Tracking Event Model, Status

Status of the implementation of the Track Event Model

Jose A. Hernando, E. Rodrigues

1. The plan, and the classes (again)
2. The packages modified or to be modified
3. Interactive reconstruction
4. Some ideas
5. Conclusion and plans
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 a Track! (for on/off line)
  - Define data and tools base classes for and tracking developers and clients

- **Method:**
  - Modify the current code adiabatically
  - Reusing almost all the code: “adapting” and not “writing new code”

- **Organization:**
  - Task Force (G. Raven) to:
    - ‘define the classes, requirements and implementation constrains’

- **Plan:**
  - Step I: Interfaces for clients
    - Track, State, ITrackExtrapolator
  - Step II: Tracking interfaces
    - Measurement, Node, ITrackProjector, ITrackKalmanFilter

- **Scale:**
  - 6 months
Step I: Track, State, (the most regarded classes…)

**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*>: (aggregate state-measurement => residual)
- <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!

**A STATE:**
- bitfield-flag: type, location
- state-vector, covariance, z

**Methods:**
- Access to physics contents: pt(), p()

**ITrackExtrapolator**

**A Extrapolator: extrapolate a Track/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**
- bitfield-flag: type (ie RVelo)
- measure, error (double)
- "z" and LHCbID

**Projector**
- Project a state into a measurement
- **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)

**Node**
- **type** (i.e. RVelo)
- Measurement* ("refined")
- State*
- residual, error
- **Methods**:
  - chi2(), ...

**KalmanFilter (interface)**
- **methods**:
  - fit(Track, State seed);
  - filter(Track, State seed);
  - filter(State, Measurement)

*Play and we will see...*
The packages (quick look)...

C: compiles (Track version 13/5/5)
T: preliminary tested,
P: exposed to Python
W: work on progress
...: next...

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

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


The packages...

- **Event/**
  - **TrackEvent:**
    - Track, State, Measurement, Node
    - TrackKeys, StateKeys
      - enums for the flags...

- **Tr/**
  - **TrConverters**
    - TrFitTrack2TrackConv, Track2TrFitTrackConv
      - Algorithms to convert: TrFitTrack <-> Track
  - **TrackExtrapolators**
    - Track<T>Extrapolator:
      - T: Linear, Parabolic, FastParabolic, Herab, (FirstClever-> Master)
  - **TrackFitEvent**
    - <T>Measurement, FitNode, MeasurementProvider
      - T: OT,VeloPhi,VeloR,IT
        - the
      - FitNode: Node for the Kalman Filter
      - MeasurementProvider:
        - returns a Measurement from a LHCbID
        - to be move to Tr/TrackTools
The packages II

- **Tr/**
  - **TrackIdealPR:**
    - TrueTrackCreators
      - Algorithm: From MCParticles to Clusters to LHCbID to Measurements
  - **TrackProjectors**
    - &lt;T&gt;Projector
      - VeloR, VeloPhi, IT, OT and **Master**
        - Reusing the code from MeasurementOnTrack
        - The master projector projects any measurement
          - it dispatches the projection to the specific projector, project(State, Measurement)
  - **TrackTools**
    - Interfaces:
      - ITrackExtrapolator, ITrackProjector, ITrackKalmanFilter
        - (before in Kernel/LHCbInterfaces)
    - Tools:
      - Bintegrator, TrackPtKick, TrackReconstructible, TrackAcceptance, TrackSelector
  - **TrackFitter**
    - **KalmanFilter Tool** (A tool to fit/filter a Track or a State)
      - Two external tools set by options: ITrackExtrapolator, ITrackProjector
      - **Fit(Track, State seed):**
        - fitTrack using a seed state (filter only, filter+smoother)
      - **Filter(State, Measurement):**
        - update the state, using the measurement
The packages III

- **Tr/**
  - **TrackPython:**
    - Expose to Python the Tools Interfaces
      - ITrackExtrapolator (soon: ITrackProjector, ITrackKalmanFilter)
      - In future (ITrackSimulator, IMeasurementProvider) TrackProjectors
    - Python scripts:
      - translate_tracking.py
        - automatic translation of code to the 'new' tracking event model
  - **Trg/**
    - **TrgConverters:**
      - TrgTrackToTrack, TrackToTrgTrack
        - Conversion: TrgTrack <-> Track
    - **TriggerVelo, TriggerVeloTT, TriggerForward**
      - TriggerVelo Private version to re-adapt to the last version of Track
        - Re-adapt the Trg reconstruction packages of DC04 (DV12 series) for the new Track
        - Compare the Trg (DC04) tracking with the new pattern recognition tracking code.
        - Backwards compatibility:
          - with minor modifications (TrackEvent, TrackFitEvent?) we can run in DC04 data.
  - **Vis/**
    - **SoEvent**
      - SoTrackCnv.cpp
        - Drawing the tracks in *Panoramix*
          - Improvements to draw: Measurements, States and maybe Nodes
Interactive reconstruction

Interactive reconstruction?: Via Python

- Already there:
  - GaudiPython and ‘Bender’
    - Expose the Gaudi framework to Python: `>> gaudi.run(1)`
    - Expose most of DaVinci tools and LoKi ‘metalenguage’: ‘Bender’
  - Interaction with Panoramix and the event display (T.Ruf)

- In: Tr/TrackPython package

- Benefits:
  - Interactive:
    - Debuging and testing the reconstruction
      - Event by event, track by track
  - Developing:
    - Simple for newcomers to start
      - A toolkit
    - Fast developing: 4 times faster than in C++
    - Easy prototyping: later you code in C++ with clear ideas
    - In fact, it run fast as it uses underneath the C++ code
Interactive and with display

- **Python:**
  - Just import modules

```python
pol = extrapolator("TrackParabolicExtrapolator")
state = track.physicsState().clone()
z = 3000.
pol.propagate(state, z)
print state.y()
```

**Preliminary: Tracks in Panoramix**

**A scatter plot from a Python prompt**
Some ideas: TrackSimulator

- **TrackSimulator**
  - Simulator Tool: TrackSimulator
    - **Main method:** `Simulate(Track&, const State& seed)`
      - It will fill the Track with a collection of simulated measurements
    - **Idea:** simulate a Track with Measurement starting from a seed-State
      - Straight forward reuse of the Tracking Tools
    - **To do:**
      - Check that the KalmanFilter is correctly implemented
      - To check if the Extrapolator follows realistically the MCParticles
      - To do alignment studies
    - **Setup of the Tool**
      - A list of planes, or labels locations, or ‘z’ positions with the type of Measurements
      - A Master TrackProjector and an TrackExtrapolator.

Do we want?:

Measurement->Cluster->Digit->buffer bank
Some ideas: toolkit reconstruction

The toolkit elements:
- Can you do the PR and fitting with this elements?
- A missing piece: MeasurementProvider (Tool):
  - A smart storage and fast provider of Measurements
    - Methods (design ideas…), return a ordered list of measurements
      - orderByResidual(x,tolerance),
      - orderBySigma(x,sigmas), where x: 3D point
  - Using internal holders of Measurements (in tree hierarchy)
    - A holder class that could (design ideas…)
      - Methods: plane(), isInside(x) -a box-, id(), etc..
- An aprox.. Example
  - From a state-seed extrapolate ‘TT’ planes
  - Get the measurements in order of sigmas around the extrapolated points
  - Make segments with them and fit them, select them according with a chi2 criteria
    - We have a collection of possible pt values associated to the seed,
Status and plans

- **Step I:**
  - Task Force has defined: Track and State
    - They are usable Track and States for:
      - Pattern Recognition, Fitting, Trigger and Offline
  - Implementation revisited 13/05/05
    - To be ready with the current status of packages: 27/05/05

- **Step II**
  - Task Force has defined 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.
    - Testing of the Extrapolators, Projectors and KalmanFilter
      - 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**