Tracking Event Model, Status

Output of the Track Event Model Task Force

O. Callot, M. Cattaneo, JA. Hernando, P. Koppenburg,

M. Merk, G. Raven, E. Rodrigues

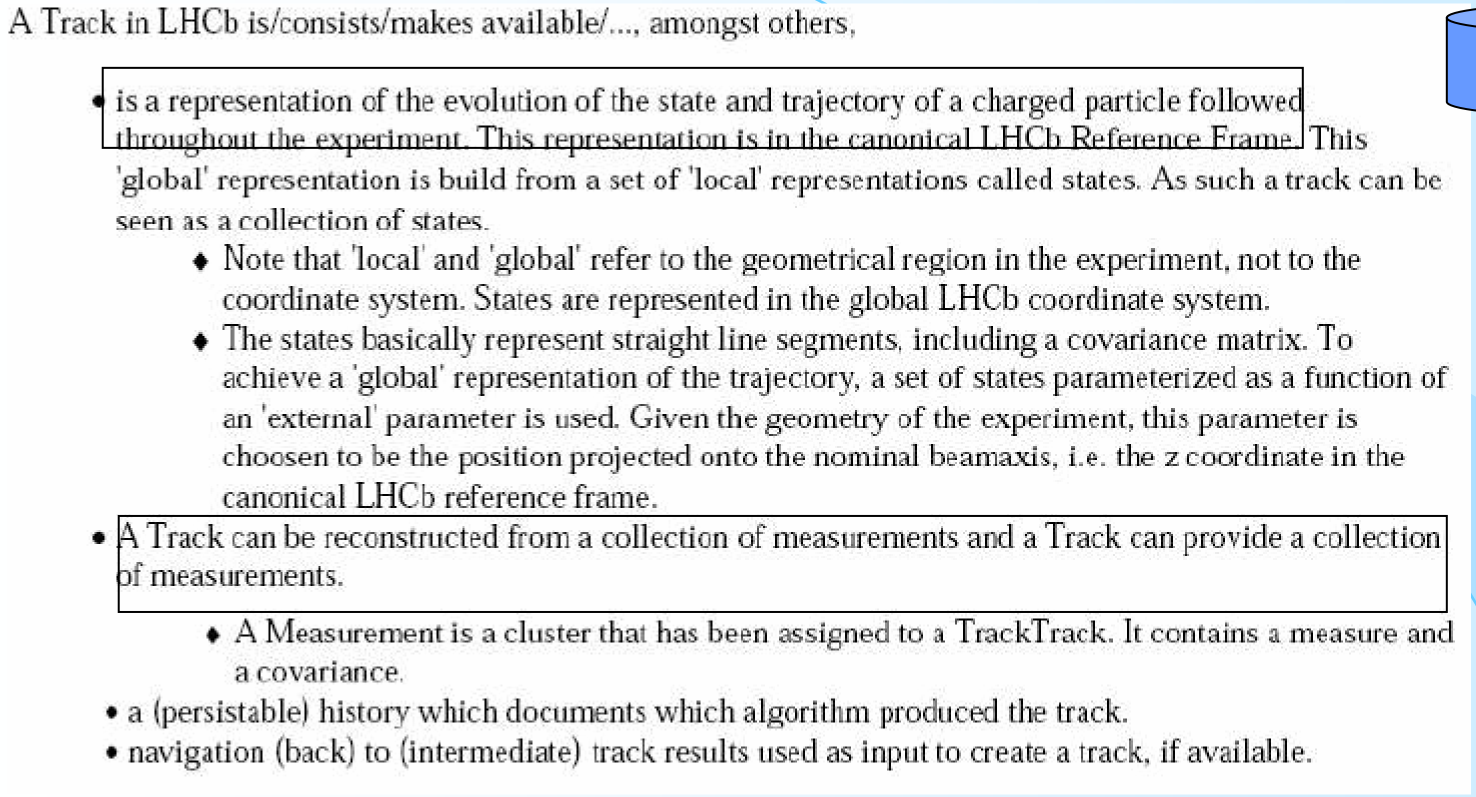
Status of the implementation of the Track Event Model

Jose A. Hernando, E. Rodrigues

Plan

- **Motivation:**
  - Revisit the tracking code to try to improve the design
  - Unify code on/off line and define an **interface for the clients**
  - Define **data** and **tools** base classes for and tracking developers and clients
- **Method:**
  - Reusing almost all the code: “adapting” and not “writing new code”
- **Task Force:**
  - Definition and requirements and user cases
    - Wiki: https://uimon.cern.ch/twiki/bin/view/LHCb/LHCbTrackModelTraskForce
  - Review of the implementation
    - Eduardos’s: http://erodrigu.home.cern.ch/erodrigu/lhcb/tracking
- **Plan:**
  - Step I: Interfaces for clients
    - Track, State, ITrackExtrapolator
  - Step II: Tracking interfaces
    - Measurement, Node, ITrackProjector, ITrackKalmanFilter
- **Scale:**
  - 6 months?

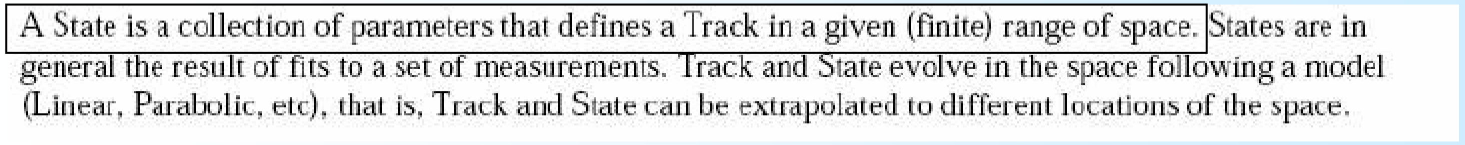
Task Force: definitions



A Track in LHCb is/consists/makes available/..., amongst others,

- is a representation of the evolution of the state and trajectory of a charged particle followed throughout the experiment. This representation is in the canonical LHCb Reference Frame. This 'global' representation is build from a set of 'local' representations called states. As such a track can be seen as a collection of states.
  - Note that 'local' and 'global' refer to the geometrical region in the experiment, not to the coordinate system. States are represented in the global LHCb coordinate system.
  - The states basically represent straight line segments, including a covariance matrix. To achieve a 'global' representation of the trajectory, a set of states parameterized as a function of an 'external' parameter is used. Given the geometry of the experiment, this parameter is choosen to be the position projected onto the nominal beamaxis, i.e. the z coordinate in the canonical LHCb reference frame.
- A Track can be reconstructed from a collection of measurements and a Track can provide a collection of measurements.
  - A Measurement is a cluster that has been assigned to a TrackTrack. It contains a measure and a covariance.
- a (persistable) history which documents which algorithm produced the track.
- navigation (back) to (intermediate) track results used as input to create a track, if available.

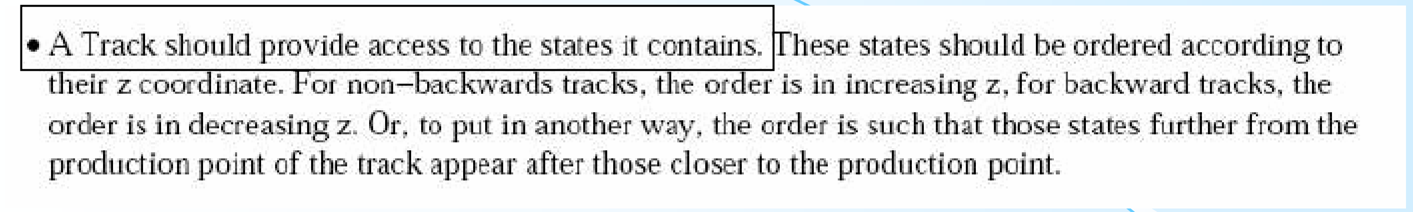
Track



A State is a collection of parameters that defines a Track in a given (finite) range of space. States are in general the result of fits to a set of measurements. Track and State evolve in the space following a model (Linear, Parabolic, etc), that is, Track and State can be extrapolated to different locations of the space.

State

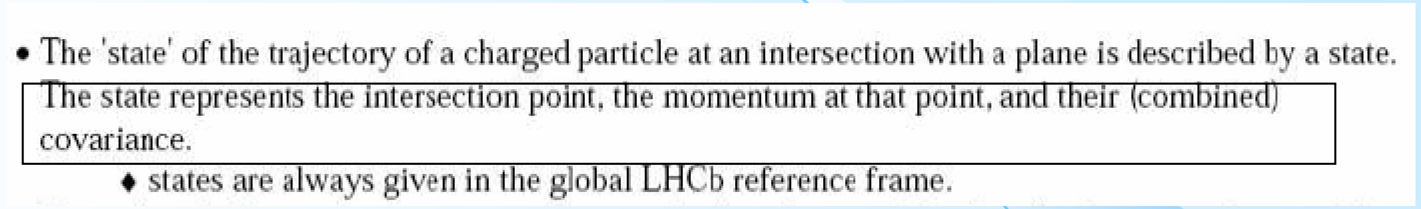
Task Force: requirements



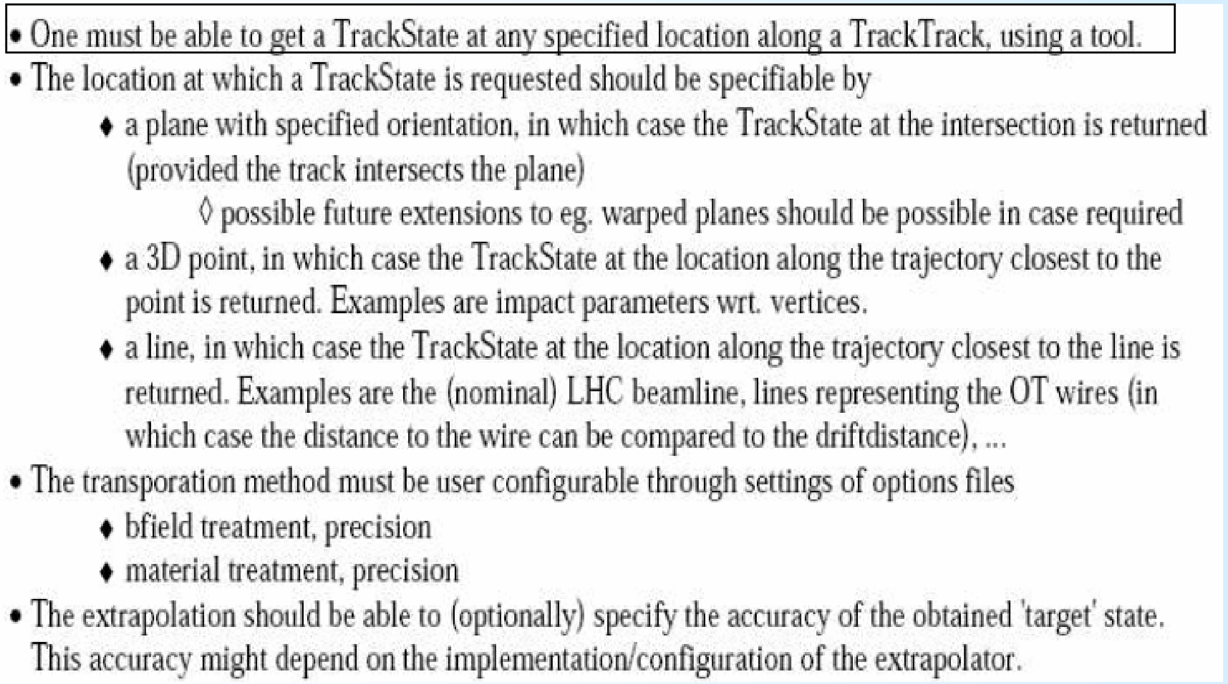
- A Track should provide access to the states it contains. These states should be ordered according to their z coordinate. For non–backwards tracks, the order is in increasing z, for backward tracks, the order is in decreasing z. Or, to put in another way, the order is such that those states further from the production point of the track appear after those closer to the production point.

Track

State



- The 'state' of the trajectory of a charged particle at an intersection with a plane is described by a state. The state represents the intersection point, the momentum at that point, and their (combined) covariance.
  - ♦ states are always given in the global LHCb reference frame.

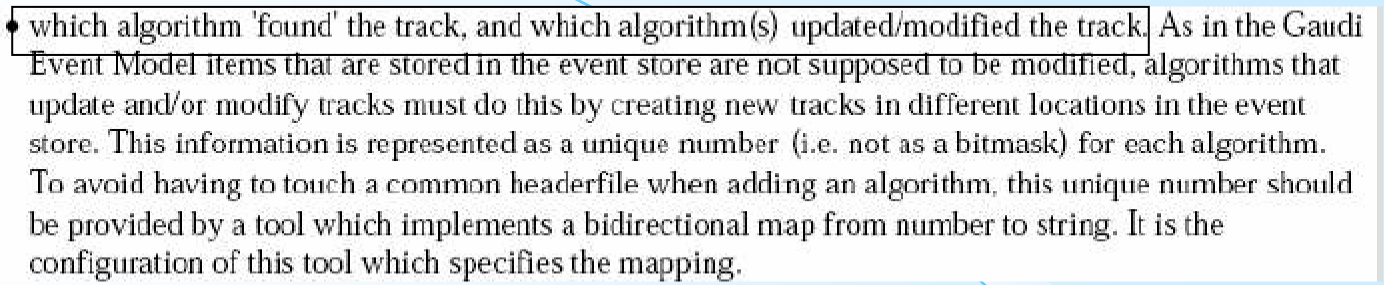


- One must be able to get a TrackState at any specified location along a TrackTrack, using a tool.
- The location at which a TrackState is requested should be specifiable by
  - ♦ a plane with specified orientation, in which case the TrackState at the intersection is returned (provided the track intersects the plane)
    - ◊ possible future extensions to eg. warped planes should be possible in case required
  - ♦ a 3D point, in which case the TrackState at the location along the trajectory closest to the point is returned. Examples are impact parameters wrt. vertices.
  - ♦ a line, in which case the TrackState at the location along the trajectory closest to the line is returned. Examples are the (nominal) LHC beamline, lines representing the OT wires (in which case the distance to the wire can be compared to the driftdistance), ...
- The transporation method must be user configurable through settings of options files
  - ♦ bfield treatment, precision
  - ♦ material treatment, precision
- The extrapolation should be able to (optionally) specify the accuracy of the obtained 'target' state. This accuracy might depend on the implementation/configuration of the extrapolator.

ITrackExtrapolator

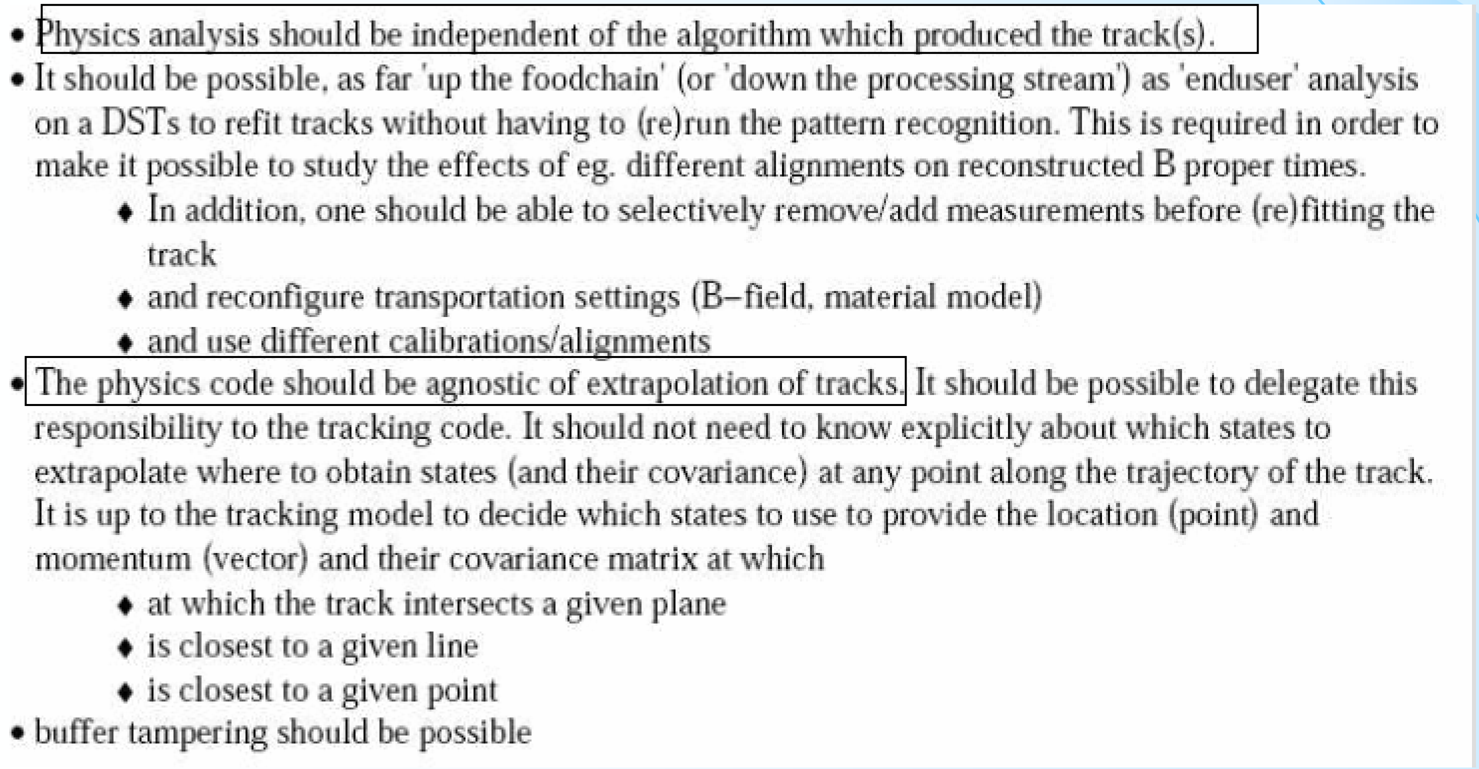
Task Force: more on requirements

History



- which algorithm 'found' the track, and which algorithm(s) updated/modified the track. As in the Gaudi Event Model items that are stored in the event store are not supposed to be modified, algorithms that update and/or modify tracks must do this by creating new tracks in different locations in the event store. This information is represented as a unique number (i.e. not as a bitmask) for each algorithm. To avoid having to touch a common headerfile when adding an algorithm, this unique number should be provided by a tool which implements a bidirectional map from number to string. It is the configuration of this tool which specifies the mapping.

Physics



- Physics analysis should be independent of the algorithm which produced the track(s).
- It should be possible, as far 'up the foodchain' (or 'down the processing stream') as 'enduser' analysis on a DSTs to refit tracks without having to (re)run the pattern recognition. This is required in order to make it possible to study the effects of eg. different alignments on reconstructed B proper times.
  - In addition, one should be able to selectively remove/add measurements before (re)fitting the track
  - and reconfigure transportation settings (B–field, material model)
  - and use different calibrations/alignments
- The physics code should be agnostic of extrapolation of tracks. It should be possible to delegate this responsibility to the tracking code. It should not need to know explicitly about which states to extrapolate where to obtain states (and their covariance) at any point along the trajectory of the track. It is up to the tracking model to decide which states to use to provide the location (point) and momentum (vector) and their covariance matrix at which
  - at which the track intersects a given plane
  - is closest to a given line
  - is closest to a given point
- buffer tampering should be possible

Step I: Track, State, (the most regarded classes…)

Track

A TRACK:

bitfield-flag: type, history, historyfit, status and flags

chi2/ndof, ndof: quality of the fit

<State*> :"transient" states and physic state

<Measurement*> :

<Node*> : (state-measurement => residual)

<Track*> : ancestor

<LHCbID>: link MC, Clusters (measurements)

Methods:

Access to physic state: *p,pt, slopes, position*

Access states: *at z, plane, LOCATION*

Persistency:

bitfield-flag, quality, physic state and LHCbIDs

the rest on demand!

State

A STATE:

bitfield-flag: type, location

state-vector, covariance, z

Methods:

Access to physics contents: pt(),p()

ITrackExtrapolator

A Extrapolator: extrapolate a Track/State

User methods: propagate(track,z,state)

Main method: propagate(state, z)

Methods:

propagate track, state to z

in the way: propagate to plane, line, point

physics access: p,pt…

Step II: Measurement, Node, Projector (the poor brothers…)

Measurement

A Measurement:

bitfield-flag: type (ie RVelo)

measure, error (double)

"z" and LHCbID

Node

A Node:

Measurement* ("refined")

Internal?…

State*

residual, error

Methods:

chi2(), …

ITrackProjector

A Projector: compute residuals

Main method: project(State, Measurement)

Internally deals with the Alignment/Calibration

(I think) it accept the two approaches:

I) global-local-global; II) global

Methods:

residual, chi2, node, ProjectionMatrix (H)

IKalmanFilter

A KalmanFilter (interface)

methods:

fit(Track,State seed);

filter(Track,State seed);

filter(State, Measurement)

Play and we will see…

The packages (quick look)…

Event/	
TrackEvent	CPT
Trg/	
TrgConverters	CT
TriggerVelo	W…
TriggerVeloTT	…
TriggerForward	…
Vis/	
SoEvent	CW

Tr/	
TrConverters	CT
TrackIdealPR	CT
TrackExtrapolators	CPT
TrackFitEvent	CP
TrackProjectors	C
TrackInterfaces	W
TrackTools	CW
TrackMCTools	CW
TrackFitter	CW
TrackPython	CPT
TrackSimulator	…

C: compiles (Track version 13/5/5)

T: preliminary tested,

P: exposed to Python

W: work on progress

…: next…

Prevision: 27/05/05 compiling

The packages…

Event/TrackEvent:

- Track, State, Measurement, Node
- TrackKeys, StateKeys
  - enums for the flags…

Tr/TrConverters

- TrFitTrack2TrackConv, Track2TrFitTrackConv
  - Algorithms to convert: TrFitTrack <-> Track

Tr/TrackExtrapolators

- Track<T>Extrapolator:
  - T: Linear, Parabolic, FastParabolic, Herab, (FirstClever-> Master)

Tr/TrackFitEvent

- <T>Measurement, FitNode, MeasurementProvider
  - T: OT,VeloPhi,VeloR,IT
    - the
- FitNode: Node for the Kalman Filter

Tr/TrackIdealPR:

- Tracks (with LHCbID fromMC)

Tr/TrackProjectors

- <T>Projector: VeloR,VeloPhi,IT,OT
- Master: residuals to any Measurement

Tr/TrackInterfaces

- ITrackExtrapolator,ITrackProjector, ITrackFitter

Tr/TrackTools:

- BIntegrator, TrackPtKick, MeasurementProvider

Tr/TrackMCTools

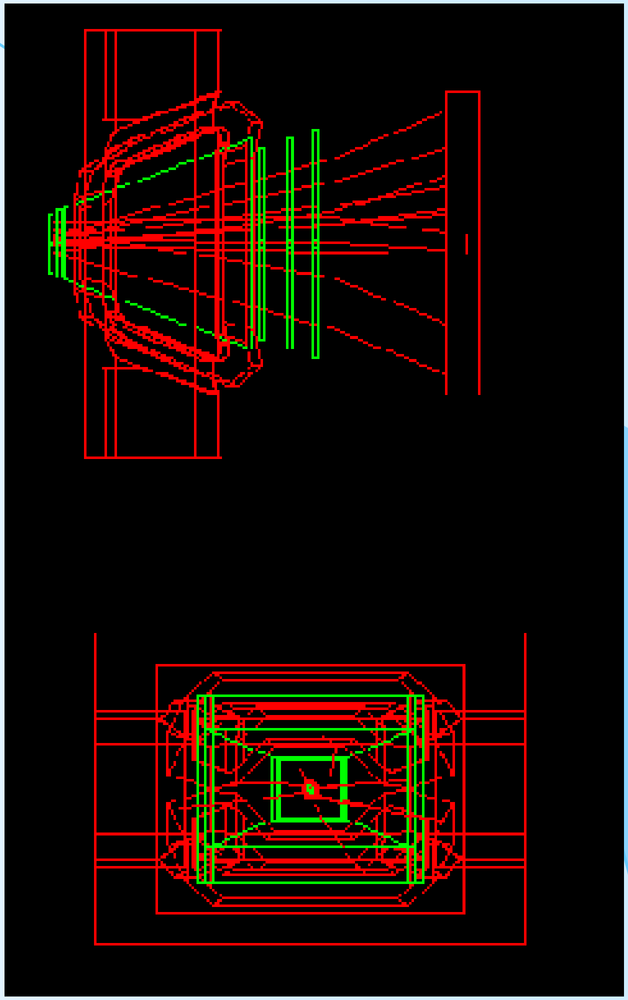
- TrackAcceptance

Tr/TrackFitter

- **KalmanFilter Tool**
  - **delegate: ITrackExtrapolator, ITrackProjector**
  - **Fit(Track,State seed):**
  - **Filter(State,Measurement)**

The packages II

- **Tr/TrackPython:**
  - Tools in Python
    - ITrackExtrapolator, ITrackProjector,ITrackFitter
  - Python scripts:
    - translate_tracking.py
    - automatic translation
- **Trg/TrgConverters:**
  - TrgTrackToTrack, TrackToTrgTrack
    - TrgTrack <-> Track
- ***Trg/TriggerVelo…***
- **Vis/SoEvent**
  - SoTrackCnv.cpp
    - Tracks in **Panoramix**
    - Next: Measurements, States and maybe Nodes



Interactive reconstruction

- **Interactive reconstruction?:**
  - Via Python:
    - GaudiPython and 'Bender'
      - P.M >> gaudi.run(1)
      - Vanya DaVinci tools and LoKi
    - Interaction with Panoramix (R.Ruf)
      - From Event Display object to Python prompt -> refit,…
  - In: Tr/TrackPython package
  - Benefits:
    - Interactive:
      - Debuging and testing the reconstruction
    - Developing:
      - Simple for newcomers to start
      - Fast and Easy prototyping:

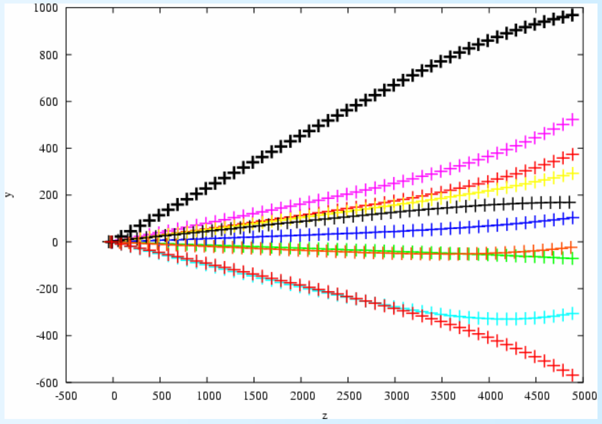
- **Toolkit:**
  - The elements:

    Track | ITrackExtrapolator

    State | ITrackProjector

    Measurement | ITrackKalmanFitter
  - Code in C++ and Python
  - Element missing (tool):

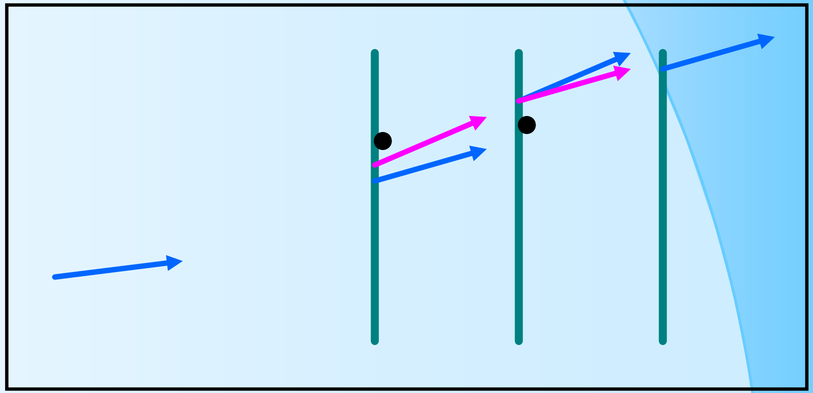
    MeasurementProvider



Interactive reconstruction

- **Measurement Provider (tool)?:**
  - A smart holder and fast provider of Measurements
  - Similar to O.C. M.N in Trigger and TSS but with the TEM
  - Holders: in trees?:
    - ‘sectors’: isInside(x), id(), plane()
  - Provider:
    - On demand:
      - At ‘TTX1’
      - At x (3d point) in tolerance
        - Residuals or sigms

- **TrackSimulator (tool):**
  - Generate Measurements from a Seed State
  - Process:
    - Measurements-> Track(States)
    - Track(State)->Measurements
  - Reusing:
    - Extrapolator, and Projectors



Status and plans

- **Task Force:**
  - Task Force has defined:
    - 6meetings, long –heated- and fruitful discussion
    - Definitions and requirements done
    - ‘almost’ final revision of main classes
- **Step I:**
  - Track/State: Implementation revisited 13/05/05
    - To be ready with the current status of packages: 27/05/05
- **Step II**
  - Preliminary versions: Measurement, Node, Projector
    - To use and see how they work
- **Plans:**
  - Pattern Recognitions packages:
    - Should fill the list of LHCbID of the Track
  - Fitting
    - Some recoding of the fitting, most already done:
      - Extrapolators, Projectors and Kalman Filter
    - Testing
      - Delicate work…
    - An eye in the alignment…
  - Visualization and Interactivity
  - MC link
    - General use of LHCbIDs, link with the MC via LHCbIDs
- **Many front, small forces**