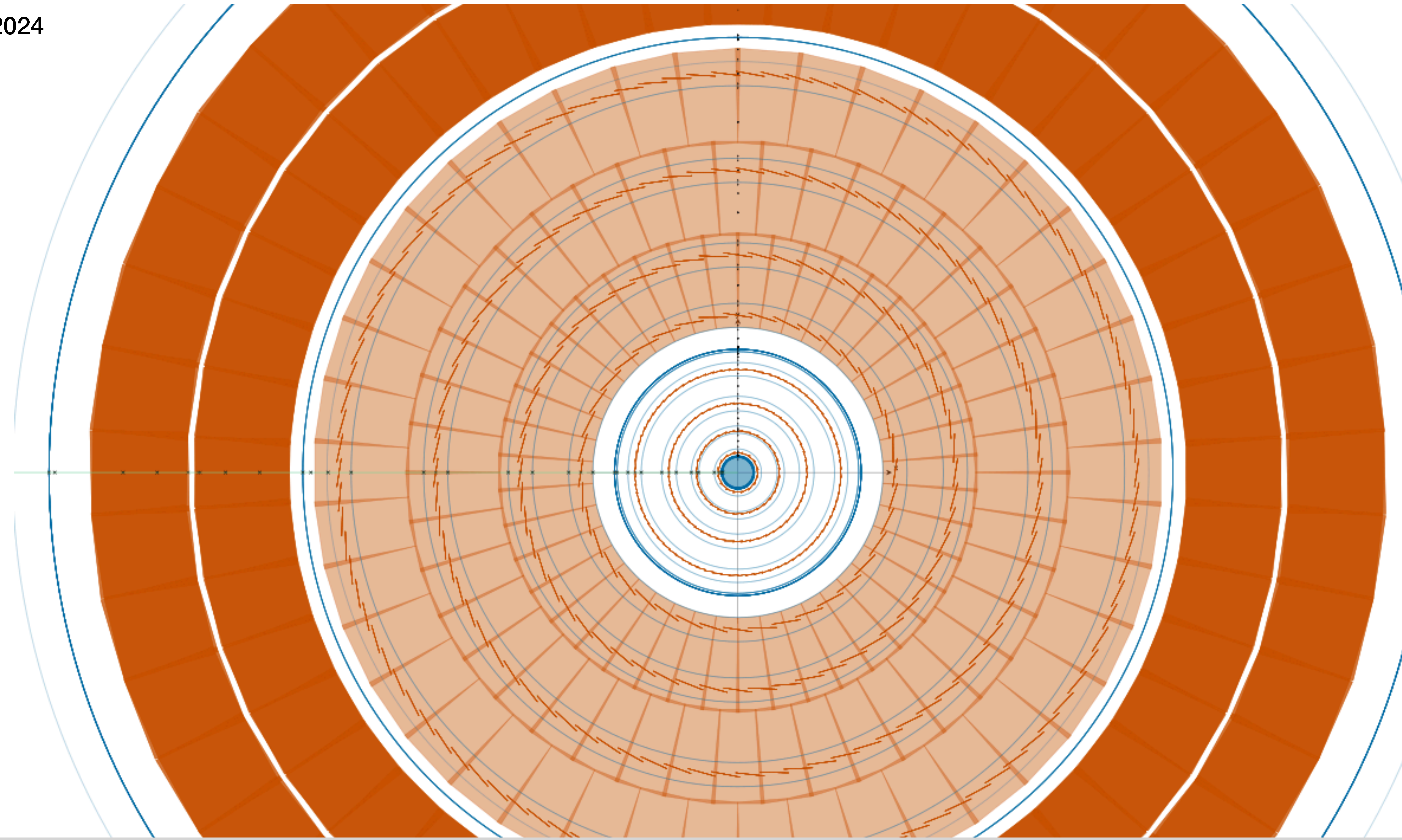
ACTS Workshop, Chateau de Bossey, Nov 19, 2024



ACTS Geometry Evolution

A. Salzburger (CERN)

History

From last workshop, 2023

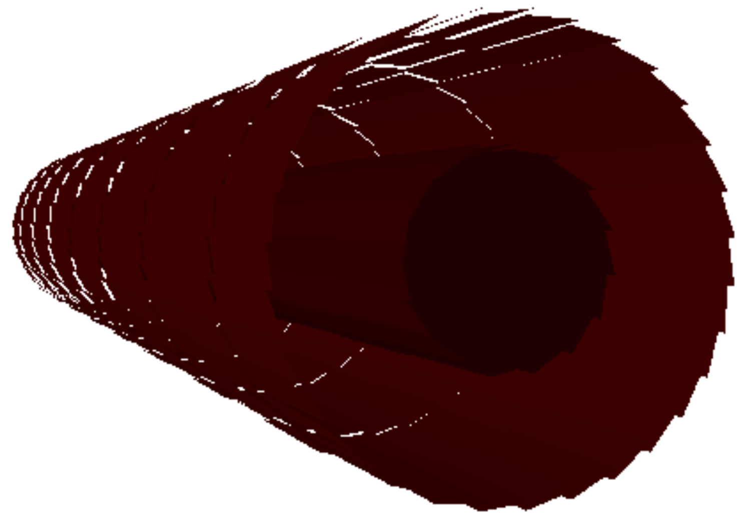
- Geometry model of ACTS stems from ATLAS `Trk::TrackingGeome`
  - Conceptual building blocks

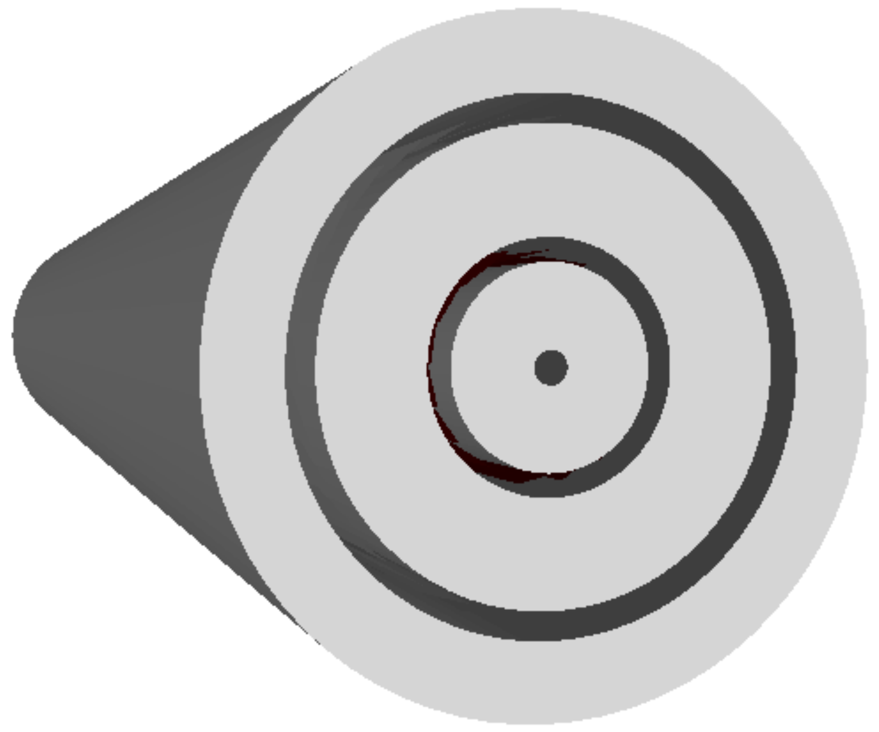
    `TrackingVolume`

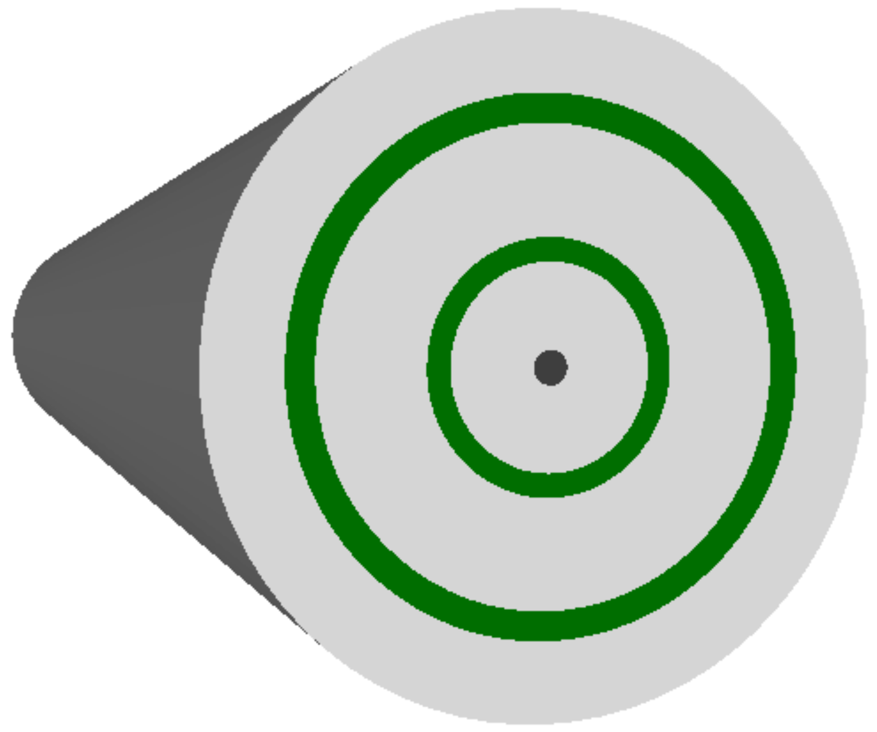
    `Layer`

    `Surface`

Quite some overlap between those







  - `detray` GPU R&D geometry: re-implemented w/o layer concept
    - huge simplification in navigation code
    - can we do this also for ACTS/Core ?

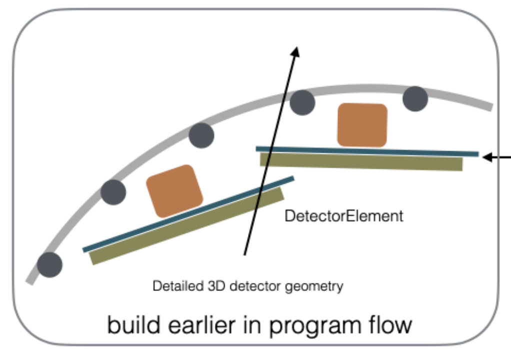
New Acts::Detector model - glossary

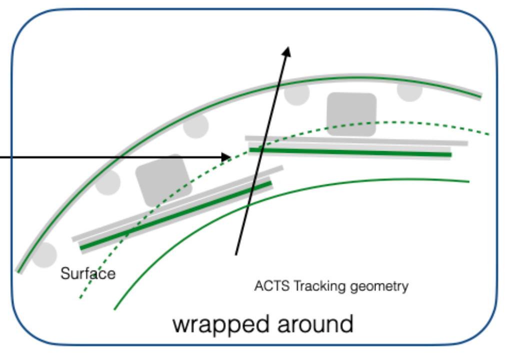
From last workshop, 2023

This geometry is currently still under the Experimental namespace

Acts::Surface	Acts::Surface	Surface objects are unchanged, allows client code to be untouched
Acts::Layer		Layer objects do not exist anymore, they are represented by volumes
Acts::TrackingVolume	Acts::Experimental::DetectorVolume	Double serving of volumes as containers or navigation volumes omitted
Acts::BoundarySurfaceT<Acts::TrackingVolume>	Acts::Experimental::Portal	Portal objects are not templated anymore, they are holder classes of surfaces and volume switches
Acts::TrackingGeometry	Acts::Experimental::Detector	Portal objects the top level entry point that will guide into the root volumes

Acts::Detector building process





Detailed geometry model,
e.g. DD4hep, TGeo, GeoModel, etc.

ACTS geometry model
with built-in navigation

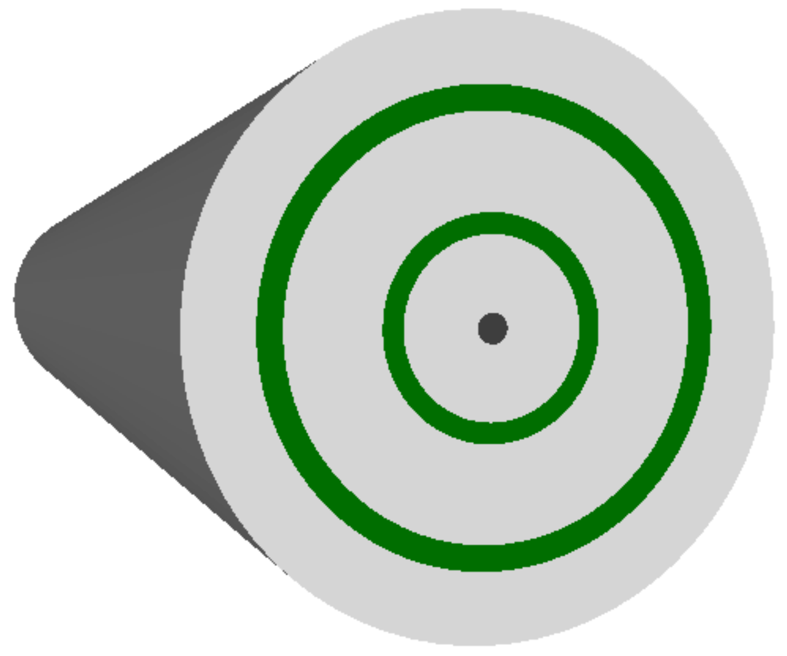
Translation of objects from geometry model,
e.g. DD4hep

from one source, but not necessarily

Logic of how to build/group
e.g. DD4hep

Detector Blueprint

Instruction set how to build the detector



Blueprint to be written to .json

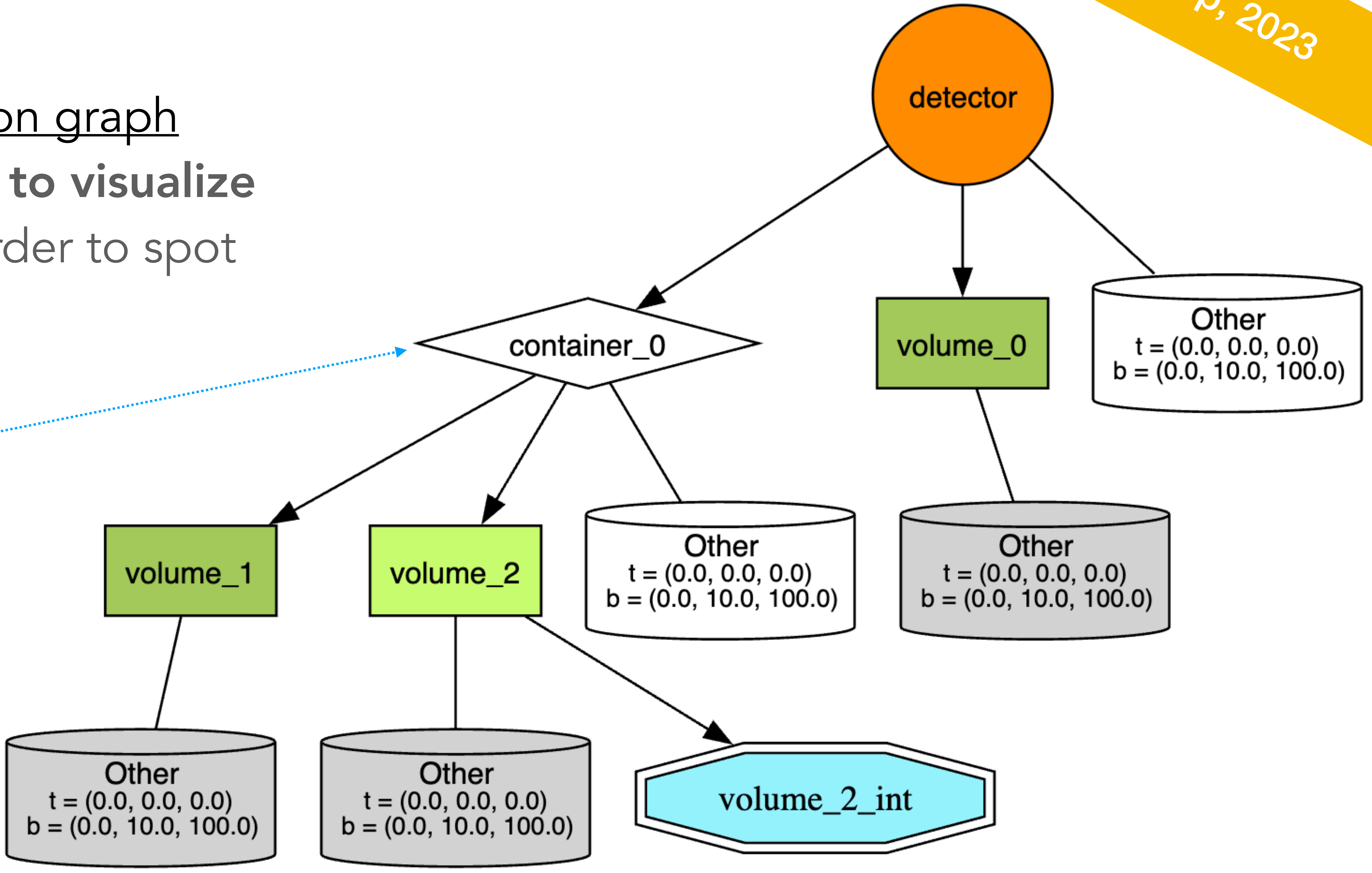
Blueprint visualisation

From last workshop, 2023

Blueprint is an instruction graph

- **Added functionality to visualize** before building, in order to spot problems

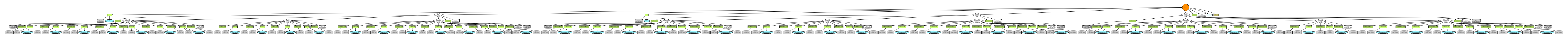
non-coloured nodes are virtual containers



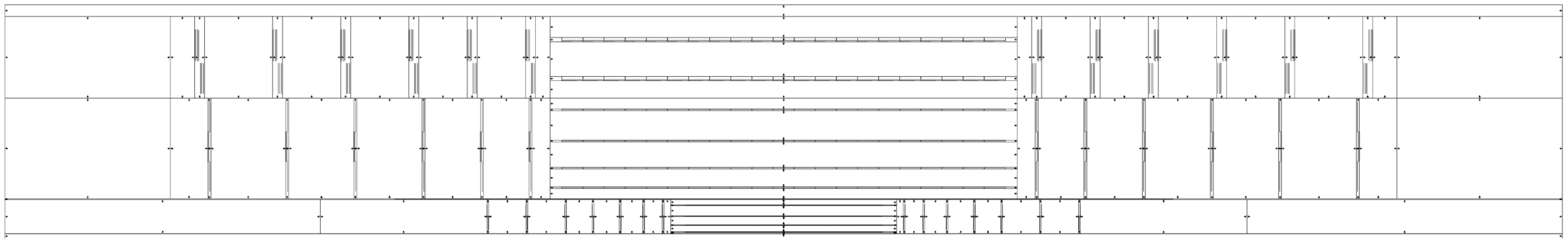
Visualisation

From last workshop, 2023

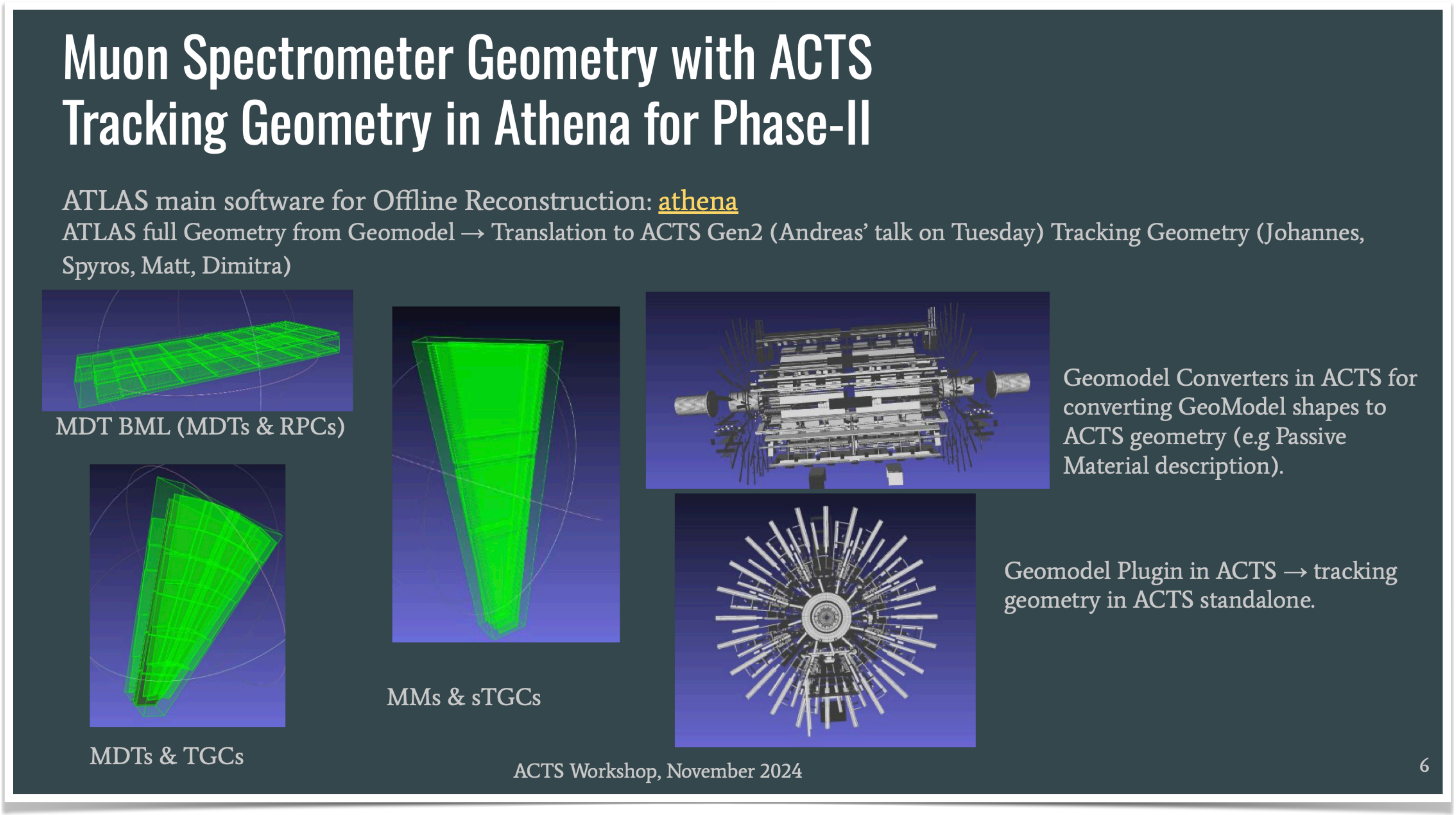
ODD building blueprint from DD4hep:



Resulting ODD detector



In the meantime: **Generation2 MS support**



Muon Spectrometer Geometry with ACTS Tracking Geometry in Athena for Phase-II

ATLAS main software for Offline Reconstruction: athena

ATLAS full Geometry from Geomodel ⟶ Translation to ACTS Gen2 (Andreas' talk on Tuesday) Tracking Geometry (Johannes, Spyros, Matt, Dimitra)

MDT BML (MDTs & RPCs)

MDTs & TGCs

MMs & sTGCs

Geomodel Converters in ACTS for converting GeoModel shapes to ACTS geometry (e.g Passive Material description).

Geomodel Plugin in ACTS ⟶ tracking geometry in ACTS standalone.

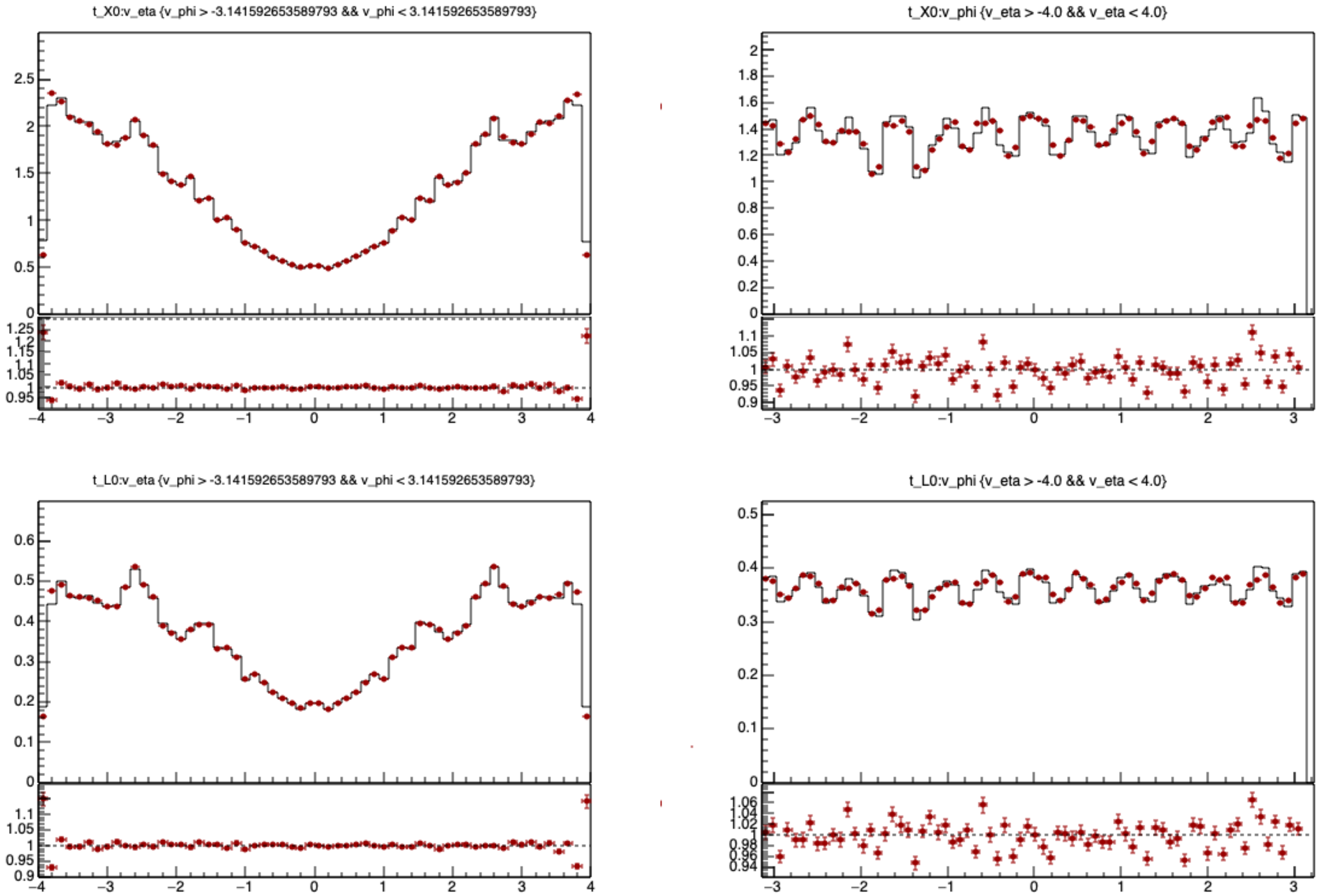
ACTS Workshop, November 2024

6

Dimitra's slides yesterday

In the meantime: **material mapping**

- Geant4
- ● ACTS



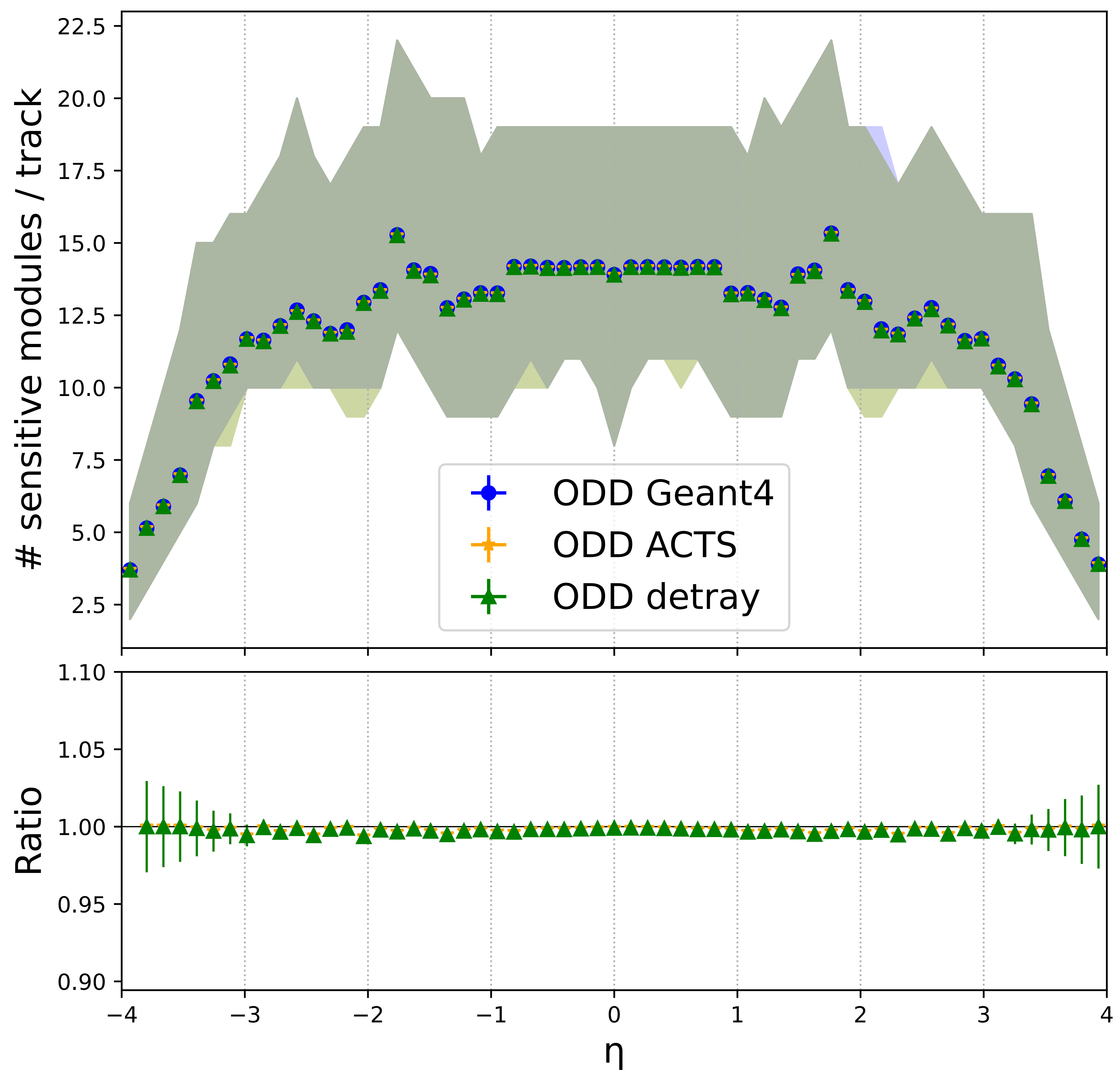
Example material mapping
(Navigation-less mapping)

In the meantime: **detray** conversion

A lot of work went into conversion to detray

- First via the **json I/O**
- Then via **direct translation** with **new** `Plugins/Detray`

New validation framework to test navigation Geant4 - ACTS - detray deployed*



* this will work also on Generation3 geometry

In the meantime

This geometry **is still** under the `Experimental` namespace (and will never leave it)

✓ works without layers
✓ has simpler navigation
✓ Demonstrated for ODD & telescope like detector
✓ Supported MS like geometries
✓ Material mapping works
✓ Translates into detray

So ….. why Generation3?

In the meantime

This geometry **is still** under the `Experimental` namespace (and will never leave it)

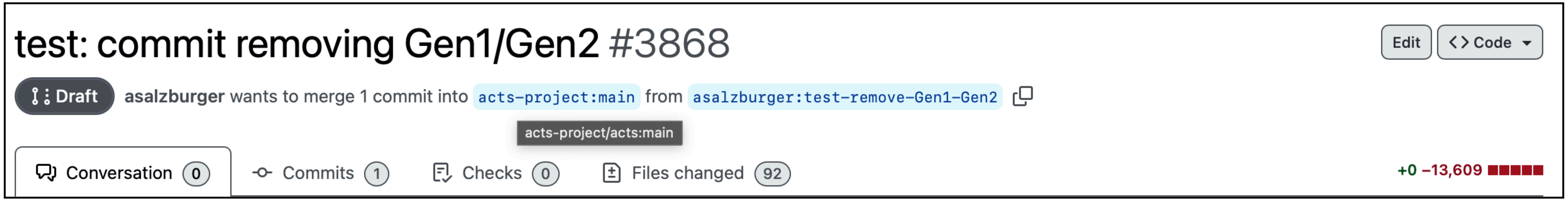
✓ works without layers … cool
✓ has simpler navigation … cool
✓ Demonstrated for ODD & telescope like detector … cool
✓ Supported MS like geometries … cool
✓ Material mapping works … cool
✓ Translates into detray … but is quite complicated and brittle
- Building infrastructure is still to complicated (with many layers)
- Renames/changes the entire geometry setup

Hence, Generation3

Generation3 Geometry

Moving to Generation3 and removing Generation1/2 code will result in a big code cleanup

✓ this is a conservative estimate (navigator code unchanged, plugins not removed)



test: commit removing Gen1/Gen2 #3868

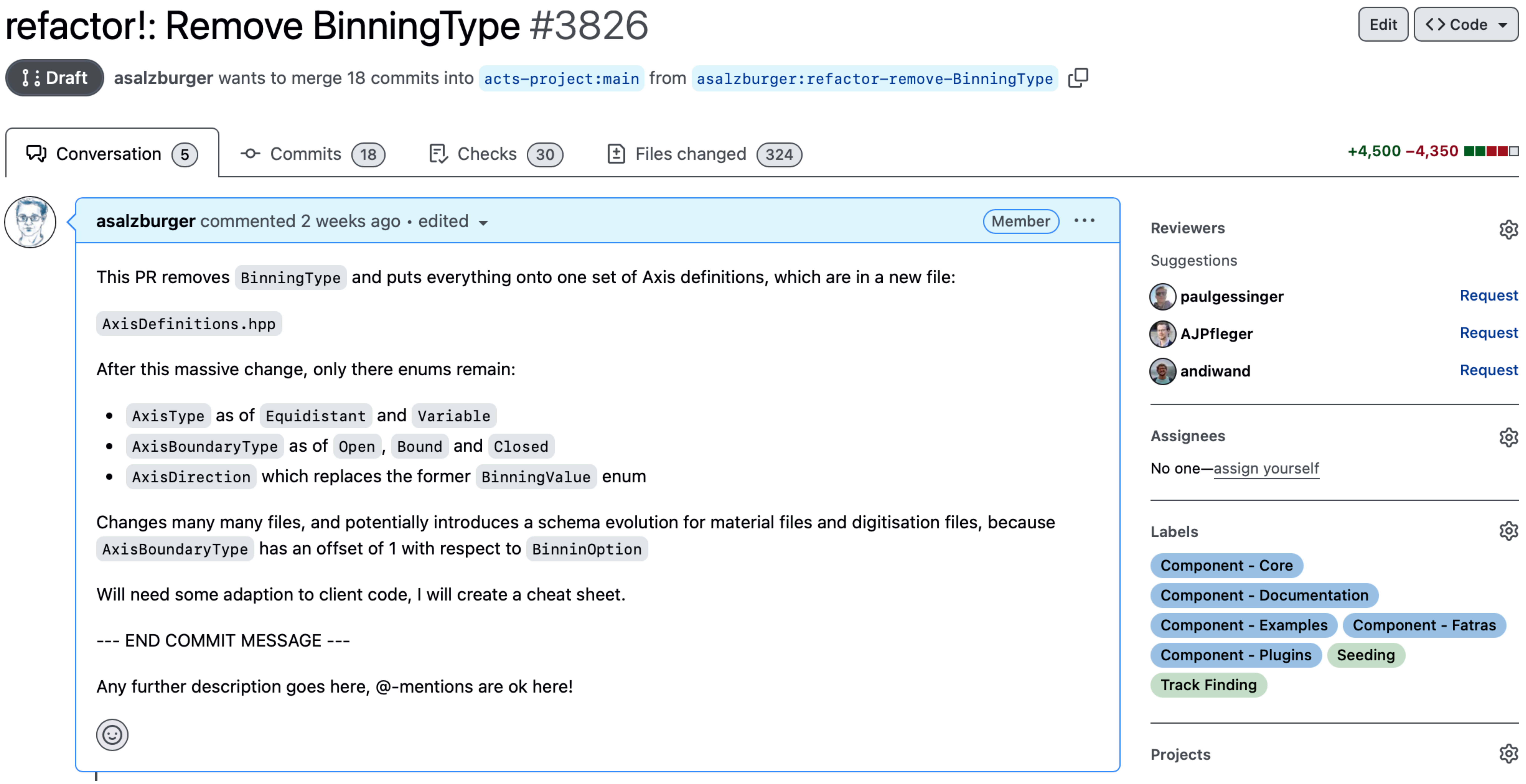
Draft asalzburger wants to merge 1 commit into `acts-project:main` from `asalzburger:test-remove-Gen1-Gen2`

acts-project/acts:main

✓ Generation1 code has shown very high code complexity, should also improve here

More cleanups

BinUtility vs. Axis vs. ProtoBinning



refactor!: Remove BinningType #3826

Edit | Code

Draft asalzburger wants to merge 18 commits into `acts-project:main` from `asalzburger:refactor-remove-BinningType`

Conversation 5 | Commits 18 | Checks 30 | Files changed 324 | +4,500 −4,350

asalzburger commented 2 weeks ago • edited · Member

This PR removes `BinningType` and puts everything onto one set of Axis definitions, which are in a new file:

`AxisDefinitions.hpp`

After this massive change, only there enums remain:

- `AxisType` as of `Equidistant` and `Variable`
- `AxisBoundaryType` as of `Open`, `Bound` and `Closed`
- `AxisDirection` which replaces the former `BinningValue` enum

Changes many many files, and potentially introduces a schema evolution for material files and digitisation files, because `AxisBoundaryType` has an offset of 1 with respect to `BinninOption`

Will need some adaption to client code, I will create a cheat sheet.

--- END COMMIT MESSAGE ---

Any further description goes here, @-mentions are ok here!

Reviewers

Suggestions

paulgessinger Request

AJPfleger Request

andiwand Request

Assignees

No one—assign yourself

Labels

Component - Core

Component - Documentation

Component - Examples

Component - Fatras

Component - Plugins

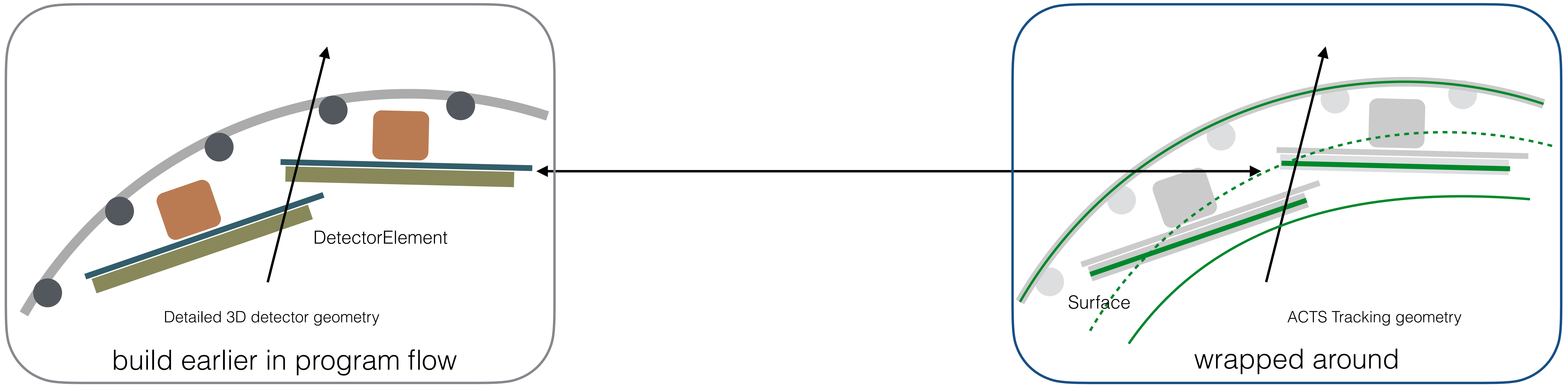
Seeding

Track Finding

Projects

`transform()`: copy vs. const reference

From last workshop, 2023

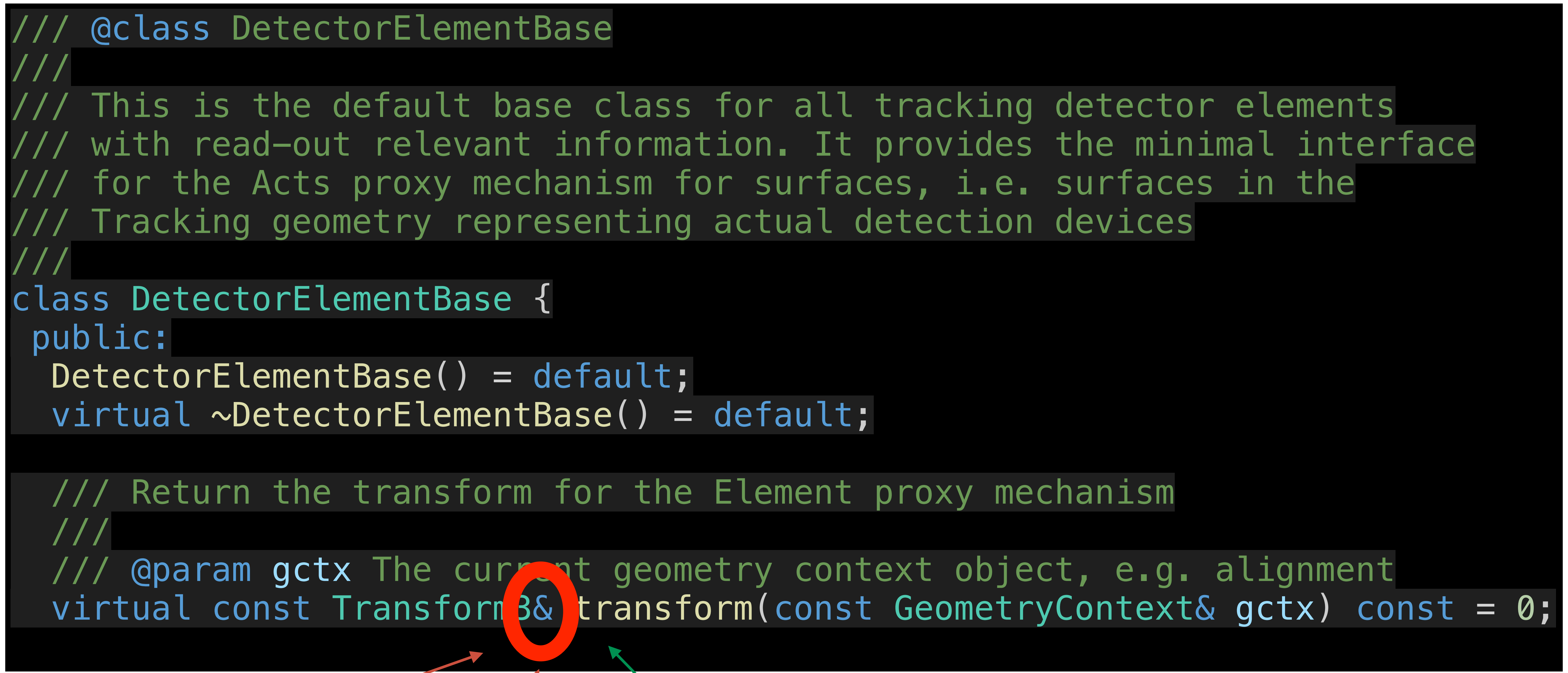


Detailed geometry model,
e.g. DD4hep, TGeo, GeoModel, etc.

ACTS geometry model
with built-in navigation

transform(): copy vs. const reference

From last workshop, 2023



```
/// @class DetectorElementBase
///
/// This is the default base class for all tracking detector elements
/// with read-out relevant information. It provides the minimal interface
/// for the Acts proxy mechanism for surfaces, i.e. surfaces in the
/// Tracking geometry representing actual detection devices
///
class DetectorElementBase {
 public:
  DetectorElementBase() = default;
  virtual ~DetectorElementBase() = default;

  /// Return the transform for the Element proxy mechanism
  ///
  /// @param gctx The current geometry context object, e.g. alignment
  virtual const Transform3D& transform(const GeometryContext& gctx) const = 0;
```

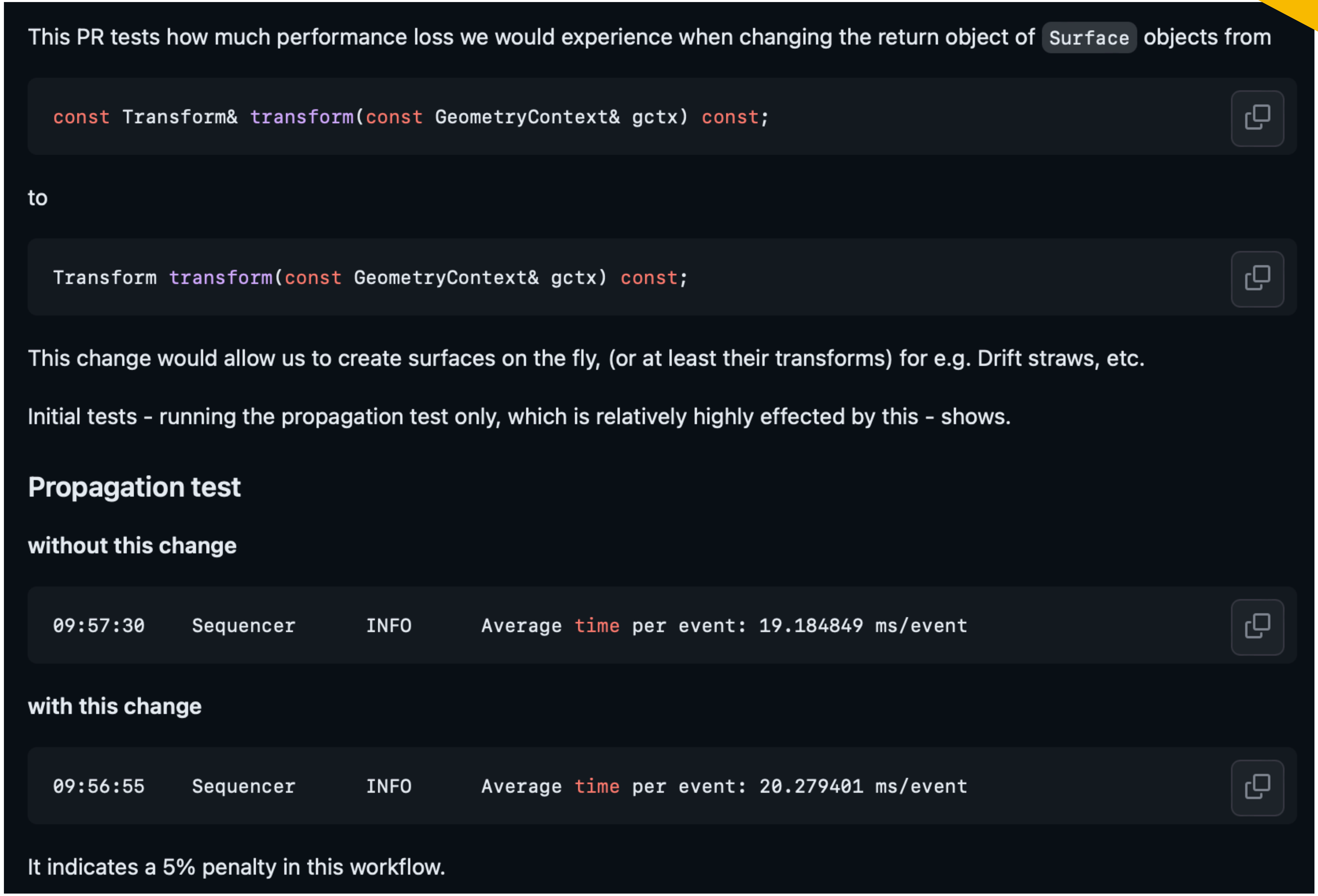
Alignment:
aligned position
can not be generated
on the fly

Wire chambers:
surfaces need to
be precomputed

Preformance
No copy/allocation
needed

transform(): copy vs. const reference

From last workshop, 2023


This PR tests how much performance loss we would experience when changing the return object of `Surface` objects from

```
const Transform& transform(const GeometryContext& gctx) const;
```

to

```
Transform transform(const GeometryContext& gctx) const;
```

This change would allow us to create surfaces on the fly, (or at least their transforms) for e.g. Drift straws, etc.

Initial tests - running the propagation test only, which is relatively highly effected by this - shows.

Propagation test

without this change

```
09:57:30    Sequencer    INFO     Average time per event: 19.184849 ms/event
```

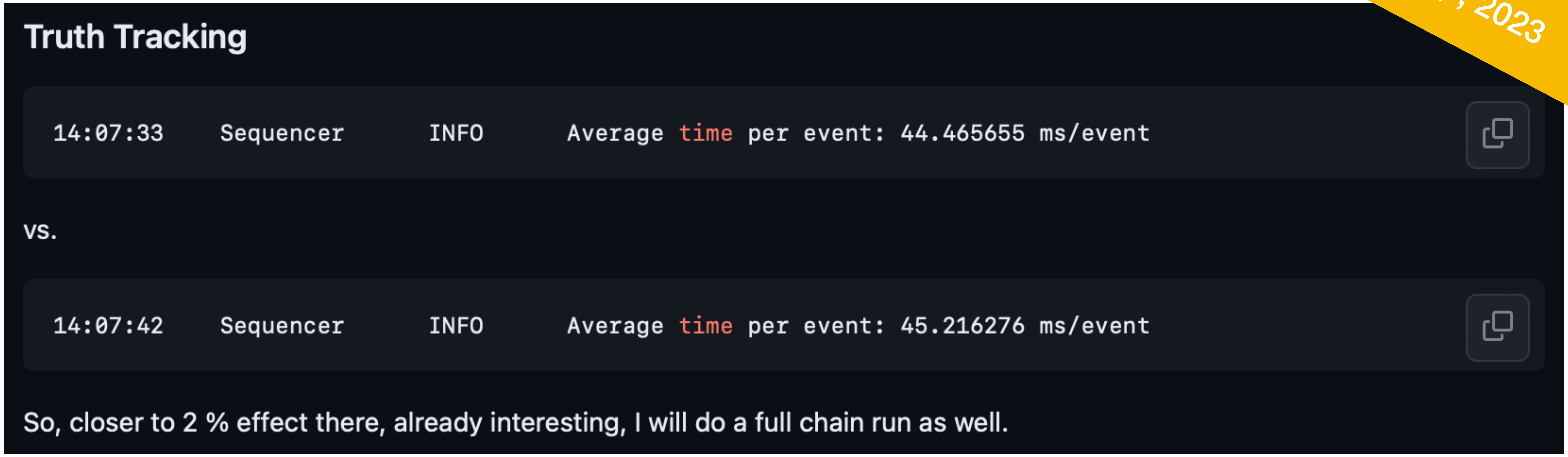
with this change

```
09:56:55    Sequencer    INFO     Average time per event: 20.279401 ms/event
```

It indicates a 5% penalty in this workflow.

transform(): copy vs. const reference

From last workshop, 2023



Truth Tracking

```
14:07:33     Sequencer     INFO     Average time per event: 44.465655 ms/event
```

vs.

```
14:07:42     Sequencer     INFO     Average time per event: 45.216276 ms/event
```

So, closer to 2 % effect there, already interesting, I will do a full chain run as well.

- Next steps:
  Test with ODD full chain example and then we should decide on it in a future developers meeting.