Work Packages

Draft list of Work Packages

This is a draft list of work packages that have been identified during the discussions this week. This list is not prioritized.

In the coming weeks we need to prioritize this and add names to jobs.

Some of the code already exists in people's contrib areas or on their laptops.

1. Complete the new geometry.
2. Make the changes to RawTrackerHit that are described in Summary
3. Deploy the code to create RawTrackerHits from SimTrackerHits. We need two versions of this code.
    - A fast version that is good enough for developing downstream code that must be fast but does not need high fidelity.
    - A high fidelity version that is allowed to be slow.
4. Hit collection management. These are collections that can be added to the event.
    - A structured collection of RawTrackerHits, suitable for finding clusters.
    - A structured container of clusters that link back to RawTrackerHits.
    - A structured container of crosses that link back to clusters.
5. Cluster finders for both strips and pixels.
6. Cross finders.
    - Definitely needed for forward detectors.
    - Should also work in the barrel since we will wish to evaluate stereo in the barrel.
    - Should also clusters to cross at other than right angles.
7. Geometry for forward trackers:
    - For existing devices that are described as the annulus of a disk.
        - Create code to cover the surface with a tiling of pixels or strips. But the representation as the annulus of a disk is not changed.
        - Hans calls this a "readout geometry". Norman calls it virtual segmentation.
    - Implement a more realistic geometry with overlapping planar wafers and associated support structure. Create a scheme to tile each individual wafer with strips or pixels.
8. Cheaters for creating clusters that have unbiased positions and true gaussian resolution, for both strips and pixels. Probably not necessary for crosses since I expect that crosses will be split into clusters when given to the fitters.
9. Port exsiting pattern recognition cheater code to use the new machinery. Where appropriate, extend it to work in the endcap. The code should continue to work with the old geometry, which may mean that the port is a new pacakge.
10. Port code that feeds hits to fitters to work with the new machinery. Where appropriate, extend it to work in the endcap.
11. Understand what really works and does not work in the world of track fitters and decide which one we will use as the main tool for future development.
    - Norman's TRF
    - Nick's weight matrix fitter.
    - Caroline's Kalman filter
    - Colorado code (KFitter +SOD Tracker - not sure what is what here).
12. Be ready to contribute to the trajectory class so that it implements what we need. This may end up as an extended track class instead of a new trajectory class.
13. Port existing non-cheating pattern recognition code to the new geometry (it should also continue to work with the old geometry, which may mean that the port is a new pacakge. This includes the code that is partially cheating, such as the code that cheats to get a track in the pixel detector then does real work with the strips and the code that starts with calorimeter MIP stubs and does real work with the strips.
14. Develop integrated pattern recognition that considers all vertexer and tracker hits in unison.
15. Develop pattern recognition that uses a Kalman filter to grow the track (as opposed to distinct pattern recognition and fitter steps).
16. Port ZVTOP so that it runs in org.lcsim (Jan Strube has partly done this and an earlier version ran in JAS2).
17. Port the jet tag flavor tag code developed by LCFI.
18. Develop a method to keep track of which hits are used by which track. The method should not modify objects that are already part of the event. The methods should allow a series of tracking algorithms to be run, each using the hits left over by the previous step. It should also consider how to deal with running two competing methods in the same job, both running on the full same set of hits.