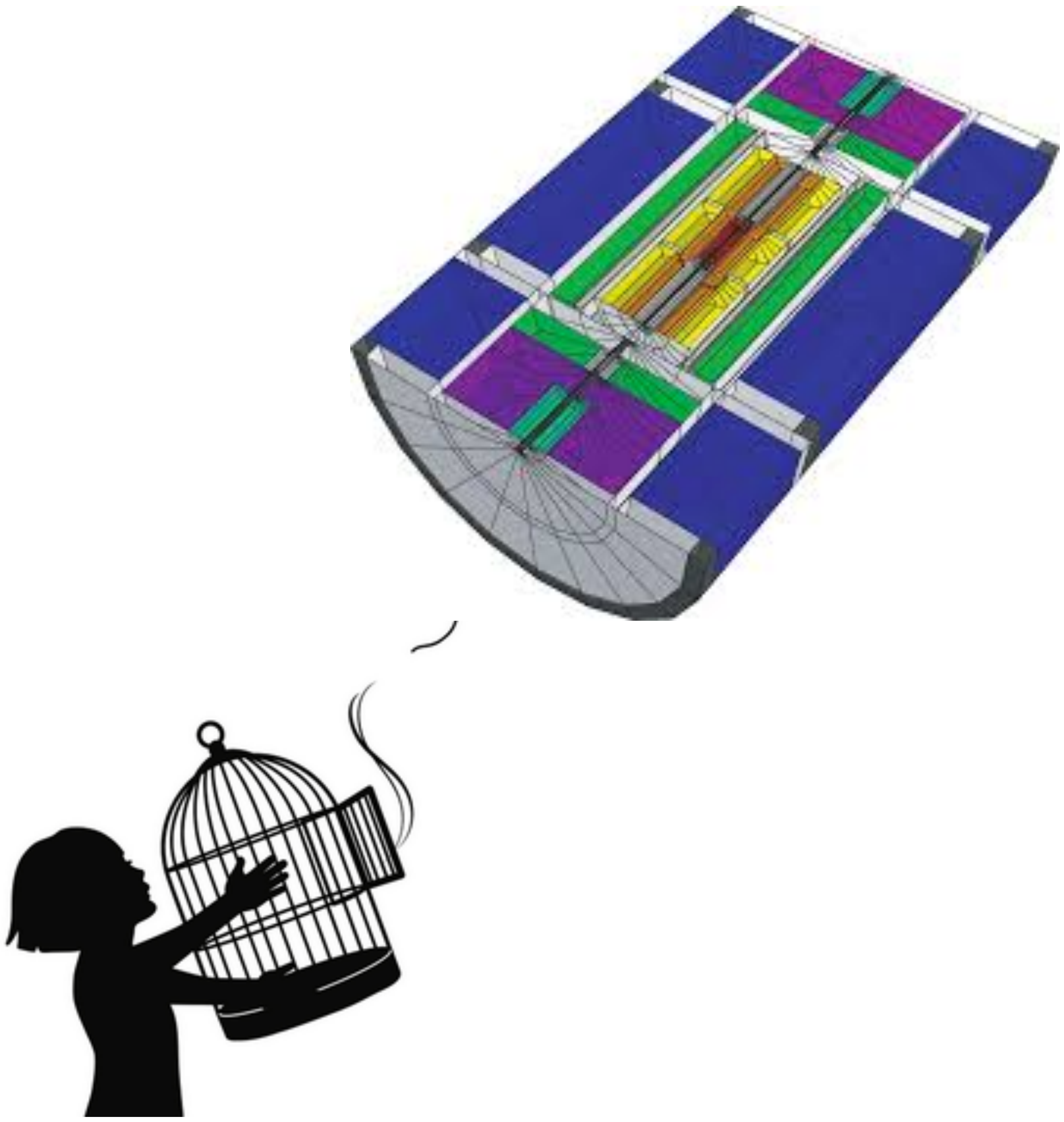
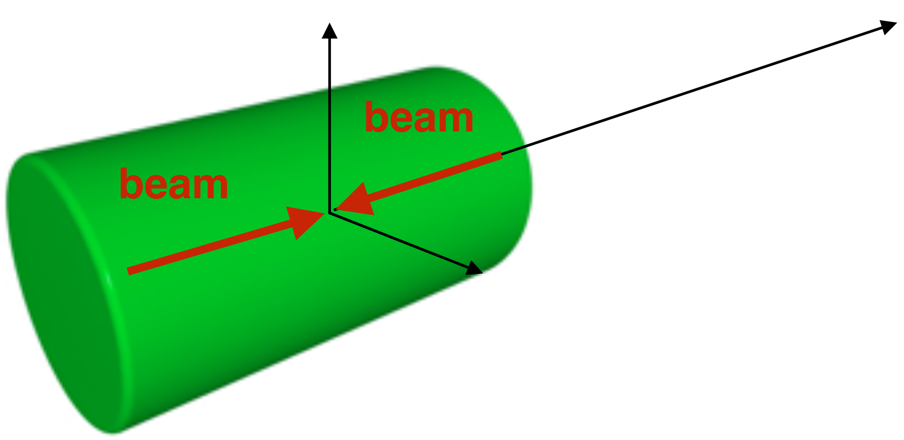


A(TLAS) Tracking Software

A. Salzburger (CERN) - for the ATS developers

Track reconstruction at LHC

- Very similar concepts deployed for ATLAS/CMS
  - ATLAS/CMS in HL-LHC & FCC(-hh) experiments will be similar detector concepts
  - focus on cylindrical hadron-hadron detectors in this talk
- Track reconstruction strategies, algorithms, parameterisations
  - track seeding
  - combinatorial track finding
  - track classification

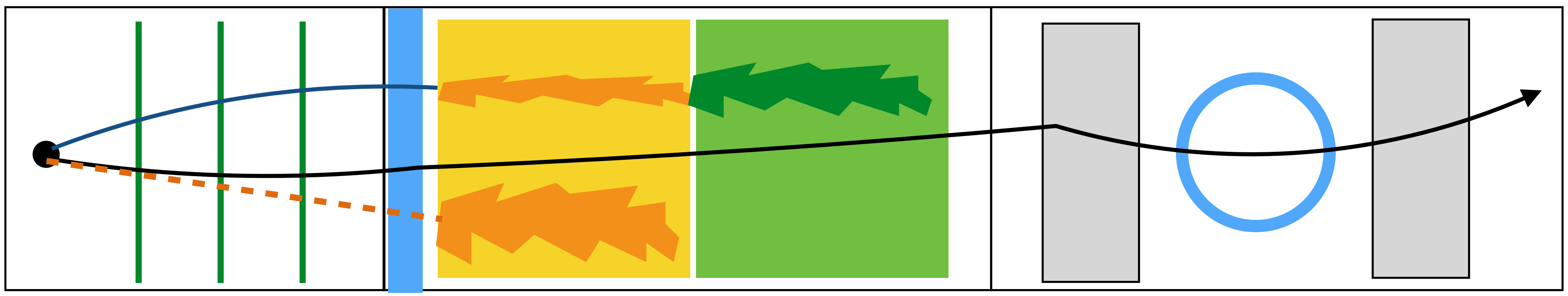


CDF $\mathbf{q}'' = (l_1, l_2, \phi, \cot(\theta), C)$

CMS $\mathbf{q}' = (l_1, l_2, \phi, \lambda, q/p)$

ATLAS $\mathbf{q} = (l_1, l_2, \phi, \theta, q/p)$

A simplified view of ATLAS



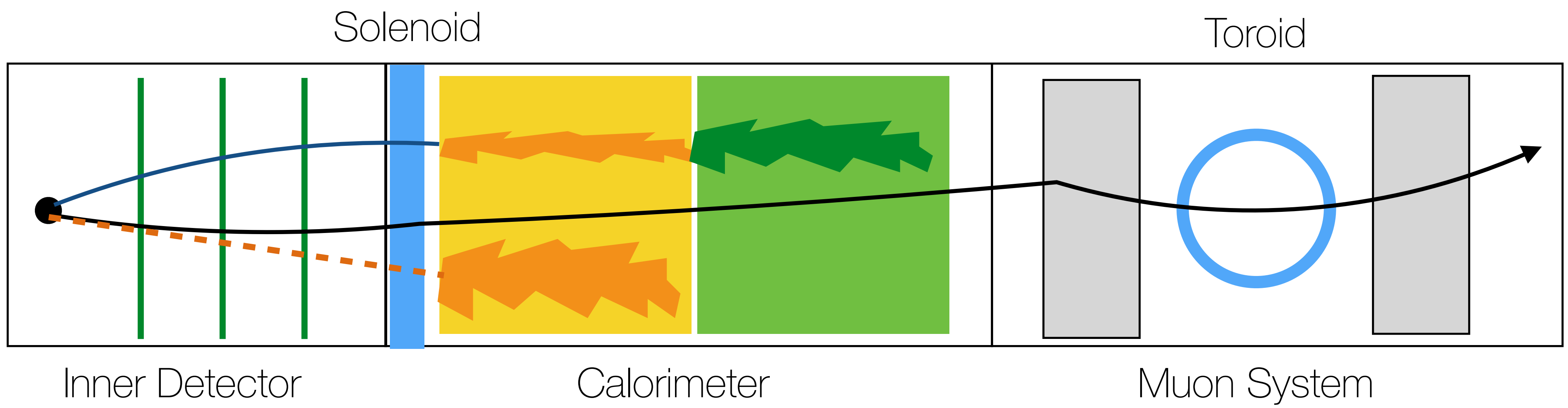
- A simplified "Tracker" view of ATLAS
  - two precision tracking systems having

    very different magnetic field setups

    very different detecting technologies

    very different dimensions
  - some lump of material in between

Current structure - ATLAS repository



Tracking

Geometry (TrkDetDescr)	Event Data Model (TrkEvent)	A Extrapolation (TrkExtrapolation)	A Fitting (TrkFitter)	A Calibration, general (TrkTools)	A Alignment (TrkAlignment)

InnerDetector

EDM (InDetRecEvent)	A Geometry (InDet TrackingGeometry)	A Tools (InDetRecTools)

Calorimeter

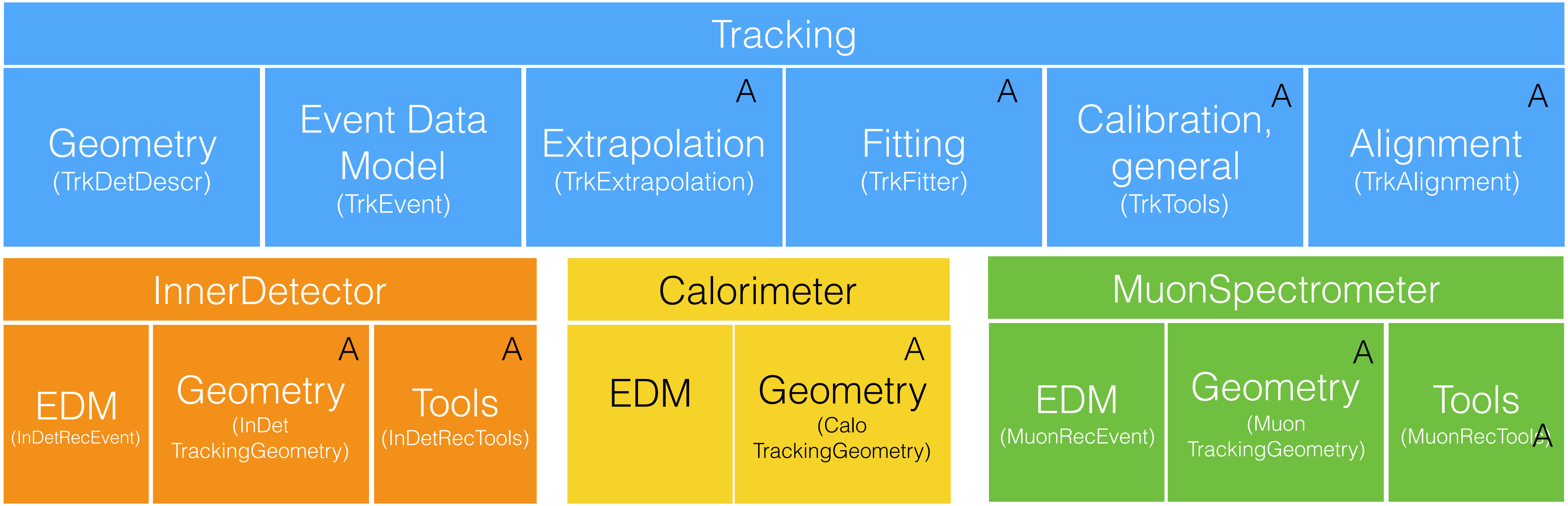
EDM	A Geometry (Calo TrackingGeometry)

MuonSpectrometer

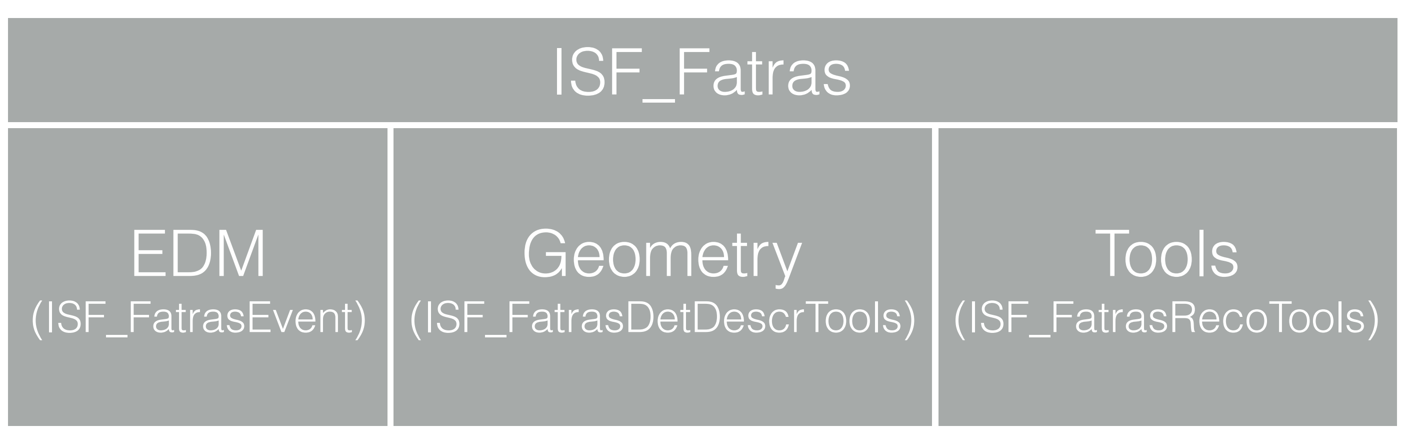
EDM (MuonRecEvent)	A Geometry (Muon TrackingGeometry)	A Tools (MuonRecTools)

A … embedded in Gaudi/Athena structure with AlgTools/Algorithms/Services

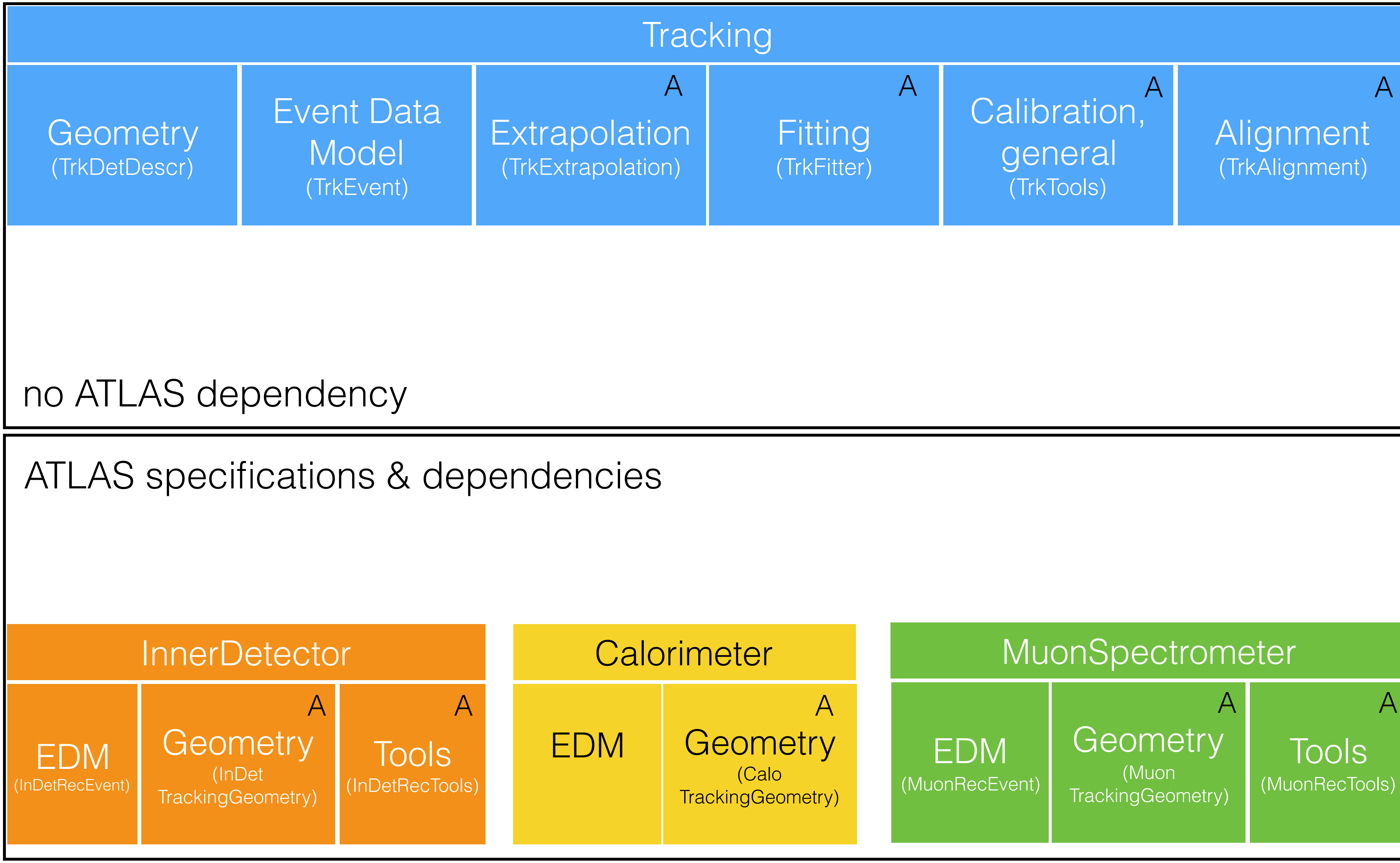
Extending the ATLAS Tracking SW structure



- Within Phase-2 upgrade we developed a fast detector prototyping
  - extended ATLAS tracking EDM/geometry with generic XML based builders
  - could easily run fast track simulation and refitting without actually building ATLAS (2014/15 in parallel a test study within FCC software context & DD4Hep binding)



Decoupling the ATLAS Tracking SW ?



ACTS - Why ?

- LHC detector software has really been stress-tested
  - and I think we learned a lot, and we start working on Upgrade/FCC
  - however, our concepts are sometime > 30 years old !

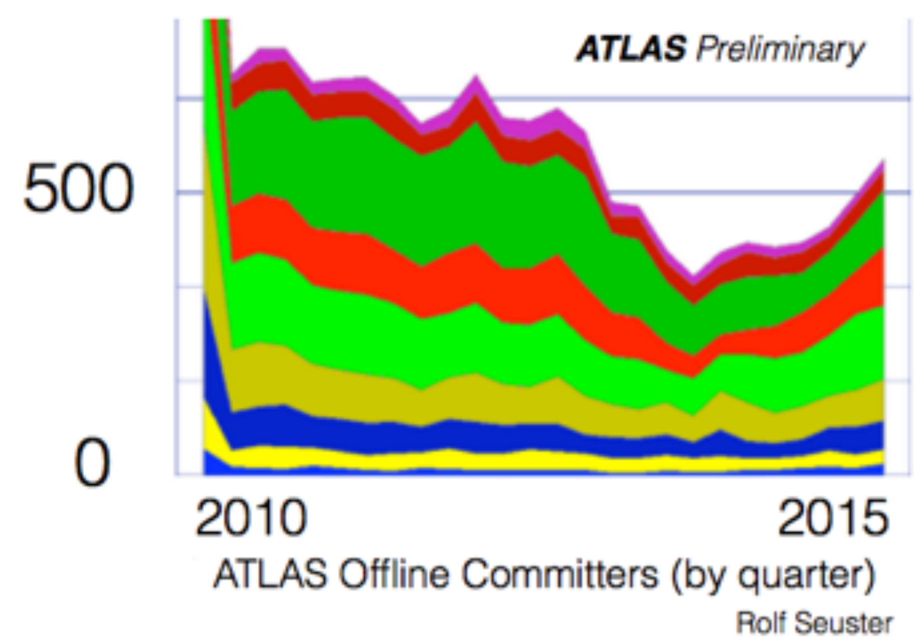
APPLICATION OF KALMAN FILTERING TO TRACK AND VERTEX FITTING

R. FRÜHWIRTH
Institut für Hochenergiephysik der Österreichischen Akademie der Wissenschaften, Vienna, Austria

Received 30 June 1987

- More importantly even

Algorithmic Code Evolution

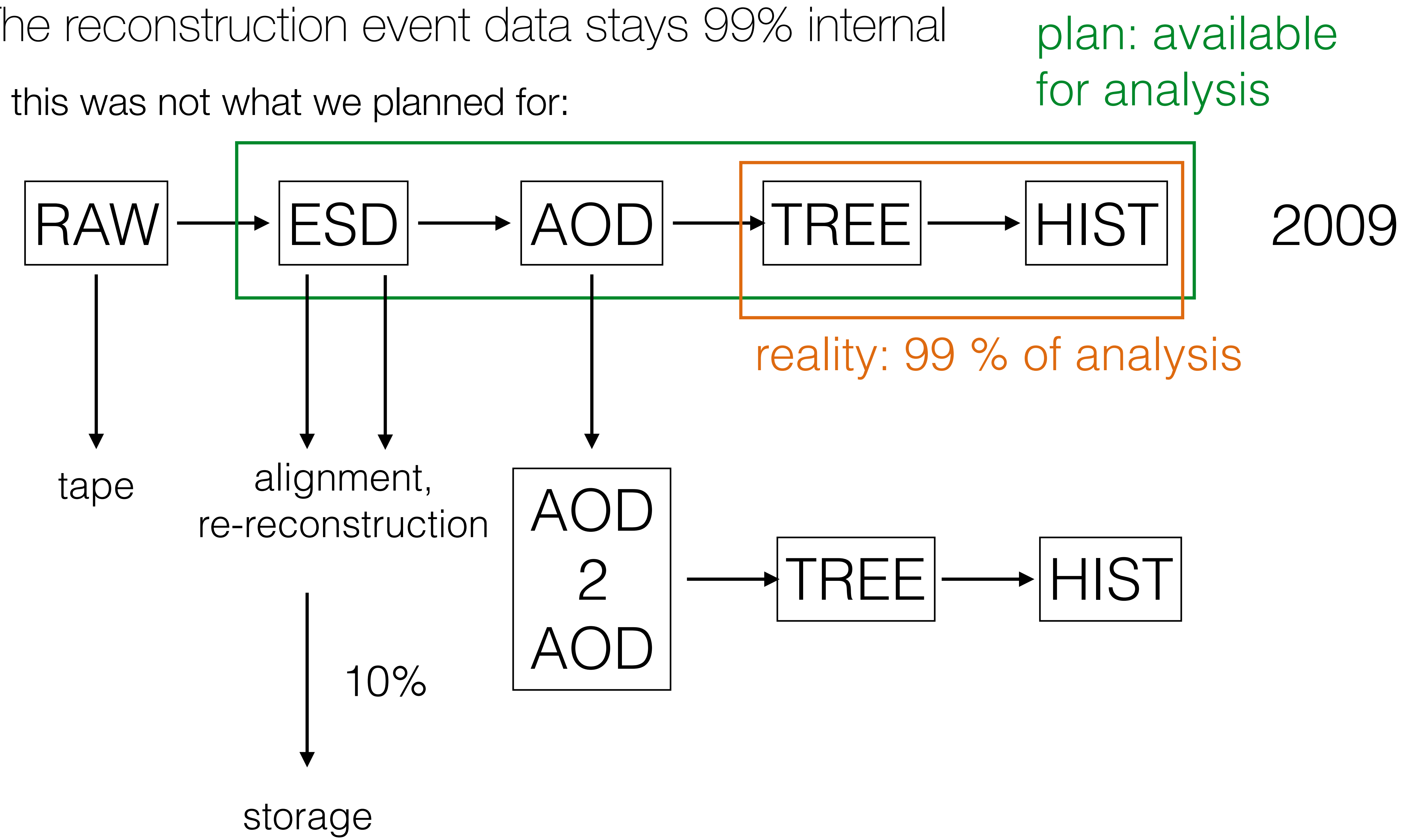


- Highest investment in algorithmic code — O(100M$) for LHC experiments
  - Vast majority of offline packages

[Graeme Stewart, Evolution of HEP SW, CHEP2015]

Software lessons from Run-1 - EDM (1)

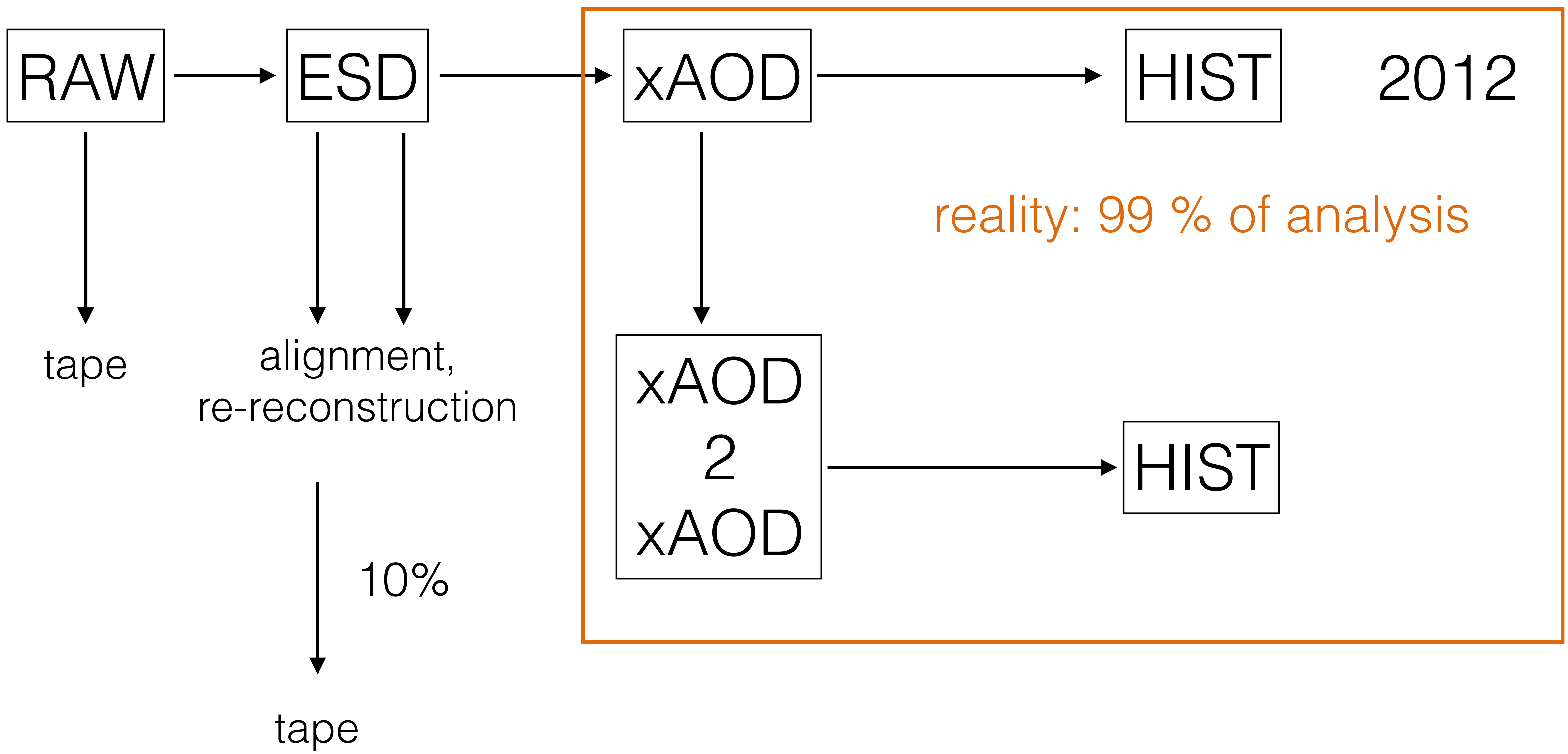
- The reconstruction event data stays 99% internal
  - this was not what we planned for:



this led to an evolution of the ATLAS EDM

Software lessons from Run-1 - EDM (2)

- Introduction of xAOD which is fully ROOT-readable



"life without ESD"

EDM - consequences

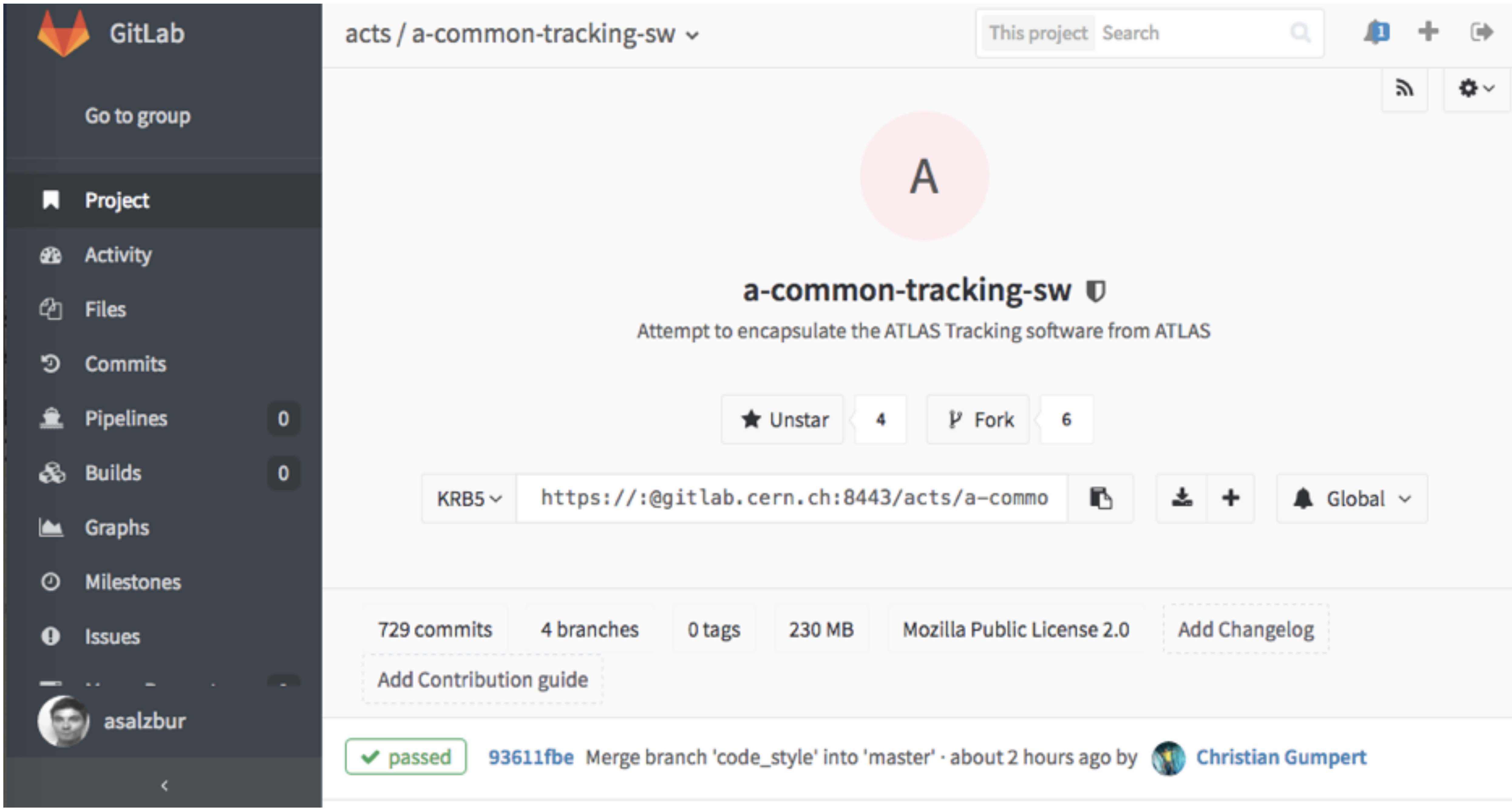
- When designing the Tracking EDM (2003-2005) we were very worried about the 99% non-experts using it, tried to make it
  - as little conventions as possible
  - fully fail & type safe
  - very user friendly
  - complicated inheritance structure

- Did not care about concurrency at that stage

- Obviously, this ended up not in the optimal design
  - lazy initialisation
  - no real thoughts about data locality

<u>Lesson:</u> if we plan for 1% experts only, we can be more aggressive in concepts & design

ACTS - a tracking R&D Testbed



ACTS - some basics

- Minimal dependency to externals
  - ACTS core depends on Eigen*, boost
- Plugin mechanism
  - Geometry is defined by the client
    (e.g. GeoModel, DD4Hep, TGeo, Geant4 can exist as geometry backend)
- C++14 standard
- Workflow totally in GitLab at CERN
  - we did experience the problem of account rights
  - continuous integration using Jenkins

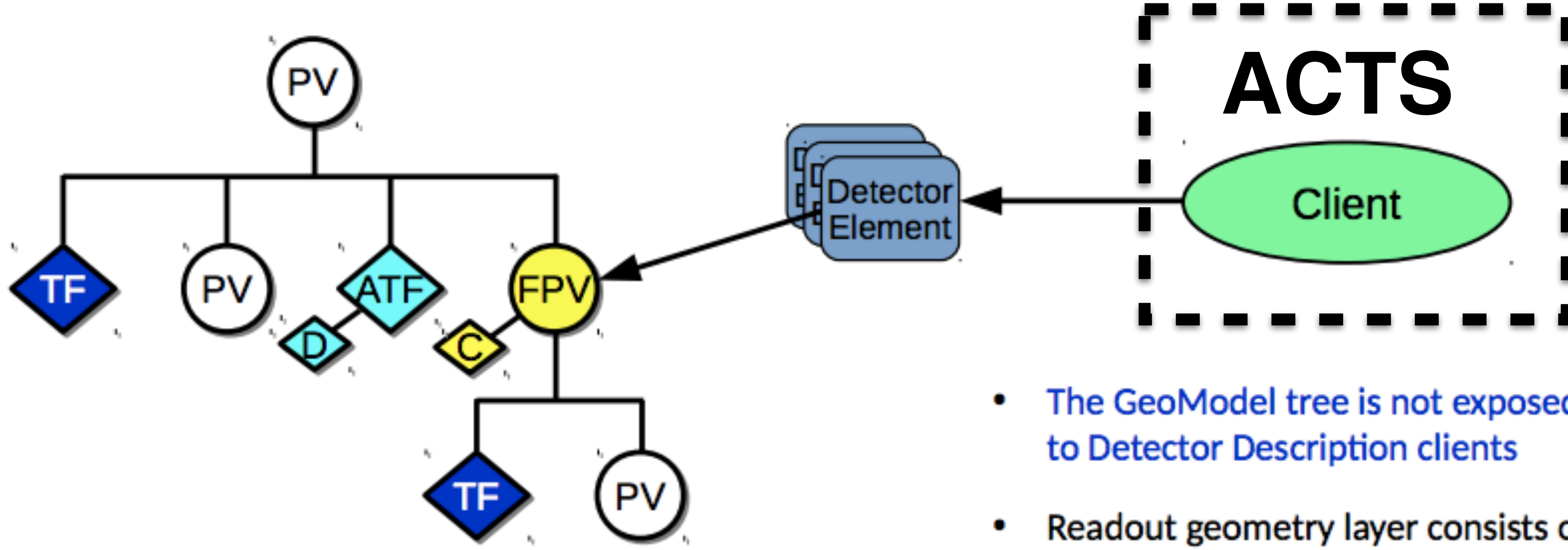
AtlasReadoutGeometryPlugins (->hackatron)

DD4hepPlugins	about 14 hours ago
Geant4Plugins/include/ACTS/Plugin..	a day ago
TGeoPlugins	about 14 hours ago

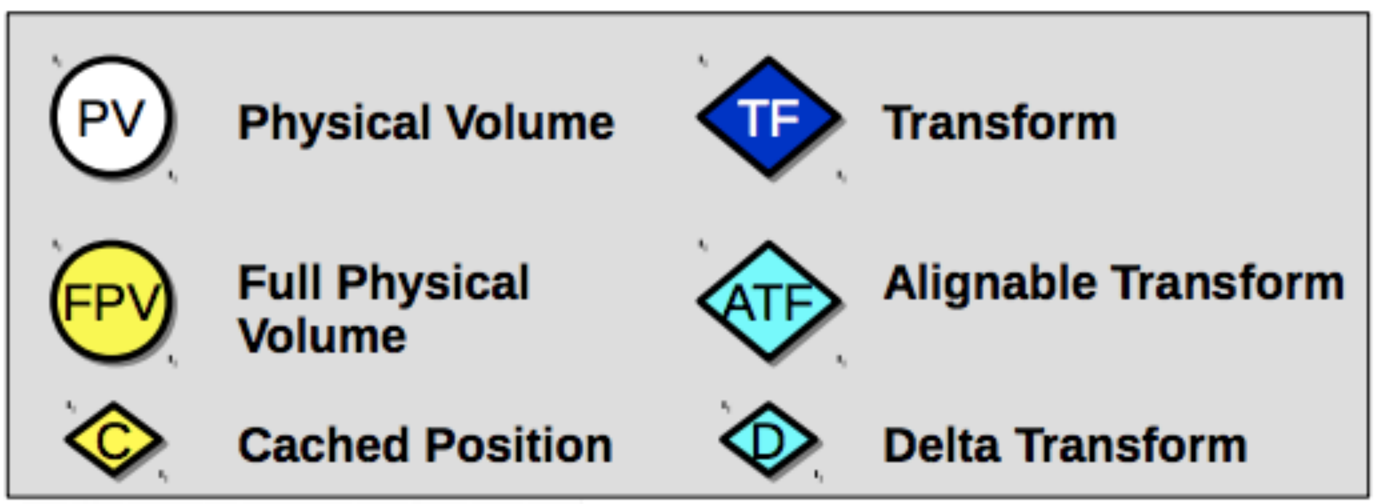
* we could potentially also make an ROOT SMatrix version at some point
(had this discussion of real benchmarking at CHEP last year)

ACTS - Geometry Plugin mechanism

Readout geometry



- The GeoModel tree is not exposed to Detector Description clients
- Readout geometry layer consists of substem specific **Detector Elements**
- Each Detector Element has a **pointer to Full Physical Volume**



U.S. DEPARTMENT OF ENERGY | Office of Science

BERKELEY LAB

Vakho's slide from yesterday

ACTS - Geometry Plugin mechanism

- Example for ROOT TGeo plugin
  - ACTS need detector elements (only sensitive ones) for geometry building
  - needs to extend a DetectorElementBase class and provide an Acts::Surface view

```
class TGeoDetectorElement : public DetectorElementBase
{
public:
  /** Constructor  */
  TGeoDetectorElement(const Identifier&                          identifier,
                      TGeoNode*                                  tGeoDetElement,
                      std::shared_ptr<const Acts::Transform3D> motherTransform
                      = nullptr);
  /** Identifier */
  virtual Identifier identify() const override;

  /**Return local to global transform associated with this identifier*/
  virtual const Transform3D&
  transform(const Identifier& identifier = Identifier()) const override;

  /**Return surface associated with this identifier, which should come from the  */
  virtual const Surface&
  surface(const Identifier& identifier = Identifier()) const override;
```

Tools - Extrapolation

```
/** charged extrapolation - public interface */
ExtrapolationCode extrapolate(ExCellCharged& ecCharged,
                              const Surface* sf = nullptr,
                              const BoundaryCheck& bcheck = true) const final;

/** neutral extrapolation - public interface */
ExtrapolationCode extrapolate(ExCellNeutral& ecNeutral,
                              const Surface* sf = nullptr,
                              const BoundaryCheck& bcheck = true) const final;


/** define for which GeometrySignature this extrapolator is valid - this is GLOBAL */
GeometryType geometryType() const final;

private:
  /** main loop extrapolation method */
  template <class T> ExtrapolationCode extrapolateT(ExtrapolationCell<T>& eCell,
                                                    const Surface* sf = nullptr,
                                                    PropDirection dir=alongMomentum,
                                                    const BoundaryCheck& bcheck = true) const;
```

cache that stays within the thread

Magnetic field cache

`ExCellCharged& ecCharged,`

- Example: magnetic field access
  - numerical (Runge-Kutta) field integration is one of the big CPU consumers
  - ATLAS adaptive Runge-Kutta propagator has been highly optimised

    dedicated version was back-ported into Geant4
  - field access was not yet optimised

    deep caller chain

    field data needed conversion
    was written in FORTRAN90
  - new field service implemented

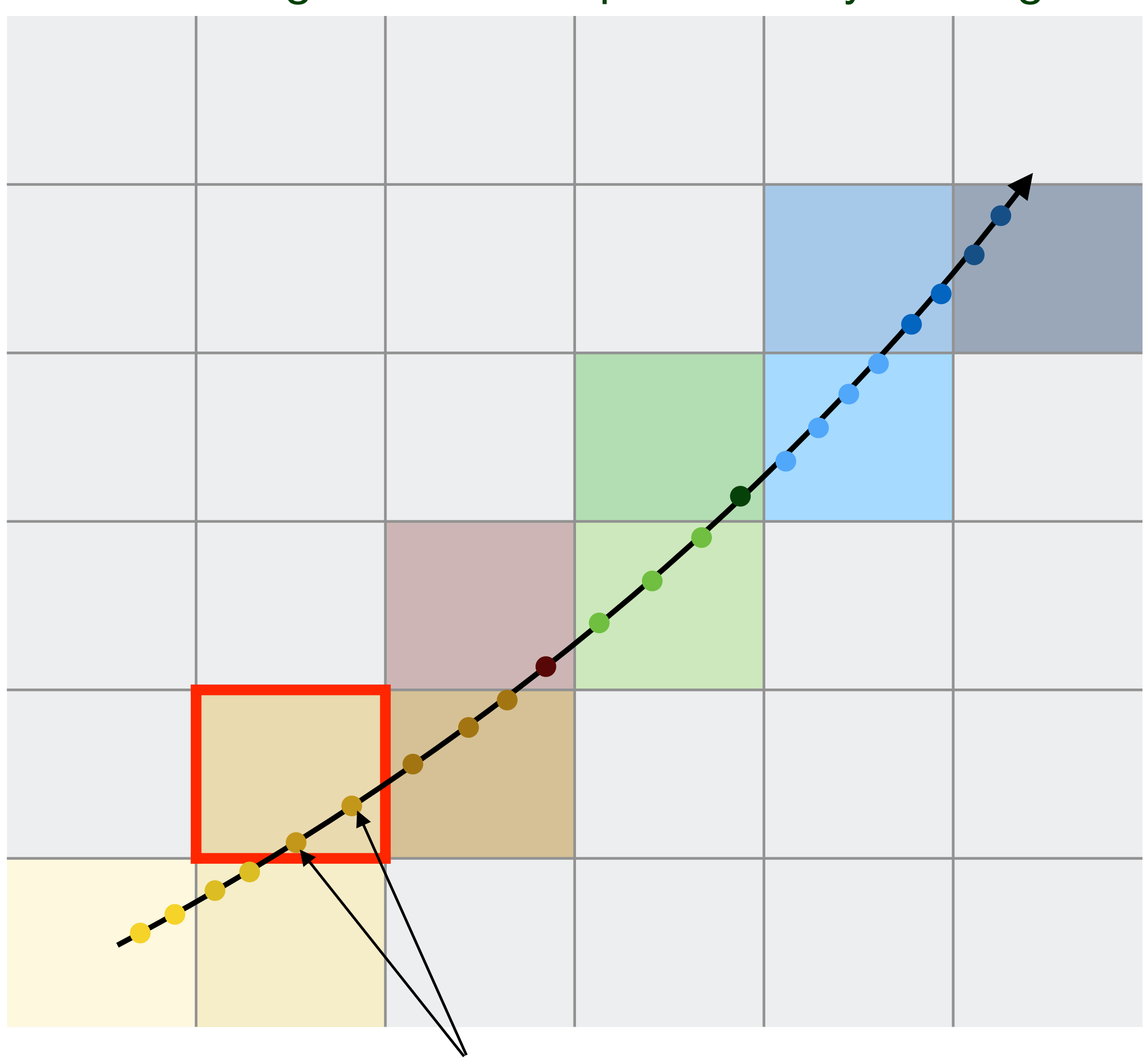
    simplified caller chain

    use native units

    use cell caching to store value of field -> minimised cache misses

    *speed-up of **20%** in simulation, **few %** in reconstruction*

Magnetic field map in memory as 3D grid



Field look up in Runge-Kutta integration

ATS release planning - alpha

0.1.0

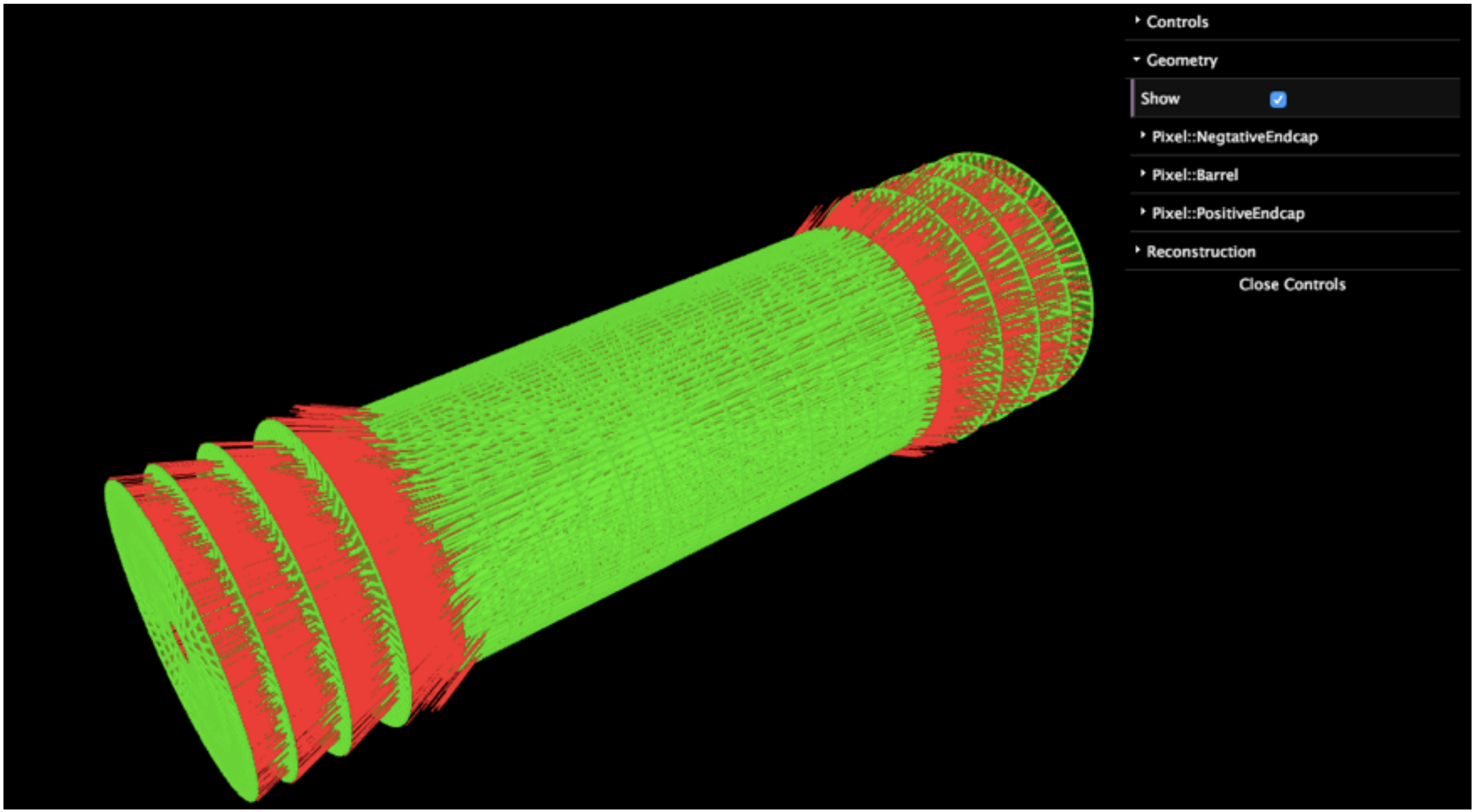
alpha release: first repository build Athena/Gaudi

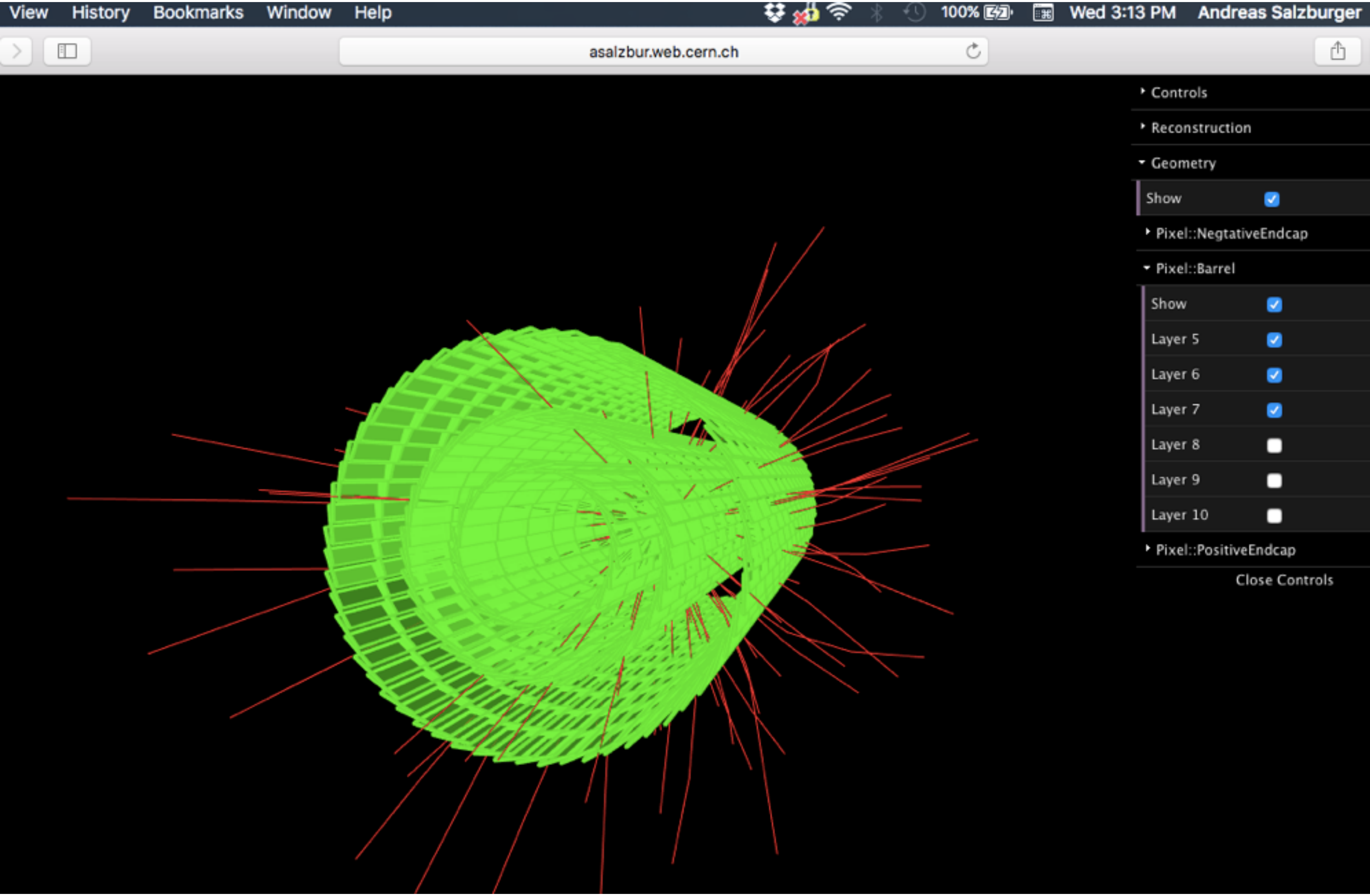
Release date: 19/Feb/16

14 Issues · Release Notes

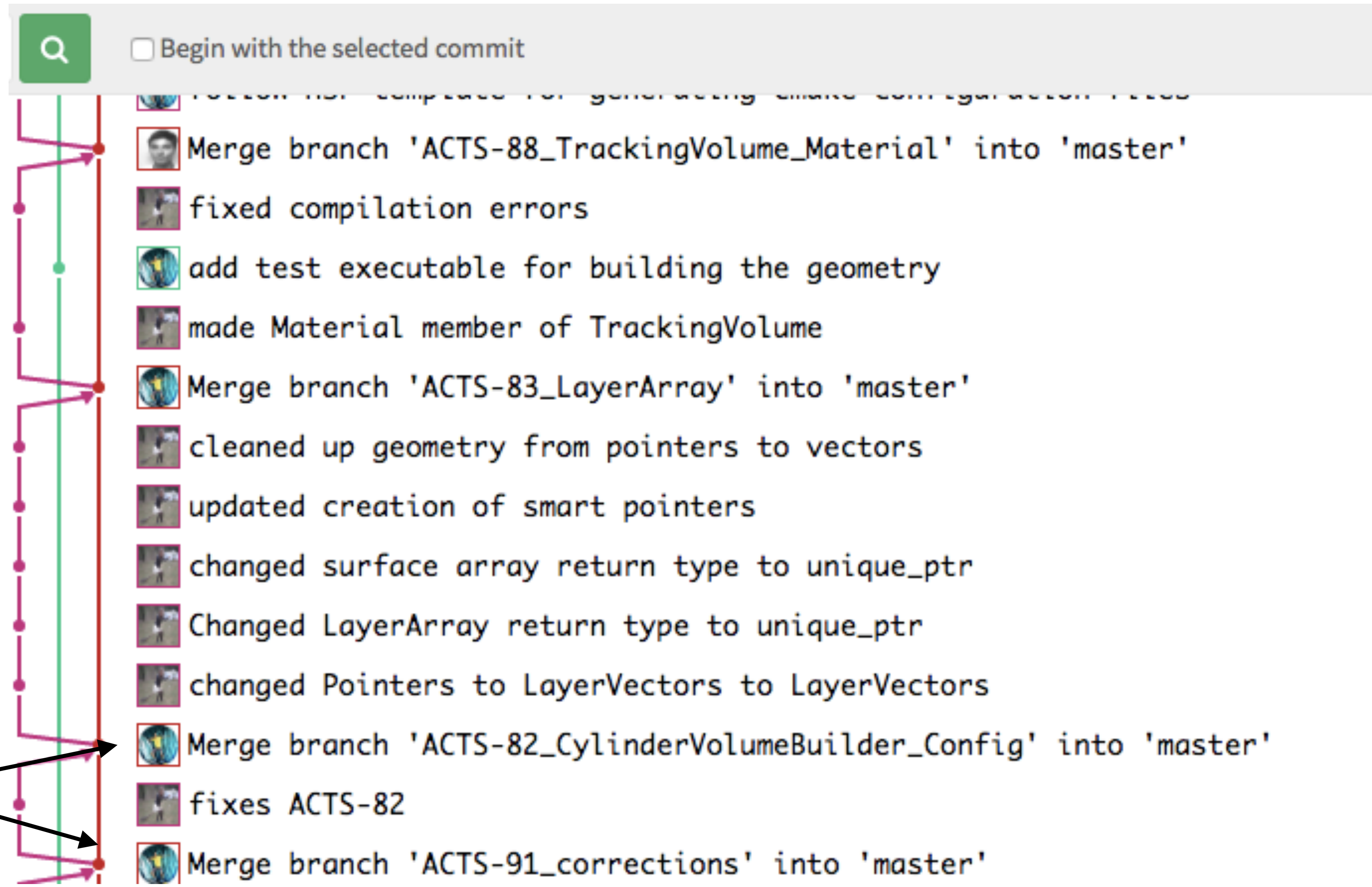
14 of 14 issues have been resolved

WebEventDisplay by Edward Moyse




View History Bookmarks Window Help
100%
Wed 3:13 PM Andreas Salzburger
asalzbur.web.cern.ch
Controls
Reconstruction
Geometry
Show
Pixel::NegtativeEndcap
Pixel::Barrel
Show
Layer 5
Layer 6
Layer 7
Layer 8
Layer 9
Layer 10
Pixel::PositiveEndcap
Close Controls

ACTS - workflow



usually name branches with their JIRA tickets and let gitlab talk to Jenkins & JIRA (see talk by Christian)

Framework de-coupling

```
class MaterialEffectsEngine : virtual public IMaterialEffectsEngine
{
public:
  /** @struct Config
      Configuration struct for the MaterialEffectsEngine
    */
  struct Config
  {
    std::shared_ptr<Logger> logger;
    bool eLossCorrection;  //!< apply the energy loss correction
    bool eLossMpv;  //!< apply the energy loss correction as most probable value
    bool mscCorrection;  //!< apply the multiple (coulomb) scattering correction
    std::string prefix;  //!< screen output prefix
    std::string postfix;  //!< screen output postfix
    std::string name;     //!< the name of this engine

    Config()
      : logger(getDefaultLogger("MaterialEffectsEngine", Logging::INFO))
      , eLossCorrection(true)
      , eLossMpv(true)
      , mscCorrection(true)
      , prefix("[ME] - ")
      , postfix(" - ")
      , name("Anonymous")
    {
    }
  };

  /** Constructor */
  MaterialEffectsEngine(const Config& meConfig);

private:
    Config  m_cfg; //!< configuration object
```

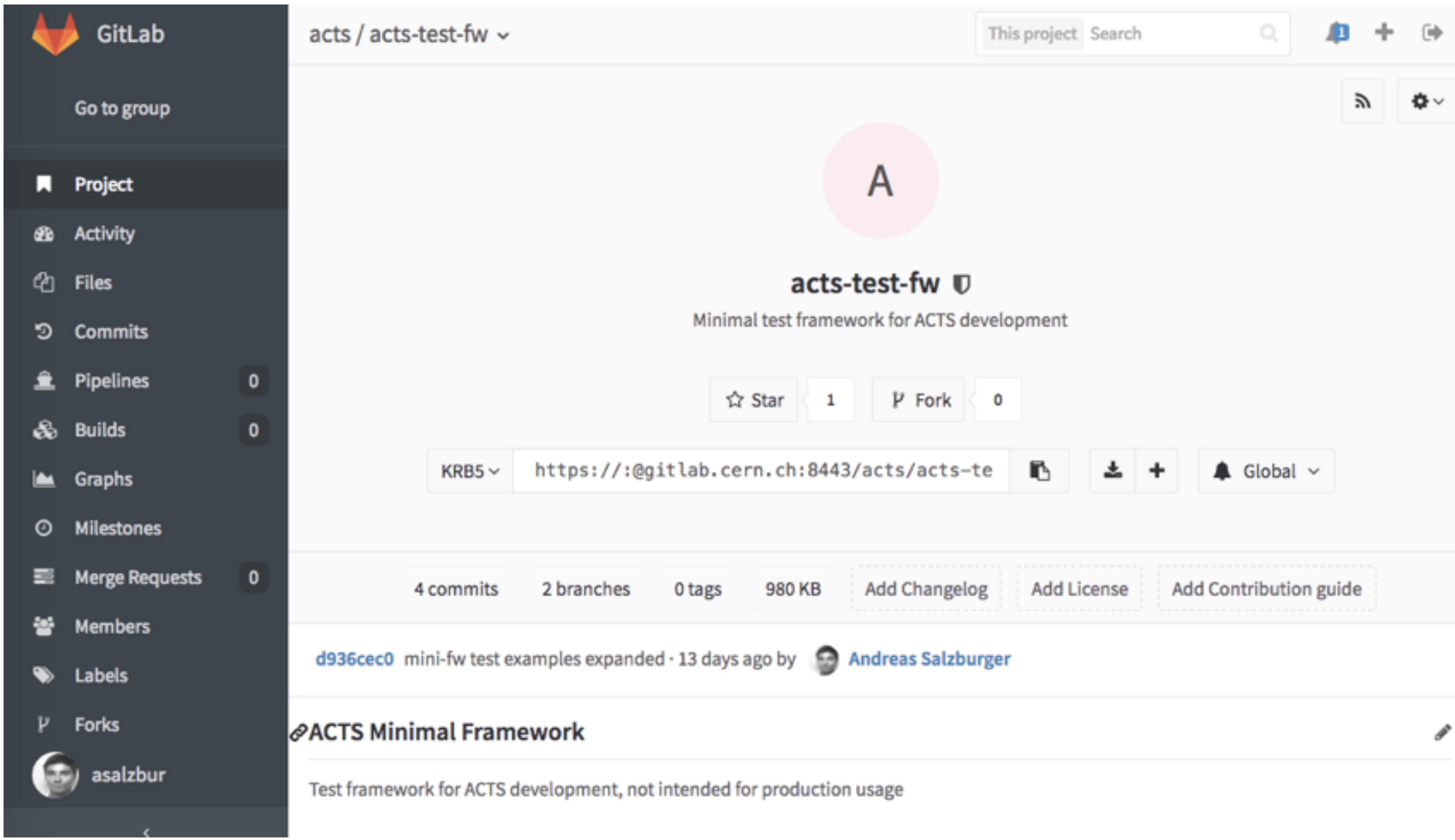
Framework re-coupling

```
typedef <class T> class GaudiMaterialEffectsEngine : public GaudiServiceWrapper<T>, virtual public IMaterialEffectsSvc {

   // the private ACTS material effects engine
   private:
       std::unique_ptr<T>    m_meuEngine;

} ;


  // create the configuration object
  Acts::T::Config meuConfig;
  // use the declareProperty interface
  declareProperty("ApplyEnergyloss", meuConfig.eLossCorrection);

  // now create the internal ACTS engine
  m_meuEngine = std::make_unique<T>(meuConfig);
```

- For the Athena(Gaudi) usage wrappers in the ATLAS(FCC) repository
  - implements defined ATLAS/FCC interfaces
  - uses declareProperty for configuration
  - can work with genConf and keep ATLAS/FCC python jobOptions as is

ACTS - Extensions (1)

- We have a mini test framework for debugging/testing
  - mimics Gaudi (not GaudiHive) behaviour



ACTS - Extensions (2)

- ATLAS fast track simulation modules been put into a separate repository
  - make ACTS usable for Tracking R&D, e.g. Machine Learning challenge