JRA1 Analysis Framework

Teleconference, December 19th 2006

Possible discussion points:

- meeting in Geneva
- cvs repository
- general code structure
- geometry description
- event data structure

Meeting in Geneva

Additional JRA1 Analysis meeting is temporarily scheduled for January 26th in the morning:

- Who will participate?
- Where can we hold the meeting?
- Should we invite others (email to community)?
- Try to invite special guests (Roman Poeschl, Frank Gaede)?

cvs repository

We can use the existing ILC software server.
However, we need to decide about:

• the collection name
 current proposals are: "eutelescope", "eutrack" and "eutelana"

• list of people participating in the code developement.
Any volunteers ?

• What about collection structure?

code structure

Do we want to use the concept of data "processors", as in Marlin (Processor class; similar concept is for TSelector in root).

Each task which we want to perform on the event by event basis could be defined as class inheriting from Processor class, with predefined "entry points" for initialization, start of run, event by event analysis and final calculations ('init', 'processRun', 'processEvent' and 'end' methods in Marlin).

We would probably have to define our own processor class, to meet our configuration and data structures (?)

Marlin: from Processor.h

```
/** Called at the begin of the job before anything is read.
 * Use to initialize the processor, e.g. book histograms.      */
  virtual void init() { }

  /** Called for every run, e.g. overwrite to initialize run dependent
   *  histograms.     */
  virtual void processRunHeader( LCRunHeader* ) { }

   /** Called for every event - the working horse.      */
 virtual void processEvent( LCEvent * ) { }

  /** Called for every event - right after processEvent()
   *  has been called for all processors.
   *  Use to check processing and/or produce check plots.     */
  virtual void check( LCEvent* ) { }

  /** Called after data processing for clean up in the inverse order of the init()
   *  method so that resources allocated in the first processor also will be available
   *  for all following processors.     */
 virtual void end(){ }
```

geometry description

MySQL can easily meet our requirements for geometry data base and at the same time it can be used for simulation.

• Should we also provide the method for reading geometry from (text) file, what about writting it to the MySQL database (!?).

• How we will refer to detector/sensor data in the code.
Int_t IDs are OK if the geometry (eg. ERunConfiguration class)   is globally visible or is passed to all processors at initialization stage.

•Any comments to the proposed geometry description classes?

event data structure

If we use „processors” we should probably have one global „container” for all event data. Could look like:

```
class ETelecsopeEvent {
// Large part missing

private:
    std::vector<EDetectorEvent> m_detectorEvent;
    std::vector<ECluster>           m_cluster;

// output from tracking and fitting (still to be defined)
    std::vector<ETrackPoint>    m_trackPoint;
    std::vector<ETrack>         m_reconstructedTrack;

    UInt_t                      m_eventNumber
};
```

Raw event data stored on detector level:

```
class EDetectorEvent {
// Large part missing
private:
 std::vector<EPixelMatrix > m_rawFrames;
 std::vector<EPixel >         m_oTPixels;
 UInt_t                        m_detectorId;
 UInt_t                        m_eventNumber;
};
```

Pixel clusters are input to „global” reconstruction – stored in one list?

```
class ECluster {
// Large part missing
private:
 UInt_t                   m_clusterId;
 UInt_t                   m_detectorId;
 std::vector<EPixel > m_cluster;
};
```