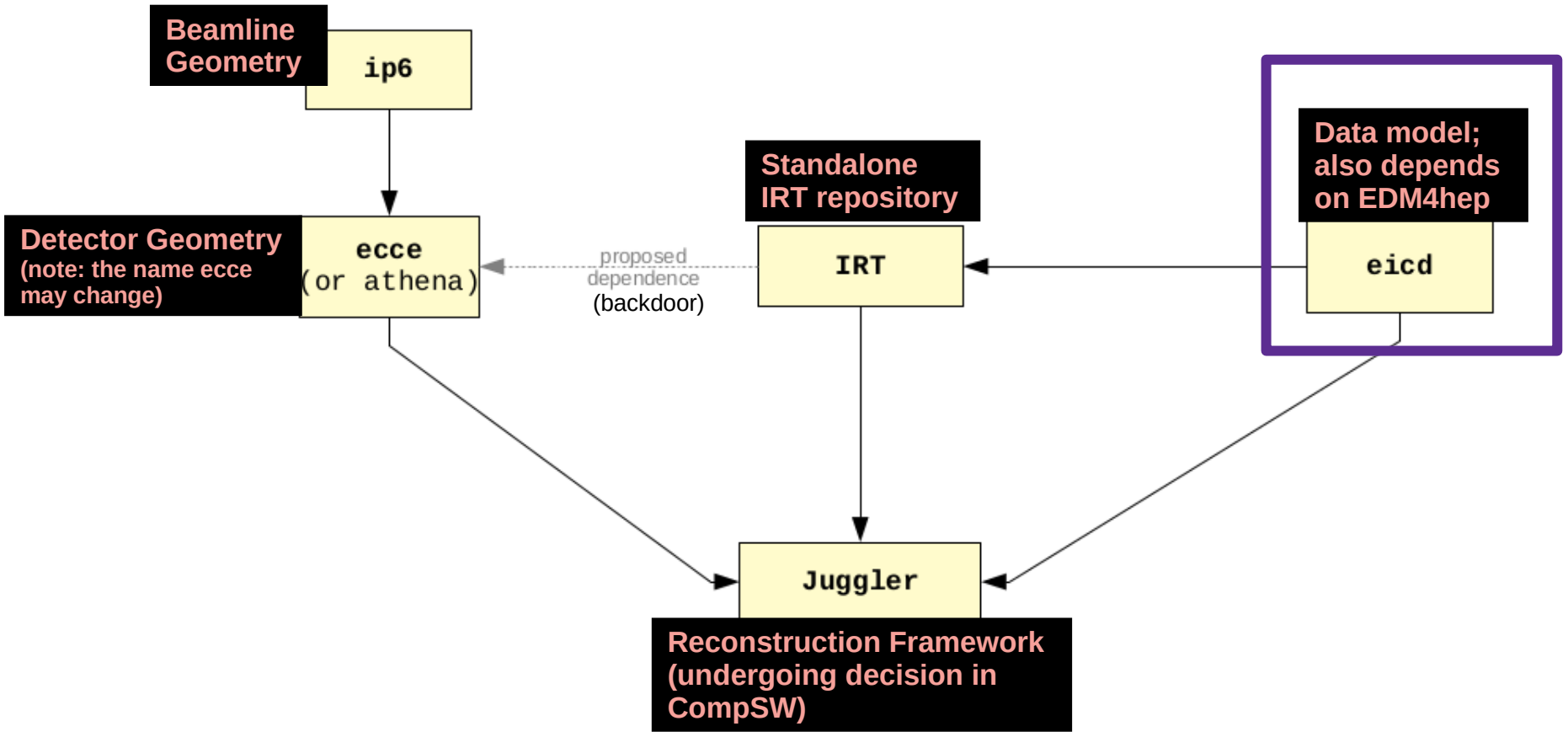
Inverse Ray Tracing implementation in ATHENA

Repository: https://eicweb.phy.anl.gov/EIC/irt/-/tree/irt-init-v02

- A compact C++ library
  - Can be used in a standalone GEANT code as well as in ATHENA environment
  - Optical geometry ROOT class instance is created *in the same code, which creates RICH detector* (therefore simulation-vs-reconstruction consistency is guaranteed)
  - Persistency model: optical setup dump in ROOT format
  - Newton-Gauss iterative solver for optical path defined by arbitrary sequence of refractive and reflective surfaces in 3D (presently flat and spherical boundaries only)
  - Absorbtion length accounting (azimuthally-asymmetric shift of emission point)
  - Emission angle uncertainly calculation (it does depend on the azimuthal angle!)
  - A wrapper for sampling along the charged particle trajectory (magnetic field case, etc.)

Slide from Alexander Kiselev, From meeting on RICH Pattern Recognition Challenges
https://agenda.infn.it/event/30966/

Modules and Dependency



from https://github.com/c-dilks/drich-dev/blob/main/doc/docDiagram.pdf

Data Model: eicd

https://eicweb.phy.anl.gov/EIC/eicd

new components:

MR: https://eicweb.phy.anl.gov/EIC/eicd/-/merge_requests/70

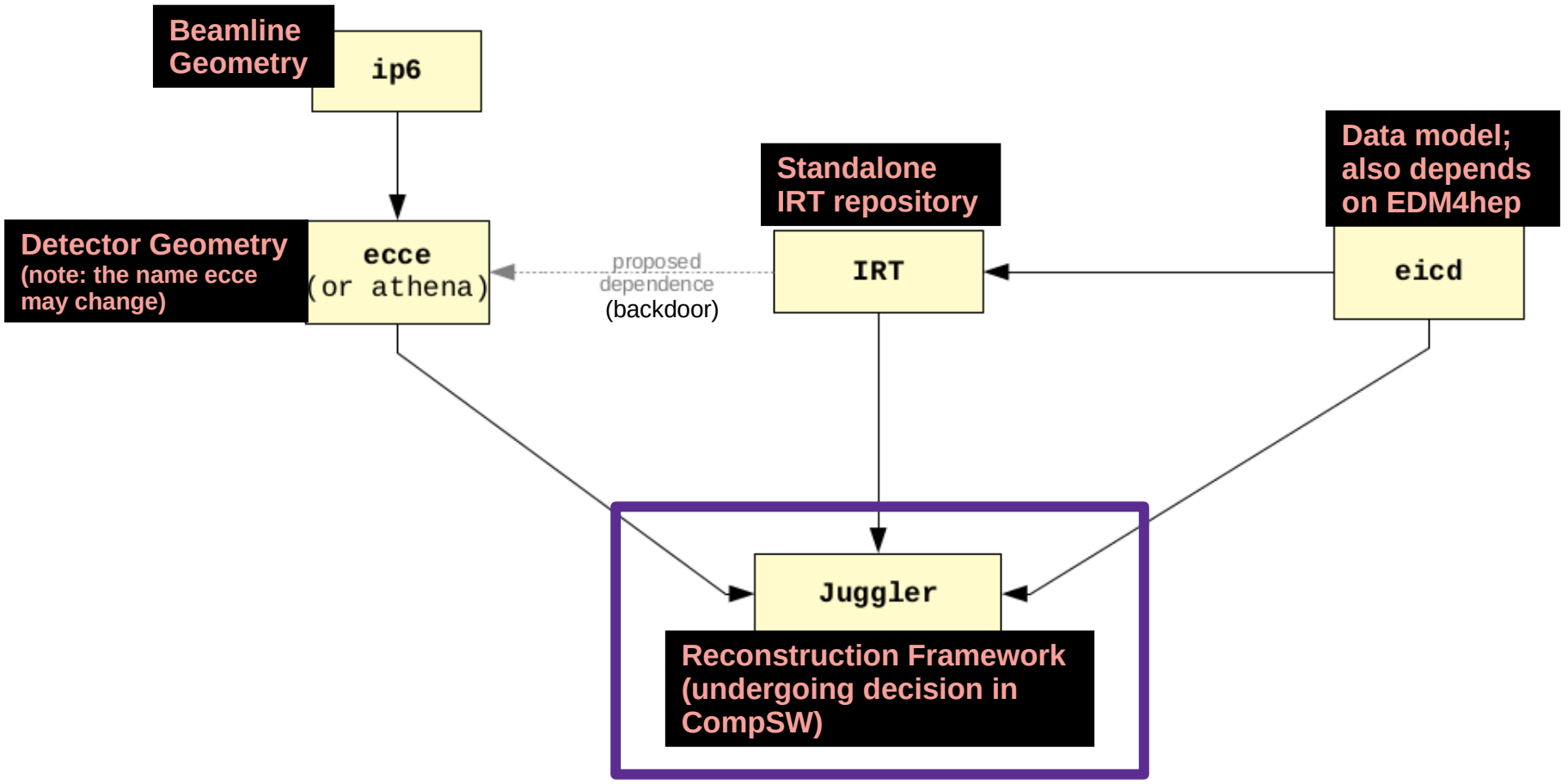
```
## PID hypothesis from Cherenkov detectors
eicd::CherenkovPdgHypothesis:
  Members:
    - char              radiator          // Radiator number (0/1/..) in a sequence of the IRTAlgorithm configuration file
    - int32_t           pdg               // PDG code
    - float             npe               // Overall p.e. count associated with this hypothesis for a given track
    - float             weight            // The weight associated with this hypothesis (the higher the more probable)

## Cherenkov angle measurement for a given radiator
eicd::CherenkovThetaAngleMeasurement:
  Members:
    - char              radiator          // Radiator number (0/1/..) in a sequence of the IRTAlgorithm configuration file
    - float             npe               // Overall p.e. count associated with this estimate
    - float             theta             // Cherenkov theta angle
    - float             rindex            // Average refractive index for this collection of photons
    - float             wavelength        // Average wavelength for this collection of photons
```

new datatypes:

```
eicd::CherenkovParticleID:
  Description: "Cherenkov detector PID"
  Author: "A. Kiselev, C. Dilks"
  VectorMembers:
    - eicd::CherenkovPdgHypothesis options        // Evaluated PDG hypotheses, typically (e/pi/K/p)
    - eicd::CherenkovThetaAngleMeasurement angles // Evaluated Cherenkov angles for different radiators
  OneToOneRelations:
    ## @TODO: should it be one-to-one?
    - eicd::ReconstructedParticle associatedParticle // associated reconstructed particle
```

Modules and Dependency



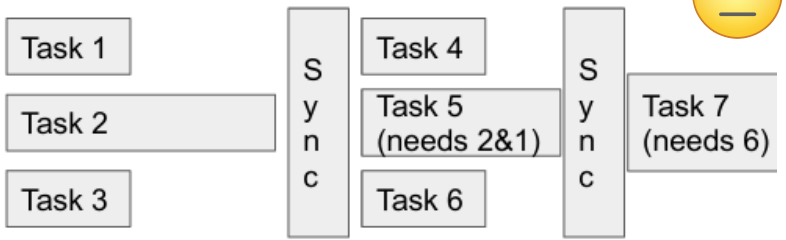
from https://github.com/c-dilks/drich-dev/blob/main/doc/docDiagram.pdf

Reconstruction Framework: Juggler

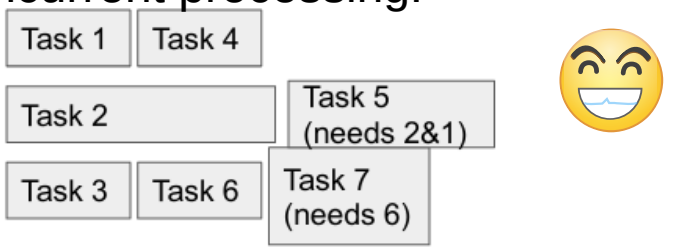
https://eicweb.phy.anl.gov/EIC/juggler

- Reentrant algorithms: https://en.wikipedia.org/wiki/Reentrancy_(computing)
  - Interruptible
  - Reentrancy != thread-safety
- Concurrent execution
- Keep algorithms “small” and “simple” → modularity
- Can we decompose the IRT module into smaller “bite-sized” algorithms? Something to think about for the future…

Parallel processing:

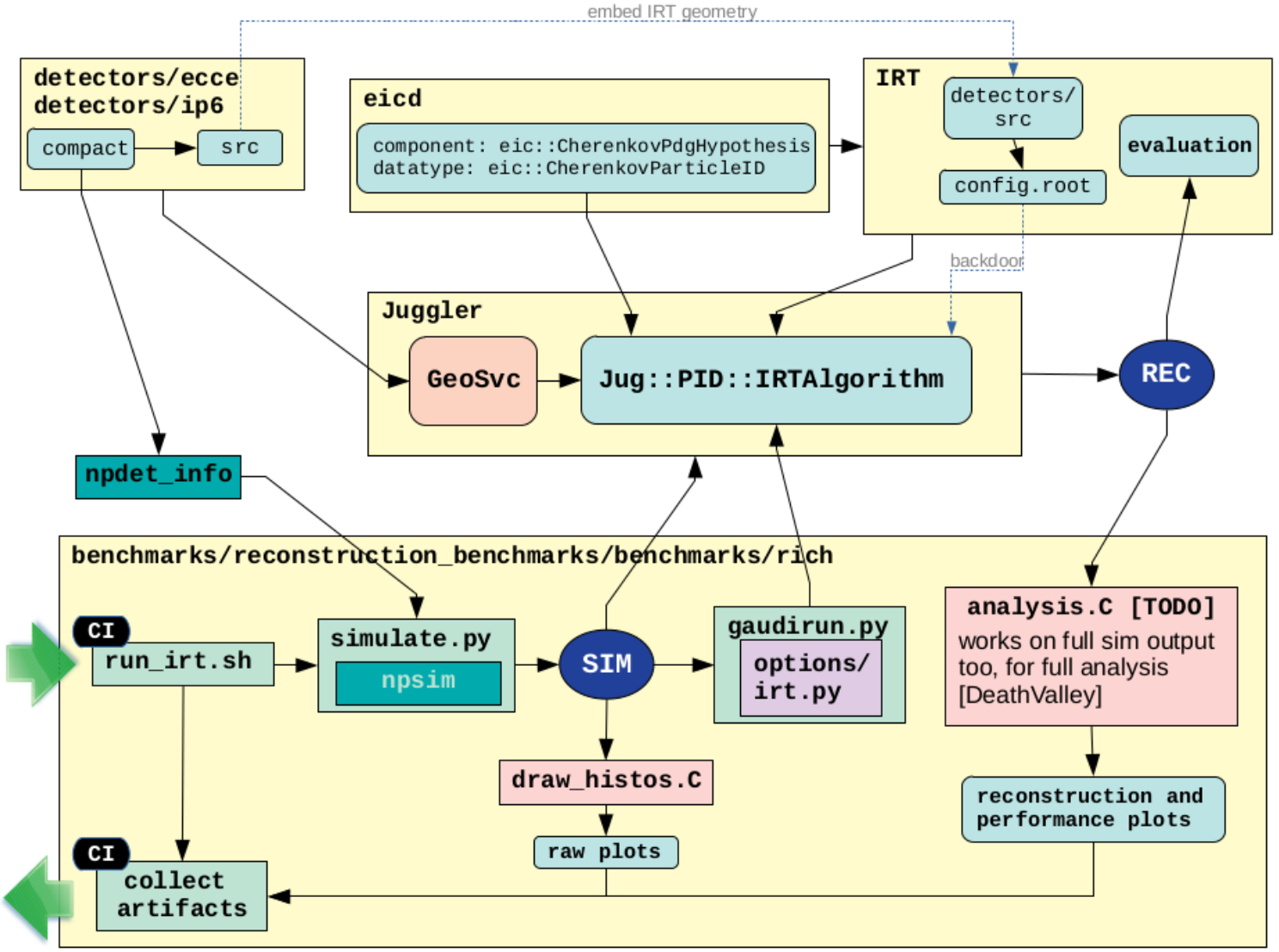


Concurrent processing:



See Sylvester’s slides https://indico.bnl.gov/event/15644/

Details


embed IRT geometry
detectors/ecce
detectors/ip6
compact
src
eicd
component: eic::CherenkovPdgHypothesis
datatype: eic::CherenkovParticleID
IRT
detectors/
src
config.root
evaluation
backdoor
Juggler
GeoSvc
Jug::PID::IRTAlgorithm
REC
npdet_info
benchmarks/reconstruction_benchmarks/benchmarks/rich
CI
run_irt.sh
simulate.py
npsim
SIM
gaudirun.py
options/
irt.py
analysis.C [TODO]
works on full sim output
too, for full analysis
[DeathValley]
draw_histos.C
reconstruction and
performance plots
raw plots
CI
collect
artifacts

IRT Geometry Approaches

Frontdoor: use the Juggler geometry service ‘GeoSvc’

- Pro: read-only, adaptable access the geometry from the reconstruction side
- Pro: thread-safe, concurrent, scales
- Pro: could produce auxiliary file from Juggler too, if we *really* need it
- ✖ Con: while we can keep the geometry reading general, it’s possible some changes in the geometry will need to be accounted for in our GeoSvc usage
- ✖ Con: difficult to traverse Boolean geometry trees, but that’s also difficult in the dd4hep code (in particular, the mirrors)

Backdoor: embed IRT geometry creation within dd4hep geometry

- Pro: embed IRT geometry code within the dd4hep geometry code
- Pro: auxiliary ROOT file (for book-keeping)
- ✖ Con: ecce must depend on irt
- ✖ Con: reading auxiliary ROOT file is not thread-safe (Juggler geometry service *is* safe) → won’t scale

For the long term, we need to use the front door

IRT Geometry Status and Plans

Frontdoor: use the Juggler geometry service ‘GeoSvc’ **LONG TERM**

- Implement this, and use the backdoor auxiliary file as a cross check
- We have an early attempt of this, and can start from there; see https://eicweb.phy.anl.gov/EIC/juggler/-/commit/78f373e228aa7c85e1b25726c8643e46c93a0c2e

Backdoor: embed IRT geometry creation within dd4hep geometry **SHORT TERM**

- Mostly done, we already had this in ATHENA
- Cleaned up and organized DRICH_geo.cpp (see recently merged MRs)
- The MR is now minimal, and *only* has additions
  - Propose to merge to main, and promise to revert it later, after we have a working ‘frontdoor’
  - If not merged, keep branch in sync with main

Merge Request:
https://eicweb.phy.anl.gov/EIC/detectors/ecce/-/merge_requests/31