A proposal for a tracking data model

A proposal for a tracking data model

Jose A. Hernando (CERN) , E. Rodrigues (NIKHEF)

Discussions (via email and private) with input from O. Callot, M. Merk, M. Needham, T. Ruf, J. van Tilburg, to:

- *define base classes for tracking that can be used by the trigger and offline reconstruction*
- *define the output of the tracking reconstruction*

base classes that will facilitate the development of new pattern-recognition and fitting algorithms

Details of last proposal (not complete i.e. full functionality not yet described…) at:
erodrigu.home.cern.ch/erodrigu/lhcb/tracking/event_model/2004-09-13/index.html

A proposal for a tracking data model

Definition of a tracking data model:

A collection of BASE classes

1. **Define Input/Output of the tracking algorithms**
    1. **Unify the output of the Trigger and Offline reconstruction**
    2. **Declare the data that other sub-detectors algorithms will use from the tracking.**
2. **Create interfaces for (Gaudi) tools that operate on tracking classes**
    1. **I.e compute efficiencies, matching with MC, raw-buffer, etc.**
3. **Share common data structures to develop code:**
    1. **Adding/reusing other pattern recognition and fitting methods**
    2. **I.e a "measurement" smart server**

A proposal for a tracking data model

The gain and the freedom:

1. **The bases define a standard:**

   We gain in generality without limiting developer freedom

2. **Trigger and Offline should provide the base classes (as much as they can)**

   … "specific" code can use "internally" local tracking classes

   … to be avoided as much as possible …

3. **An algorithm using base tracking classes is a general algorithm**

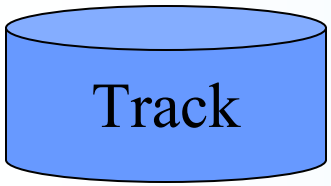
   (I.e RICH reconstruction algorithms, Muon algorithms, etc.)

4. **A tool that operates with base tracking classes is a general tool**

   Ie. Compute efficiencies, matching with MC, exRawBuffer, etc.

Note: **The bases classes should be designed without any external constrain (i.e persistency).!**

Track



Track
{flags}
Chi2, ndof
{States*}
{Measurements*}, …
----
clone()
reset(), position(), pt()
, …

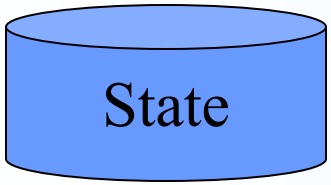
General agreement reached … !

Idea would be to make the states not directly accessible to the end-user … fine as long as position(z), etc. are public methods …

A Track:

1. **A collection of states.**
   1. **Each state is a local parameterization of the track (trajectory)**
2. **A collection of measurements (and "nodes"? <-> issue still being discussed …):**
   1. **A measurement is a signature of the track "in a detector"**
   2. **(A node could be the link between the measurement and the state in the measurement location.)**
3. **The quality of the agreement of the measurement with the track model**
   1. **chi2, ndof**

State



General agreement reached …

State
---
type
state vector
cov-matrix
z
---
position()
slopes()
momentum()
pt()

Type is an *enum*

(x,y,tx,ty,q/p)

Connection with Geometry

Connection with Physics

A State:

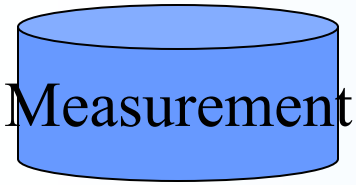
1. **The local parameters of the trajectory:** ***a vector and a covariance matrix***
    1. **Type (*enum* of different types: "main")**

        "main": vector = (x,y,tx,ty,q/p); tx,ty slopes; q/p ("q" is a signed curve)
    2. **Methods for connection with the geometry:** ***position() and direction()***

        Helps location in the geometry, compute IP, extrapolations, etc!
    3. **Methods for connection with physics:** ***momentum(), pt()***

        Very often needed

Measurement



General agreement not yet reached …

Measurement
---
type
measurement
error
{flags}
[LHCbId]

Enum types: Rvelo, TT, …

Measured value: local-r,local-u-distance
Error of the local mesured value

A Measurement (may be):

1. **A measurement of the detector associated of the track**

   Or a signature of the track

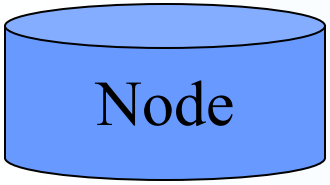
2. **Different types of measurements: Rvelo,Phivelo,TT, etc.**
3. **Has the measurement value and error of the measurement,**

   ie. u distance perpendicular to the strip in the Si strips in TT
4. **Additions (optional):**

   Flags to be use by the pattern-recognition

   LHCbID(s) to be linked to Geometry, Digits(RawBuffer), MCParticles

Node



Node
---
type
residual
error
Measurement*

Chi2()

Need for this still being actively discussed – Jose's proposal

Measurement is not own by the node

A Node:

1. **The link between the local state and a measurement**

   Has the residual, its error, and its chi2()

2. **The Node can "hide" the alignment**

   State is in a "general" frame. Measurement is in a "local" frame.

   Between State and Measurement mediate a *projection*

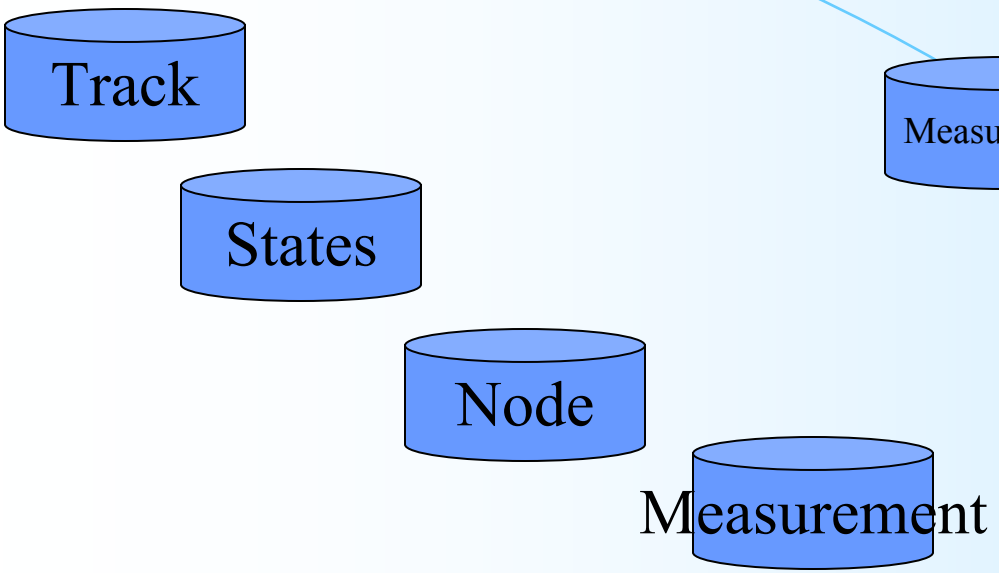
3. **The *projection* between State and Measurement computes the residual:**

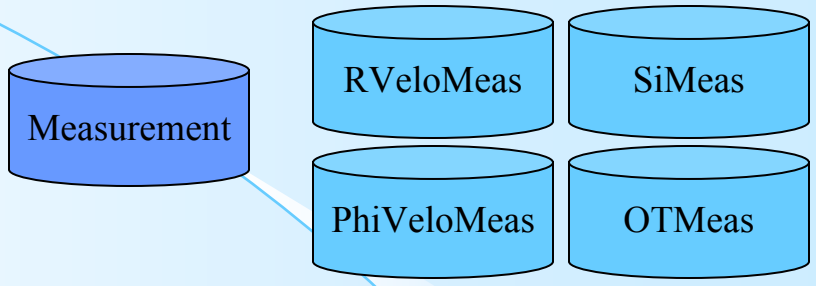
   Project the state-vector to be compared with the measurement value

   (Jose's proposal: projection as an "external" operation to the node)

Jose's original proposal at http://erodrigu.home.cern.ch/erodrigu/lhcb/tracking/event_model/jose/2004-07_proposal/index.html

A larger picture





"Tools"

Already exist

Fit(track)
Project(state,measurement)
Propagate(state,length)
Propagate(state,DetElement)

Additional Gaudi Tools:

Efficiency(track)
MCMatching(track)
retrieveDigits(track), etc…

Additional "Servers"

Smart Measurement server

LHCbID

LHCbID:

A LHCbID (personal view) is an ID for each smallest piece of the LHCb detector able to provide a measurement:

I.e each strip of TT has a unique LHCbID

(O.Callot suggested that a LHCbID can hold several LHCbIDs)

A LHCbID is a key:

- **Able to access the geometry**
- **Can be linked to the Digits (an therefore to the RawBuffer)**
- **Can be linked to a list of MCParticle (most time only one)**

With this extra links ☺, the LHCbID can be quite helpful

If any reconstruction object can provide a list of LHCbIDs …

A lot of common task can be performed at the base level:

i.e buffer-tampering!! or MC association

We can think that a LHCbID can be defined in the LHCb-framework and leave an space for it in the Measurement base class

How far in the model we go?

Trigger and Offline:

1. ***The agreement of the model Off/trigger goes fine till… Measurements***
    1. **Will use the trigger Measurements*?**
    2. **Can the trigger provide the Measurement* in the Node?**
    3. **But…we still want to give the "access" to the "Measurements" of the Node? Now we use the xxxClusterID, and (maybe) in the future LHCbID?**
2. **Measurement should be a base class for the xxxCluster**
3. **LHCbID could be a base class for xxxChannelID**

A 2 step approach

1. **1st step: go ahead, agree on Track, State base classes**
    1. **Re-code and make available Track and State**

        Note: Most of the clients of the tracking ask only for Track and State
    2. **Solve the link with "Digits" using LHCbID?**
2. **2nd step: go ahead and add Measurement as a base class of the full model**

Conclusions and questions

- Tracking BASE classes:

Track : agreement!

State : agreement!

Measurement/Node: in the way to get an agreement?

- Do we accept the full tracking data model?

De we proceed with the 2 steps?

- We should think more about the LHCbID class

- one will probably need to introduce other requirements from specific users (e.g. RICH group, muon tracks)

- The agreement implies:

1. The base class will be the input/output for the algorithm that depends on the tracking
2. The tracking "internal" code can use any other representation
3. Should be recommended that "internal" tracking tools and algorithms use the base classes.

- In short: implementation to start NOW … first version for end of year …