From Rob Kutschke

This is the first draft of a list of some of the issues that we are now encountering with the Track class.

1. I would like to be able to ask for track parameters and covariance matrices that are valid at different locations along the track. Examples of such locations include but are not limitted to:
    - the point of closest approach to the origin
    - just outside of the beam pipe
    - the innermost hit
    - at the outermost hit
    - at the footprint on the calorimeter
    - at an arbitrary hit
    - at an arbitrary surface with a missing hit
    - at an arbitrary location along the track
2. The class should be rich enough to hold several fits of the same hits, each assuming a different mass hypothesis, e, mu, pi K, proton.
    - Here is a use case: if you want to identify electrons within jets, you need to remove electrons from photon conversions. To do an optimal job of this requires that the track be fitted as an electron so that the covariance matrices used in the vertexing are optimal.
3. For point 1, how do we report the range of validity of the returned set of track parameters and covariance matrix. Ideally we should develop this code inparallel with vertexing algorithms that expliot the information.
4. For points 1) and 2), how should the track behave if the exact requested information is not available. Is it allowed to return its best approximation? If so how should that be indicated?
5. The Track class contains a getReferencePoint() member function. Within SiD people have understood the concept of reference point to have different meanings and code exists for each of the alternate meanings. These codes do not work together. Within the LCIO documentation I could only find a cursory definition and I believe that it is wrong.
6. The Track interface specifies that the track parameters and covariance matrix should be stored as floats. In my experience this is inadequate for some constrained fitting applications and they should be stored as doubles.
7. The class contains methods to access the list of hits on the track. It also needs a method to access the list of expected hits or a list of missing hits, so that one can more easily filter tracks to exclude those with improbable hit patterns.
8. The TrackerHit interface is not appropriate for stereo silicon strips: one cannot define the (x,y,z) of the hit until that hit is associated with a track. Within SiD we have a plan to go back to the precursor of the TrackerHit to get the required information. Is there a more general solution that works for all detector concepts?
9. Related to the previous point: we should distinguish the concepts of a "hit" and a "hit attached to a track", for which additional information can be computed.
10. Thinking ahead to alignment. We need a place to store residuals and the errors on the residuals. One should be able to choose if the residual is computed with the local hit included in the fit or excluded from the fit.
11. There does not appear to be a well defined way to get the momentum of the track, because the algorithm depends on both the track parameters and the magnetic field strength at the point at which the momentum is requested. Within SiD we have a crude solution in place. (Tony please correct me if the solution is not crude.). If we implement 1), then this becomes even more complicated.
12. Do we want to discuss the bookkeeping of which hits are already used on tracks and which are free for future pattern recognition. Or is this best left open since it is too difficult to anticipate the needs of all reasonable hit arbitration algorithms?