ATLAS Tracking Software: history, status & prospects

A. Salzburger (CERN) on behalf of the ATLAS Tracking SW
The Reality

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
The **FORTRAN** times

### Inner Detector
- Helical track model
- Internal event data model
- From pattern recognition to track fitting within one program

### Calorimeter
- Something nasty you have to deal with

### Muon System
- 1 main track reconstruction program in FORTRAN with a chi2 combination with the Inner Detector reconstruction, no combined fit
The C times / Early ATHENA days

2 monolithic track reconstruction programs in FORTRAN with semicolons at the end
- helical track model
- internal event data model
- from pattern recognition to track fitting within one program

something nasty you have to deal with, 2 independent models to do so

2 main track reconstruction programs, one in FORTRAN, on C, with either a chi2 combination with the Inner Detector reconstruction, or a combined track fit
ATLAS Reconstruction Task Force (RTF) review & recommendations
- final report issued in 2003, recommendations to restructure ATLAS software

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

<table>
<thead>
<tr>
<th>Feature</th>
<th>Solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>very different magnetic field setups</td>
<td>field-agnostic parameterisation</td>
</tr>
<tr>
<td>very different detecting technologies</td>
<td>technology-agnostic high level tracking</td>
</tr>
<tr>
<td>very different dimensions</td>
<td>re-calibration on demand</td>
</tr>
<tr>
<td>some lump of material in between</td>
<td>integration of calorimeter into tracking</td>
</tr>
</tbody>
</table>
Current structure - from reality to repository

- Inner Detector
- Calorimeter
- Muon System

Tracking

<table>
<thead>
<tr>
<th>Geometry (TrkDetDescr)</th>
<th>Event Data Model (TrkEvent)</th>
<th>Extrapolation (TrkExtrapolation)</th>
<th>Fitting (TrkFitter)</th>
<th>Calibration, general (TrkTools)</th>
<th>Alignment (TrkAlignment)</th>
</tr>
</thead>
</table>

InnerDetector

- EDM (InDetRecEvent)
- Geometry (InDetTrackingGeometry)
- Tools (InDetRecTools)

Calorimeter

- EDM (CaloTrackingGeometry)
- Geometry (CaloTrackingGeometry)

MuonSpectrometer

- EDM (MuonRecEvent)
- Geometry (MuonTrackingGeometry)
- Tools (MuonRecTools)

A … embedded in Gaudi/Athena structure with AlgTools/Algorithms/Services
**ATLAS Tracking Geometry - in a sketch (1)**

**Trk::Surface** class acts as a representation of a detector element (proxy mechanism allows to bind it to basically all geometry models)

```cpp
namespace Trk {
    typedef ParametersBase<5, Charged> TrackParameters;
    typedef CurvilinearParametersT<5, Charged, PlaneSurface> CurvilinearParameters;
    typedef ParametersT<5, Charged, ConeSurface> AtaCone;
    typedef ParametersT<5, Charged, CylinderSurface> AtaCylinder;
    typedef ParametersT<5, Charged, DiscSurface> AtaDisc;
    typedef ParametersT<5, Charged, PerigeeSurface> Perigee;
    typedef ParametersT<5, Charged, PlaneSurface> AtaPlane;
    typedef ParametersT<5, Charged, StraightLineSurface> AtaStraightLine;
}
```

for **measurements** and track parameterisation
ATLAS Tracking Geometry - in a sketch (2)

Trk::Surface class acts as a base for Trk::Layer

for grouping objects, e.g. detector elements on layers
Trk::Surface class acts as a (shared) boundaries for Trk::Volumes

full connective geometry, i.e. every boundary surface is attached to the next volume(s)

navigation through geometry comes as an intrinsic feature of the extrapolation process.
Trk::Volumes exist also in a dense flavour (e.g. for calorimeter description)

ATLAS has developed a special propagation engine for propagation through dense material with an instantaneous integration of material interactions

STEP_Propagator

ATL-SOFT-PUB-2008-003
(Transient) Tracking EDM is built upon two base classes describing:

- measurements:
  - **Trk::MeasurementBase**

- parameterisations:
  - **Trk::ParametersBase**

### Structure

- **Segments**
- **Calibrated Measurements**
- **Space Points**

### Columns

- **Measurements**
  - **ID**
  - **MS**

### Parameters

- **Neutral Parameters**
- **Charged Parameters**
- **MultiComponent Charged Parameters**
That’s what the TrackFitter sees

- measurements: `Trk::MeasurementBase`
  - Segments
  - Calibrated Measurements
  - Space Points
  - Measurements
    - ID
    - MS

- parameterisations: `Trk::ParametersBase`
  - Neutral Parameters
  - Charged Parameters
  - MultiComponent Charged Parameters
TrackingGeometry - simplification of nodes

- Layers are described as cylinders and disks
  - sensitive material in CURRENT tracking geometry only on those
  - even though the modules are perfectly known to the TrackingGeometry
Intermezzo 1 - Fast Track Simulation (Fatras)

- Embedded navigation with Extrapolation engine is a fast track simulation
  - simply changing stochastic material effects integration into MC based one
  - using the reconstruction geometry as a simulation geometry is a common concept for fast simulation
Intermezzo 2 - the LS1 campaign

- LHC reconstruction software is arguably the best-tested reconstruction software in HEP
  - Run-1 data and yearly MC campaigns are $O(10^9)$ events each
  - gives a lot of possibilities to test topological cases
  - takes a lot of time

- ATLAS ran a big software campaign to clean up reconstruction software for Run-2
  - infrastructure changes (move from CLHEP to Eigen math library)
  - simplification of the Event Data Model (both transient & analysis side)
  - code cleanup and optimisations
  - updates to pattern recognition strategies
Outcome of the LS1 campaign

- CPU comparison during LS-1 without loss of physics performance

![Graph showing CPU comparison and software releases](image)

\( \sqrt{s} = 14 \text{ TeV} \)
\( \langle p \rangle = 40 \)
25 ns bunch spacing

Run 1 Geometry
pp \( \rightarrow t\bar{t} \)
HS06 = 13.08

Full reconstruction
Inner Detector only

> 1 year w/o working head release

> 5 CPU gain, no loss of physics performance

A. Salzburger, CHEP2015
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?

<table>
<thead>
<tr>
<th>Tracking</th>
<th>Geometry (TrkDetDescr)</th>
<th>Event Data Model (TrkEvent)</th>
<th>Extrapolation (TrkExtrapolation)</th>
<th>Fitting (TrkFitter)</th>
<th>Calibration, general (TrkTools)</th>
<th>Alignment (TrkAlignment)</th>
</tr>
</thead>
</table>

no ATLAS dependency

ATLAS specifications & dependencies

**InnerDetector**
- EDM (InDetRecEvent)
- Geometry (InDetTrackingGeometry)
- Tools (InDetRecTools)

**Calorimeter**
- EDM (CaloTrackingGeometry)

**MuonSpectrometer**
- EDM (MuonRecEvent)
- Geometry (MuonTrackingGeometry)
- Tools (MuonRecTools)
Introducing - The ATS project

<table>
<thead>
<tr>
<th>Tracking</th>
<th>G/A</th>
<th>G/A</th>
<th>G/A</th>
<th>G/A</th>
</tr>
</thead>
<tbody>
<tr>
<td>Geometry</td>
<td>(TrkDetDescr)</td>
<td>Event Data Model</td>
<td>(TrkEvent)</td>
<td>Extrapolation</td>
</tr>
</tbody>
</table>

**a-tracking-sw**

Attempt to encapsulate the ATLAS Tracking software from ATLAS

currently restricted access

<table>
<thead>
<tr>
<th>65 COMMITS</th>
<th>5 BRANCHES</th>
<th>0 TAGS</th>
<th>13.21 MB</th>
<th>ADD CHANGELOG</th>
<th>ADD LICENSE</th>
<th>ADD CONTRIBUTION GUIDE</th>
</tr>
</thead>
</table>

**A Tracking Software (ATS) Project**

This library is based on the track reconstruction software developed by the ATLAS Collaboration.

The main philosophy is to provide high-level track reconstruction modules that can be specified for detector technologies by simple extension.

- Event Data Model (EDM)
- Geometry
- track reconstruction tools

The library is attempted to build against Gaudi and Gaudi-Athena, while additional external dependencies are kept at a minimum.

double build philosophy
ATS - Decoupling from ATHENA

- Athena is based on Gaudi, but it’s not Gaudi itself
  - first step was to decouple from Athena and AthenaBaseComponents
  - allows building blocks
    AlgToolBase
    AlgorithmBase
    ServiceBase

- Other dependencies
  - Eigen (part of LCG)
  - Identifier
ATS - Decoupling from ATLAS GeoModel

- ATLAS uses GeoModel library for detector description
  - not supported/used by other experiments
  - there are many other geometry models on the market, e.g. FCC used DD4Hep/TGeo geometry
  - single connection from Tracking software to geometry library via a TrkDetElementBase

![Diagram showing TrkDetElementBase and GeoDetElement connections](image)
ATS - 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!

- 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_
ATS - Status, plans and timescales

- Prototype for geometry building decoupled from ATHENA
  - builds against Gaudi (Cmake) and Athena (cmt): done for core classes
  - next step: dummy detector building in Gaudi/ATHENA and compare: ongoing
  - geometry building from DD4Hep in FCC context: started

- Integration of Extrapolation engine and basic EDM
  - proof of extrapolation concept and little fast track sim: started

- Integration of Truth tracking module: not started

- Integration of KalmanFitter: started

  demonstrator for FCC Rome workshop April 2016 planned
  including Geometry - Fast Simulation - Truth Tracking - Track Fitting
Some concluding remarks

- Is it time to think about reconstruction toolkits?
  - simulation (Geant) has done this 30 years ago during LEP
    it was simply not possible for experiments to write their full simulation programs
    (needed a toolkit)

- (Track) reconstruction will be one of the most challenging aspects
  - already for Phase-2 upgrade
  - even more for FCC(-hh)

- New concepts, players and R&D is needed
  - and these are also coming in!

  e.g. Machine learning approaches for HEP:
  \[\text{IM LHC Maching Learning WG Meeting, tomorrow}\]
Some further reading

ATLAS RTF report  [ATL-SOFT-2003-010]

ATLAS Tracking Geometry  [ATL-SOFT-PUB-2007-004]


Common Tools  [ATL-SOFT-PUB-2007-007]


Fast Track Simulation  [ATL-SOFT-PUB-2008-001]
Backup slides
ATLAS Tracking Geometry - summary

- ATLAS Tracking Geometry is a geometry model designed around a few basic concepts:
  - core component `Trk::Surface` class (all nodes extend, contain surfaces)
  - embedded navigation through interlinking and volume glueing

- `Trk::Surface` class builds the core of the geometry and the Tracking Event Data Model (EDM)
  - track parameterisations are done on with respect to a surface
  - extrapolation, track fitting, measurement parameterisations

- The Tracking Geometry comes with an Extrapolation tool
  - uses the embedded navigation to move through the geometry
  - uses optimised propagation & material effects integration (point-like, dense)
The Tracking Geometry

- TrackingGeometry is a simplified geometry for reconstruction purpose
  - it was one of the first modules written for new Tracking

- It’s core components are is the Trk::Surface class
  - Surface
  - Layer extend Surface
  - Volume is a collection of Surfaces

Figure 2: The different surface types and the main boundary shapes as used in the EDM and implemented in the TrkSurfaces package. In general, different surface types were chosen for different intrinsic coordinate systems. This concept is broken for performance reasons by the introduction of the PerigeeSurface type that is a specialisation of the StraightLineSurface type. The picture also includes the prototype of a SaggedLineSurface which has been designed to describe the sagging of wires due to gravitation, a more detailed description of this aspect can be found in Sec. 7.

Figure 3: The three main volume types given by the different volume boundary classes. Volumes are mainly characterised by the set of confining boundary surfaces that are handled and owned by the VolumeBounds class. More complex volume shapes such as e.g. the bevelled cylinder shape can also be found in the TrkVolumes repository, they are mainly used for the construction of the complex TrackingGeometry of the Muon System.
The Tracking Geometry - navigation

- TrackingGeometry has two jobs
  - provide the detector surfaces and navigation through the detector
  - provide the material

- Part 1 - Navigation
  - navigation is provided by a “glueing” mechanism
  - this is the key (and complicated part of the geometry building)
The TrackingGeometry knows several navigation modes

- Static: that's the ID, everything is set up at building (let's talk about this)
- Dense: these are dense volumes mostly for the call
- Detached/Unordered: this is for the muon System
TrackingGeometry - simplification of nodes

- Layers are described as cylinders and disks
  - sensitive material in CURRENT tracking geometry only on those
  - even though the modules are perfectly known to the TrackingGeometry

```
materialStepR:materialStepZ
```
ATLAS Tracking SW EDM - TrackParameters

- TrackParameters have been entirely reworked during LS1
  - among with much of the ATLAS Tracking code (95% of code line savings)

<table>
<thead>
<tr>
<th>Package</th>
<th>C++</th>
<th>C/C++ Header</th>
<th>C++</th>
<th>C/C++ Header</th>
</tr>
</thead>
<tbody>
<tr>
<td>TrkParameterBase</td>
<td>63</td>
<td>561</td>
<td>11</td>
<td>214</td>
</tr>
<tr>
<td>TrkParameters</td>
<td>1715</td>
<td>602</td>
<td>0</td>
<td>52</td>
</tr>
<tr>
<td>TrkNeutralParameters</td>
<td>1425</td>
<td>663</td>
<td>0</td>
<td>48</td>
</tr>
<tr>
<td>ExtendedTrkParameterBase</td>
<td>0</td>
<td>295</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>ExtendedTrkParameters</td>
<td>1412</td>
<td>514</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>ExtendedTrkNeutralParameters</td>
<td>1416</td>
<td>514</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>6031</strong></td>
<td><strong>3149</strong></td>
<td><strong>11</strong></td>
<td><strong>266</strong></td>
</tr>
</tbody>
</table>
ATS binding geometry link - TrkDetElementBase

- Connects any favourite geometry module to ATS

```cpp
/** Identifier */
virtual Identifier identify() const = 0;

/** Identifier hash */
virtual IdentifierHash identifyHash() const = 0;

/** Return local to global transform */
virtual const Amg::Transform3D & transform() const = 0;

/** Return local to global transform associated with this identifier */
virtual const Amg::Transform3D & transform(const Identifier& id) const = 0;

/** Return surface associated with this detector element */
virtual const Surface & surface() const = 0;

/** Return surface associated with this identifier, which should come from the PrepRawData object (i.e. Surface s = PRD.detElement().surface( PRD.identify() ). This is only really relevant for the TRT (where there are several surfaces per detector element). For other detector elements it will just return surface() ... the Identifier is ignored. */
virtual const Surface & surface(const Identifier& id) const = 0;

/** Returns the full list of all detection surfaces associated to this detector element */
virtual const std::vector<const Surface*>& surfaces() const = 0;
```