A framework for vertex reconstruction in the ATLAS experiment at LHC

E.Bouhova-Thacker, Th.Koffas,V Kostyukhin, W.Liebig, M.Limper,
G.Piacquadio, K.Prokofiev, C.Weiser, A.Wildauer

on behalf of the ATLAS Collaboration

Computing in High Energy Physics, 22 - 27 of March 2009, Prague, Czech Republic.

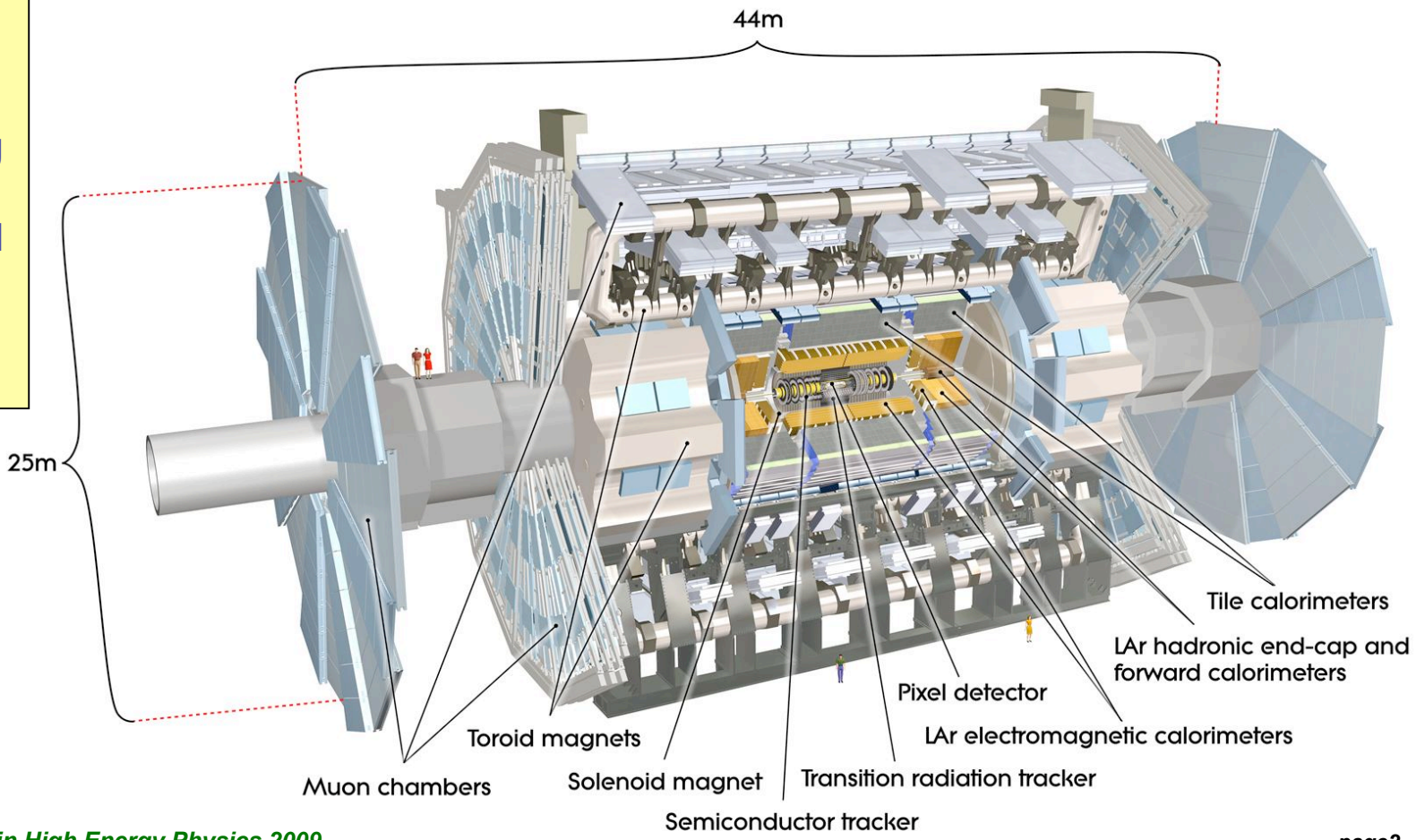
Outline

- The ATLAS experiment at the LHC.
- The ATLAS Inner Detector.
- The reconstructed vertex topologies.
- The framework principle.
- The Event Data Model.
- The Abstract Interfaces.
- Examples of implementations.
- Conclusions and Outlook.

The ATLAS Experiment at LHC

A multi-purpose particle detector at the Large Hadron Collider at CERN. Inner Tracking Detector with 2T magnetic field. Muon Spectrometer with average field of 0.5T. Designed to study pp collisions at the energy of 14 TeV at the LHC.

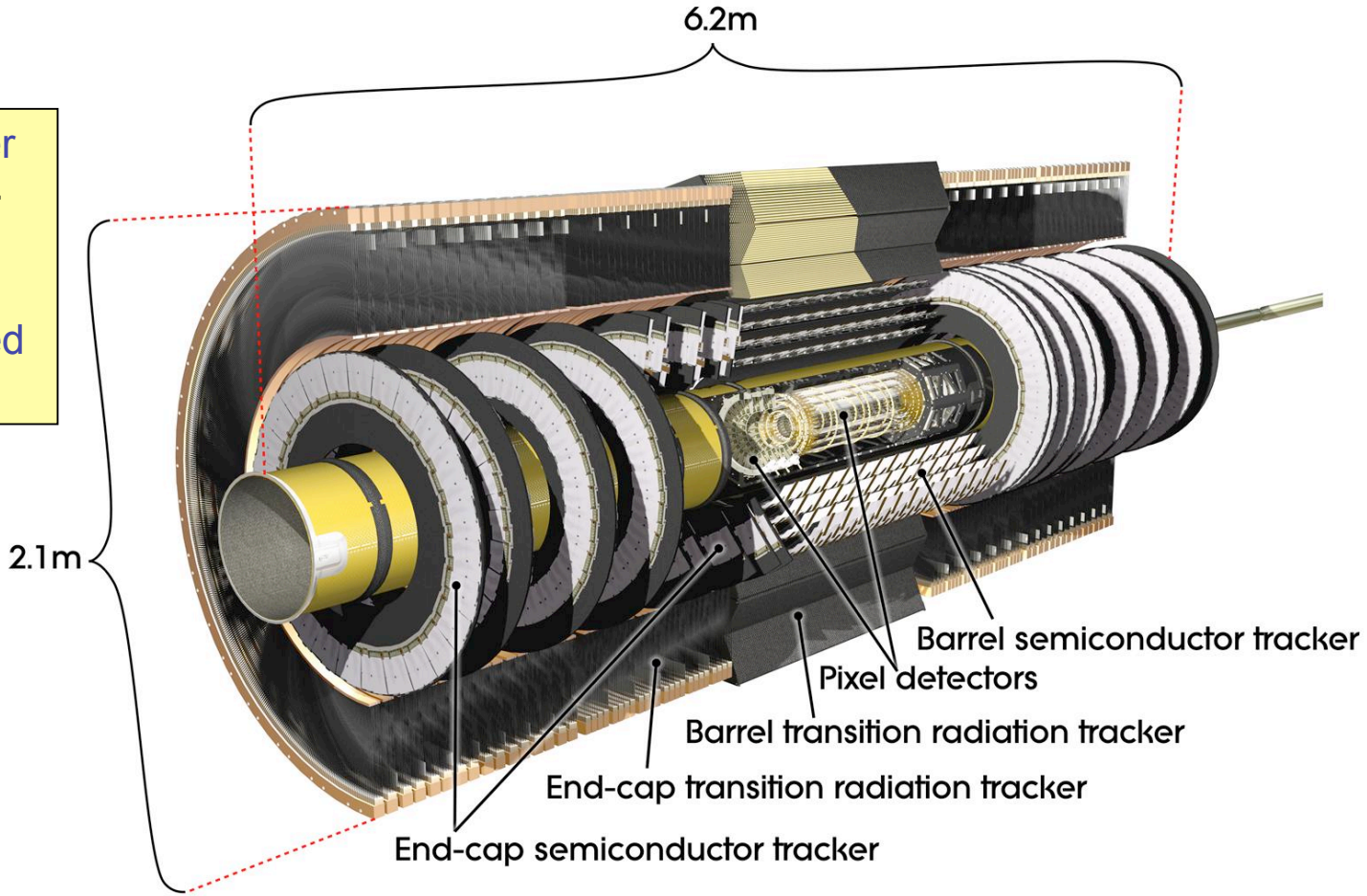
Average number of pp collisions per bunch crossing at the LHC is 4.6 at the initial phase and 24 at design luminosity.



The ATLAS Inner Detector

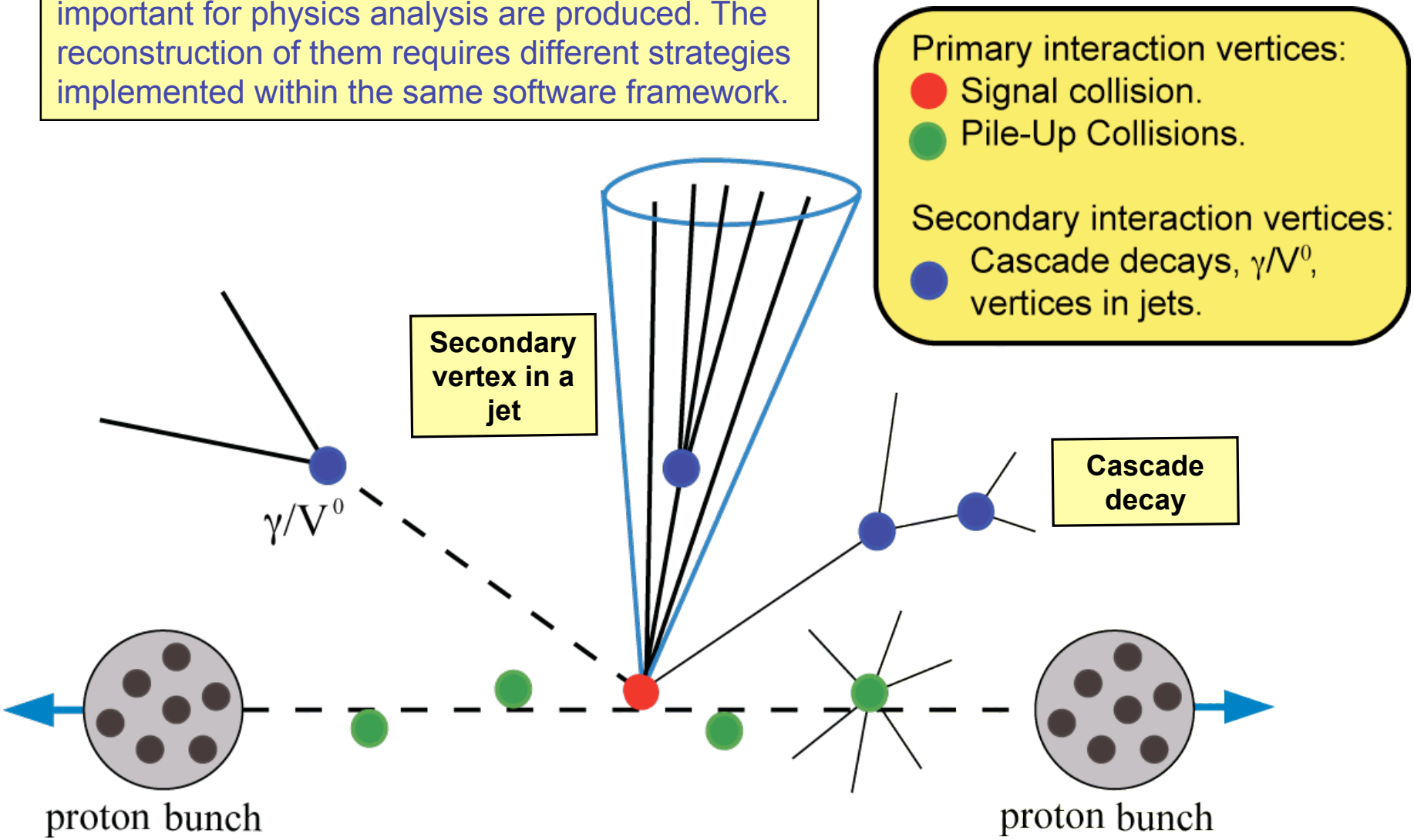
Pixel detector: 3 barrel layers at radii of 5, 8.8 and 12.2 cm and 3 endcap disks (each side); Rφ resolution about 10μm. Semiconductor tracker: 4 barrel layers of silicon microstrip detectors and 9 endcap wheels; Rφ resolution about 17μm. Transition Radiation Tracker with $e^{\pm}$ identification capability; Rφ resolution about 130μm.

On average, the Inner Tracker provides 3+4 3D and 36 2D measurements per trajectory of a charged particle.



Overview of Vertex Topologies

In *pp* collisions, several different vertex topologies important for physics analysis are produced. The reconstruction of them requires different strategies implemented within the same software framework.

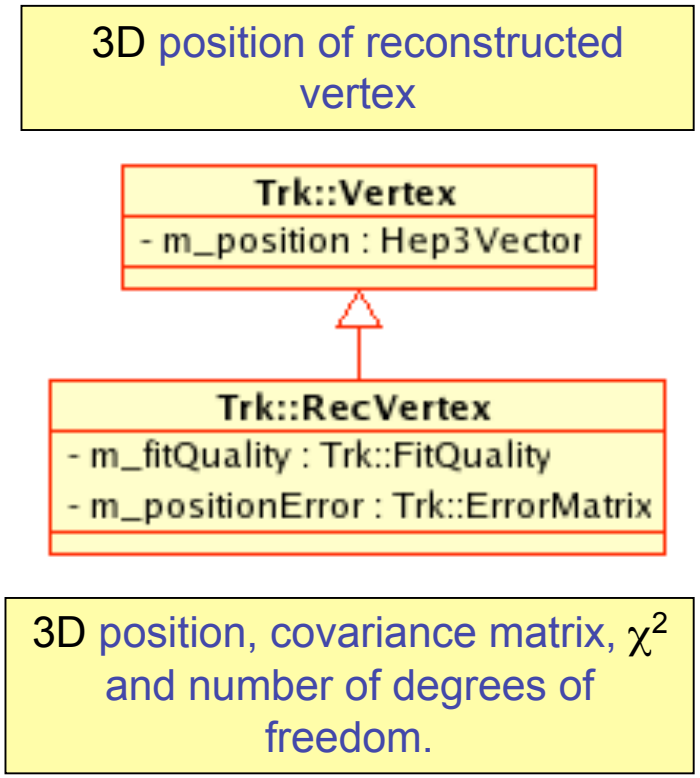


The Framework Principle

- Aim:
  - To create a common modular software environment to allow the reconstruction of different vertex topologies within the same framework.
  - The user should benefit from the common “look and feel” of reconstructing different vertex topologies.
  - The possibility of using the same algorithms and tools on both reconstruction and physics analysis levels.
- Current design: a framework based on object-oriented C++ with Python steering.
  - Use of a Common Event Data Model.
    - A group of classes representing reconstructed vertices and their relations to other objects used during the reconstruction process.
  - For all tasks related to vertex reconstruction, use of a common set of Abstract Interfaces.
  - Configuration of tools and algorithms through Python-based steering scripts.
  - Fully integrated in the ATLAS Athena reconstruction software.

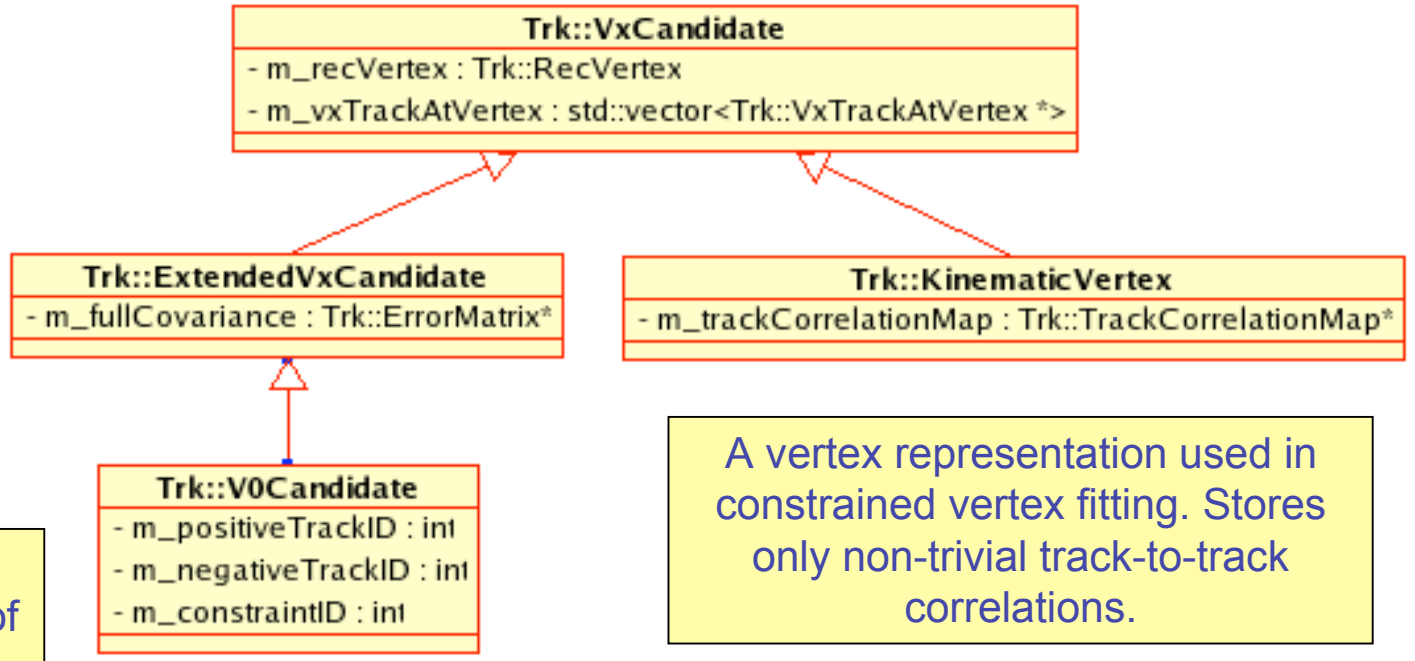
Event Data Model

- Data classes in which the information relevant to vertex reconstruction is stored.
- Reconstruction of different topologies requires different level of details to be stored in various EDM classes.
  - Several levels of inheritance.
  - Quantities, which are common to all topologies are stored and retrieved the same way.
  - Only the required amount of detail is stored in each object and written to disk.
  - To reduce the use of disk space, only the quantities which are impossible to re-calculate “on the fly” are retained.



Event Data Model

Base representation of the reconstructed vertex: vertex position, covariance matrix, fit quality and a vector of tracks fitted to this vertex.



Full covariance matrix including track-to-track and track-to-vertex correlations

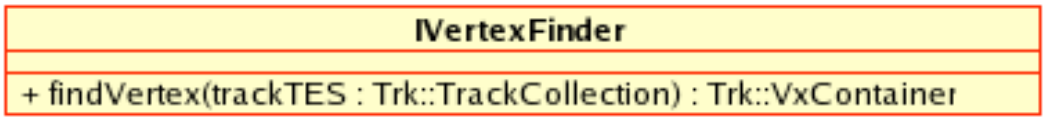
A V0 decay vertex. The masses (PIDs) of final state and of decayed particles are stored.

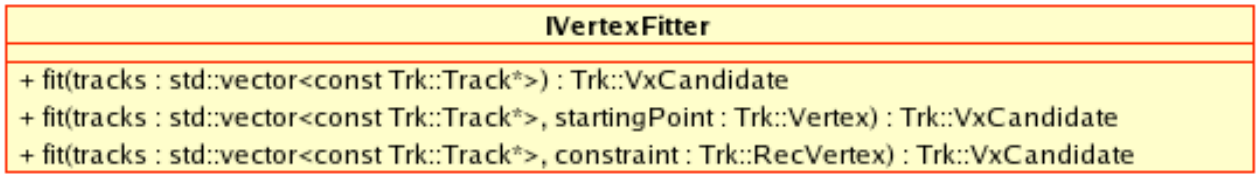
A vertex representation used in constrained vertex fitting. Stores only non-trivial track-to-track correlations.

Common Abstract Interfaces

- Abstract interfaces are defined for all reconstruction-related and helper tasks.
  - Each interface has several concrete implementations.
  - The implementation to be used at each step of the reconstruction is defined by the steering mechanism during the run time.

- The tools responsible for vertex finding (association of tracks to a concrete vertex hypothesis) and vertex fitting (reconstruction of vertex position, covariance matrix etc…) have separate interfaces.





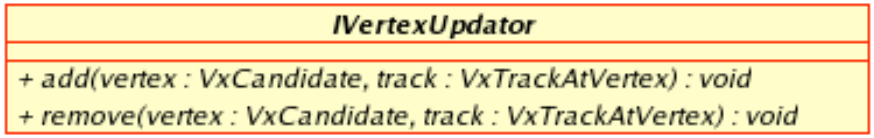
IVertexFinder : given a collection of reconstructed tracks, returns a container of reconstructed vertices.

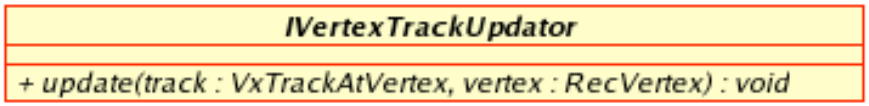
IVertexFitter : given a collection of reconstructed tracks, returns a vertex reconstructed from this collection.

Common Abstract Interfaces

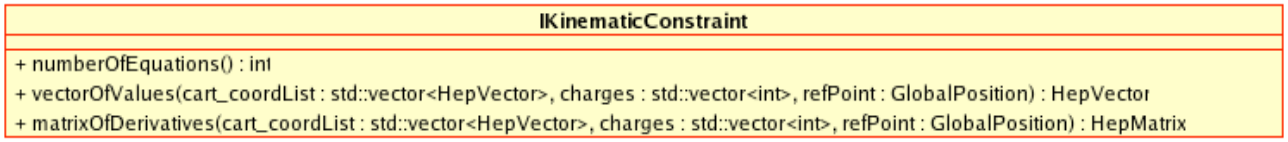
IVertexUpdator : Updates a vertex estimate with one trajectory at the time: adds or removes a trajectory from the estimate,

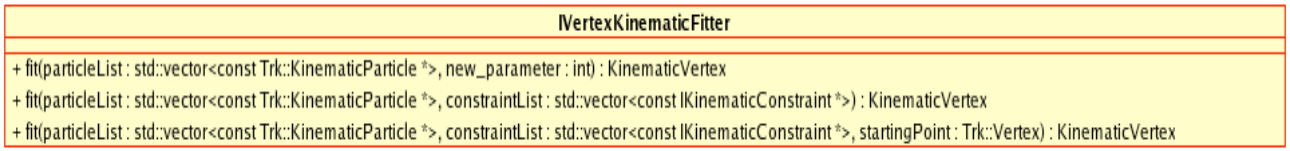
IVertexTrackUpdator : Re-fits the trajectory with the knowledge of the reconstructed vertex.





Interfaces for vertex fitting with additional constraints



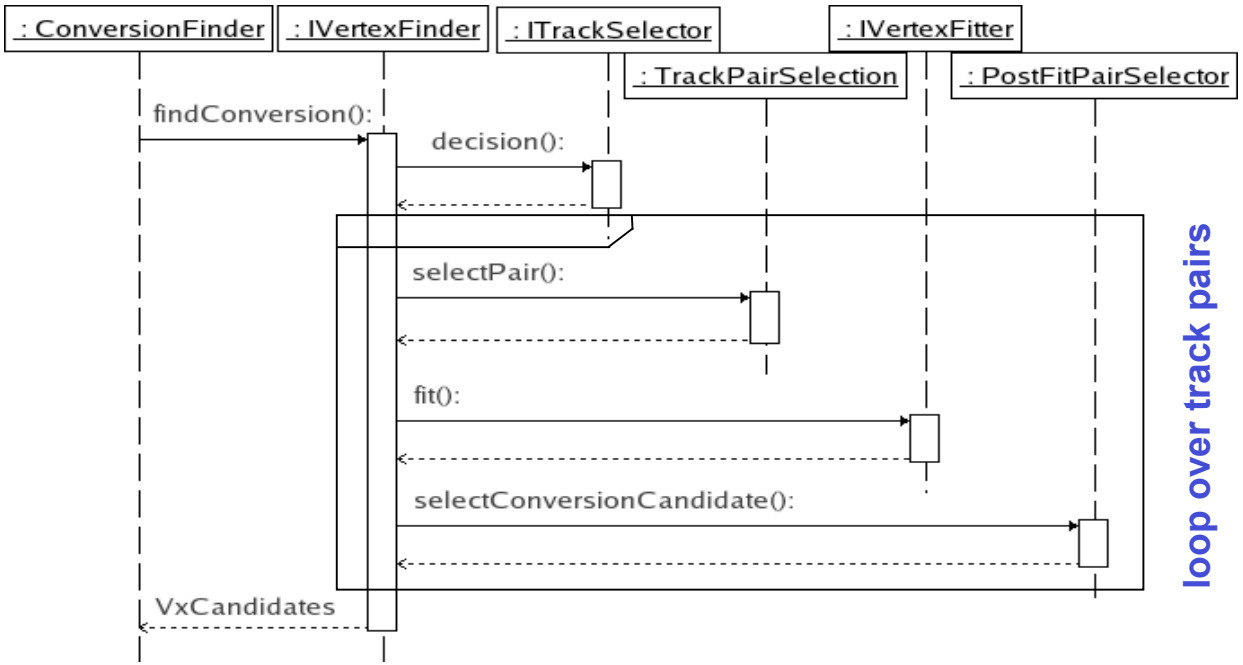


IKinematicFitter : vertex fitter with the possibility of application of additional constraints.

IKinematicConstraint : A base class for implementation of linearized constraints. Returns a matrix of derivatives and a vector of values calculated at an expansion point.

Examples of Implementation

Conversions: a photon converting into e^+e^- pair in the material of the detector. About 50% of produced photons will convert in the ATLAS Inner Detector. Reconstruction of the conversion vertices increases statistics for channels where the reconstruction of photons is crucial and allows for material mapping inside the tracker.



The *ConversionFinder* analyses a container of (electron) tracks and reconstructs conversion vertices using a dedicated *ConversionVertexFinder*. The latter pre-selects tracks using the interfaced *TrackSelector* and finds iteratively conversion candidates:

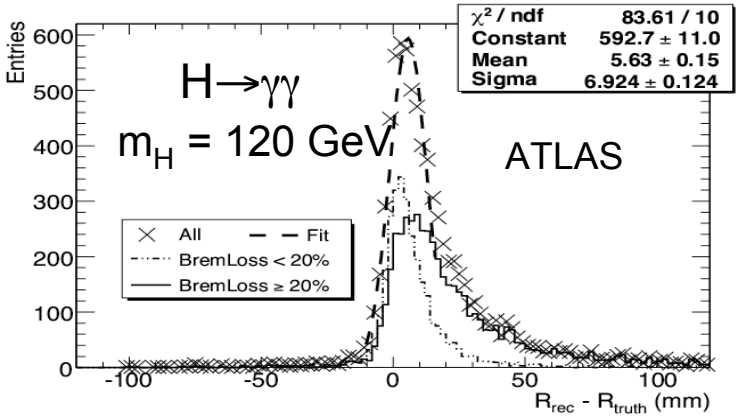
a) track pair pre-selection,

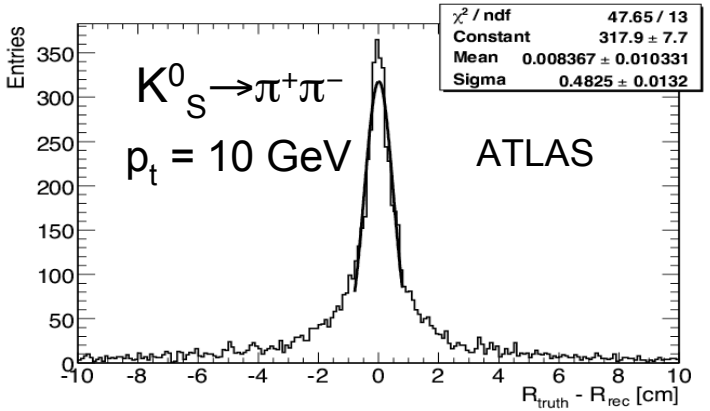
b) vertex fitting,

c) selection of conversion candidates.

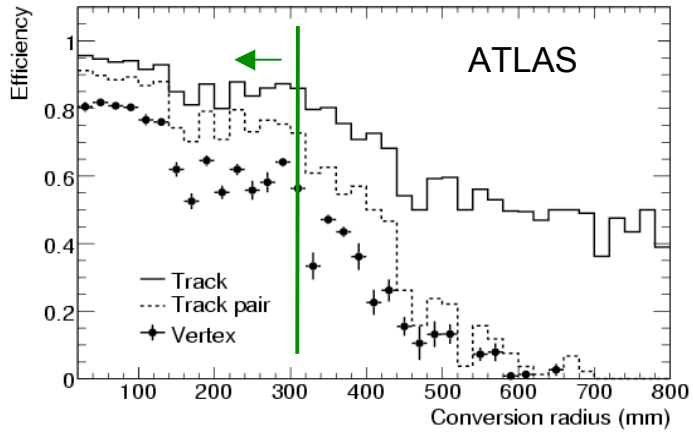
Examples of Implementation

Reconstruction of conversions: preliminary results. Distribution of residuals of transverse conversion radius.





The bremsstrahlung losses reduce the reconstruction quality in the case of H→γγ. For comparison, the distribution is symmetric in the case of $K^0_s \rightarrow \pi^+\pi^-$.



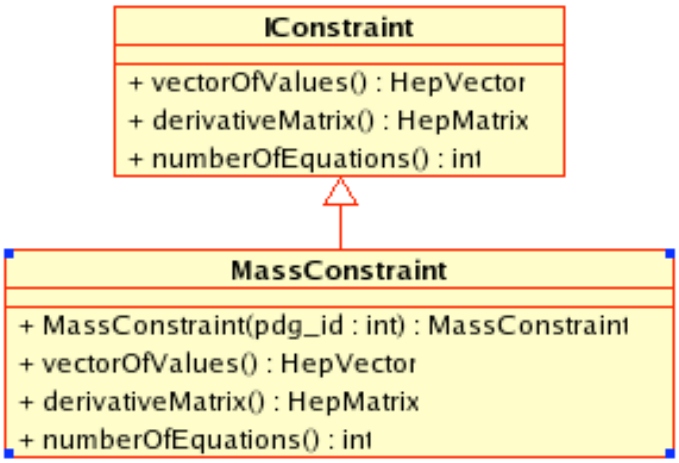
The efficiency drops at large radii because of complications in reconstructing both electrons.

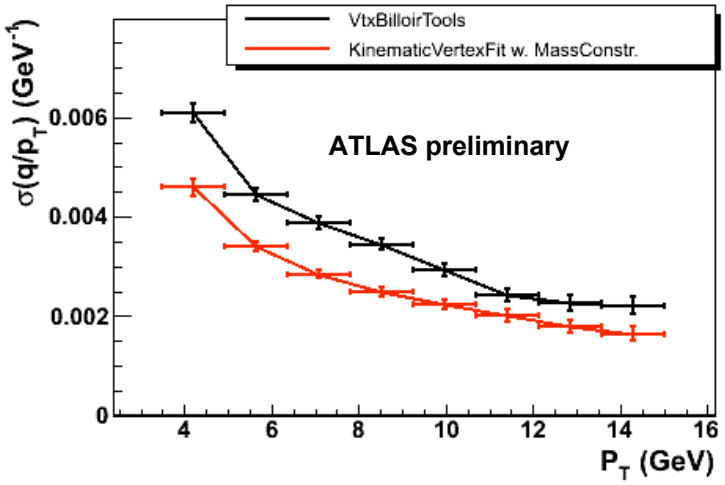
Examples of Implementation

Constrained vertex fitting: decay chain reconstruction, detector alignment etc..

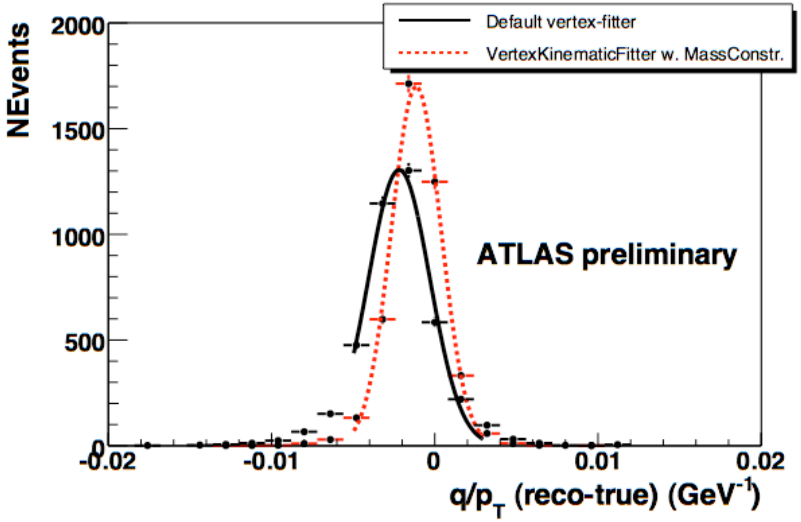
ATLAS has several “all in one” constraint fitting approaches: VKalVrt, V0Fitter..

In the new implementation, the constraints are implemented as separate classes with common abstract interface.





The application of the J/ψ mass constraint in $J/\psi \rightarrow \mu^+\mu^-$ decay leads to the improvement of resolutions on momenta of muon tracks.



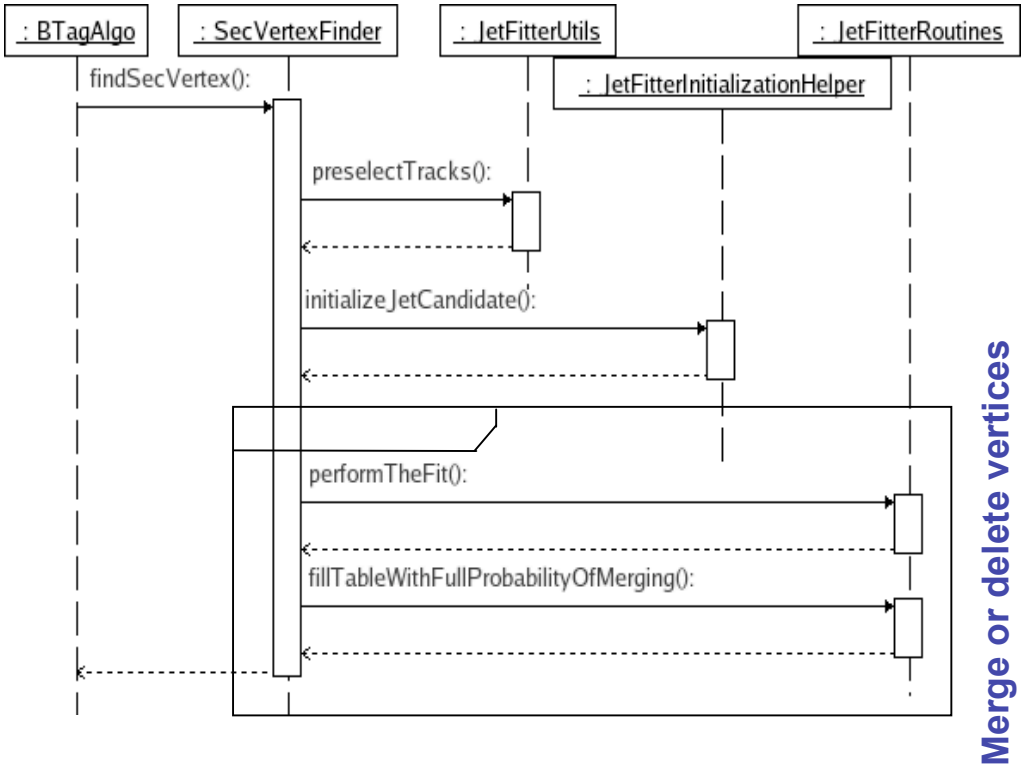
The use of a mass constraint leads to the recovery of shifts in particle momenta, which are present due to weak mode misalignment.

Examples of Implementation

Secondary vertex finding in jets. Exploiting secondary vertex topology to enhance *b*-tagging performance. Two implementations: *BTagVrtSec* and *JetFitter*.

1)Pre-selection of transversly displaced tracks.

2)Reconstruction of 2-track vertices, suppression of conversions, K_S and Λ's.



A) Inclusive secondary vertex finder:

3) Fit with surviving tracks

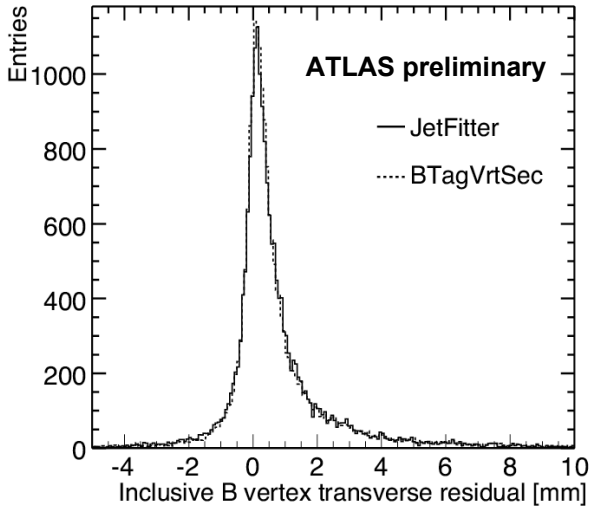
4) Iteratively remove bad tracks

B) Topological decay chain reco:

3) Initial fit with single track vertices along b-flight axis

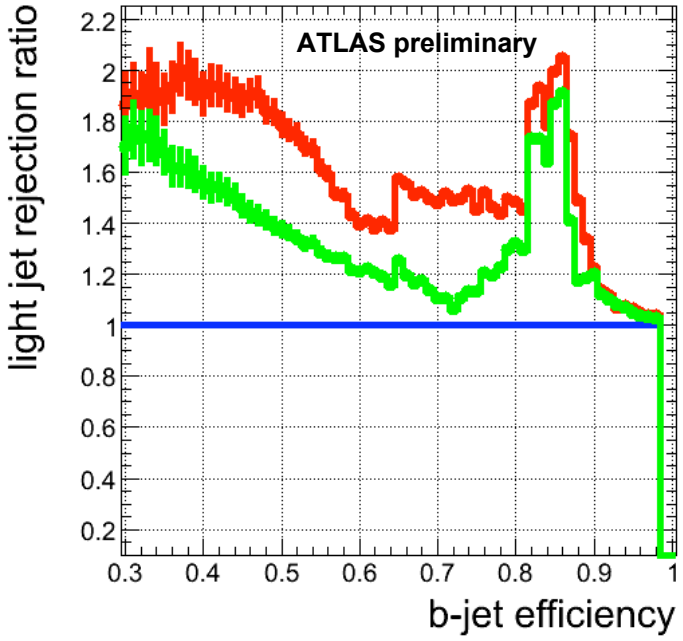
4) Iteratively merge pair of vertices or remove vertices

Examples of Implementation



Efficiency and purity of track-to-vertex association

		Track Association	
Algorithm	Topology	Efficiency	Purity
BTagVrtSec	1 inclusive B/D vertex	69%	92%
JetFitter	1 vertex	74%	91%
	1 vertex + 1track	80%	85%
	2 vertices	85%	89%



The light jet misidentification rate is decreased to

- 1/200 at 60% *b*-jet selection efficiency.
- To compare with about 1/70 after using only the impact parameter information.
- A factor of about 3 in improvement is observed.

IP3D/IP3D BTagVrtSec/IP3D JetFitter/IP3D

Conclusion and Outlook

- A framework for vertex reconstruction is implemented in the ATLAS Athena software environment.
  - The framework is based on a common Event Data Model and a set of abstract interfaces providing a common “look and feel” for the end-user.

- Several concrete implementations are currently finished. They include:
  - Reconstruction of primary vertices, reconstruction of secondary vertices, such as V0 and photon conversions, reconstruction of vertices in jets, constrained vertex fitting.

- The performance of implemented algorithms is currently being tested using Monte Carlo samples.