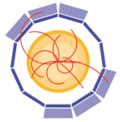
EUTelescope v1.0

Igor Rubinskiy
DESY

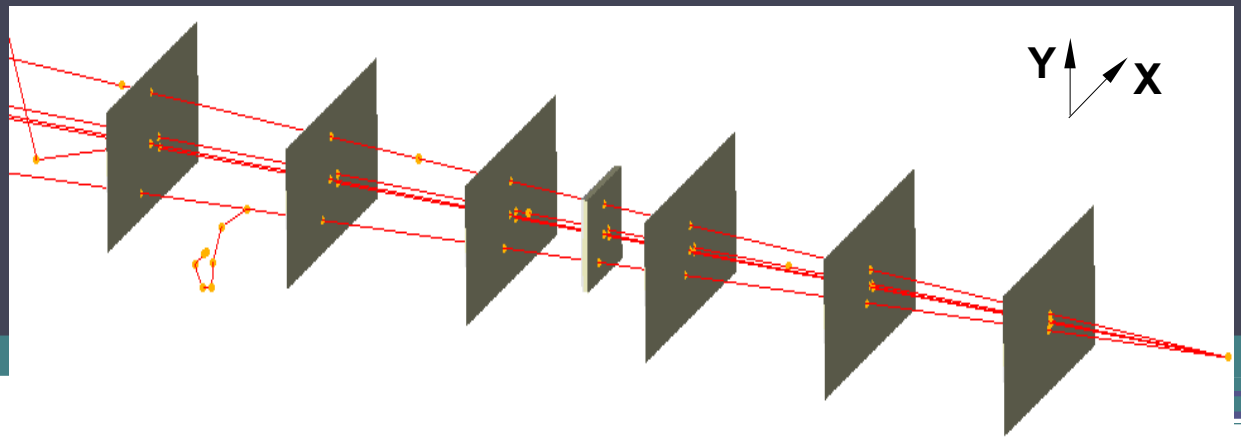
Active contributors over the last year:
Tobias Bisanz, Phillip Hamnett, Denys Lontkovskyi, Alex Morton,
Hanno Perrey, Igor Rubinskiy

Testbeam telescopes and Detector R&D workshop
DESY, 29 June – 02 July, 2014



Just a reminder: The final goal of the whole test beam

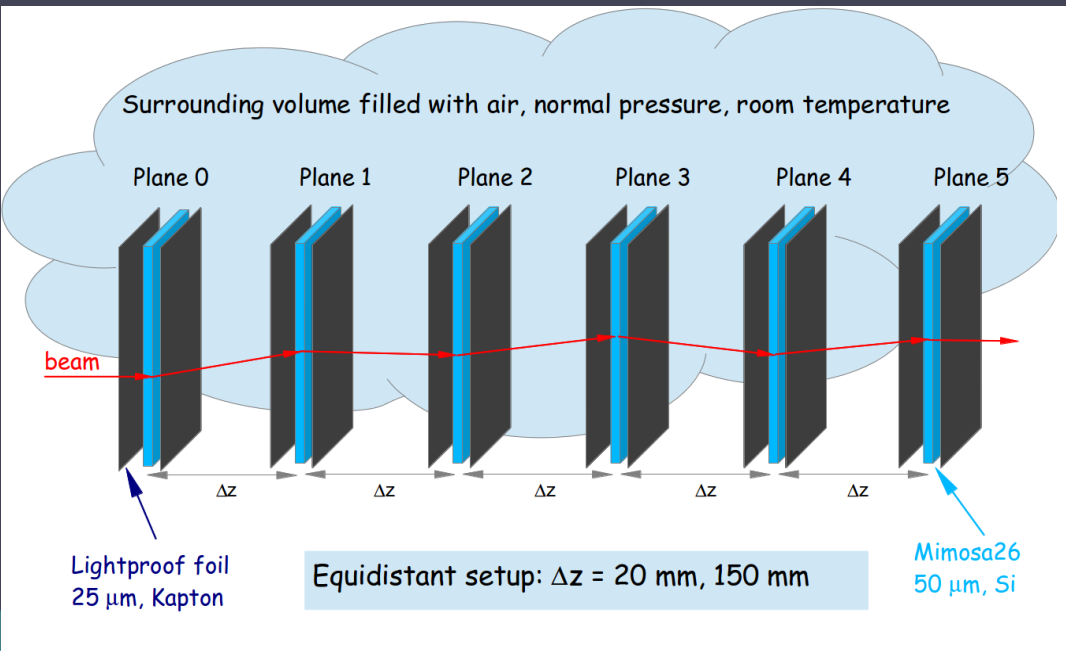
For a DUT we want to know where exactly a particle of known energy passed through the sensor
- lab measurements with sources useful for calibration, limited use, no info about incident point
- put a prototype into a test beam

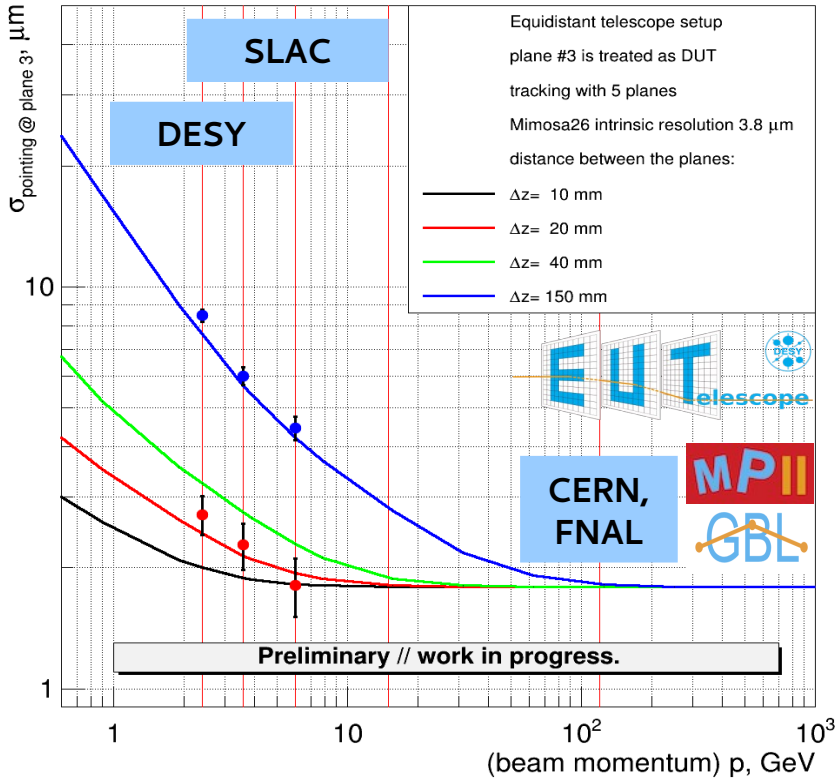


precise tracking information at the DUT

Need to get a track information for every DUT:
- X,Y in the DUT local (measurement) frame
- the track incidence angle (θ_x, θ_y)

tracking precision featuring → small pixel size & extreme thinness



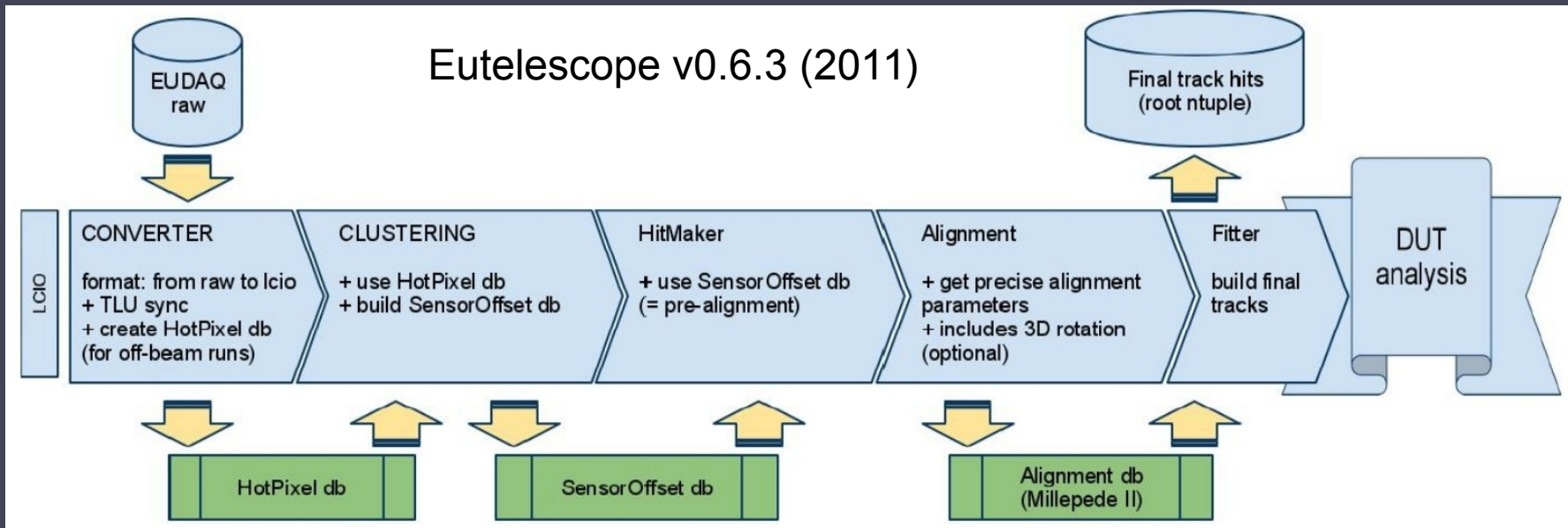


The interplay between
- the telescope detector resolution,
- multiple scattering,
- distance between telescope planes
- distance to the DUT (track fit "passive" plane)

There are more
low energy beam facilities
not mentioned on this plot

Good track pointing resolution needs
good alignment and the X_0 distribution between the measurement planes
Tracking challenges: tilted sensors, material in cooling box, BField

Few years old schematics of the EUTelescope 0.6.3 data workflow



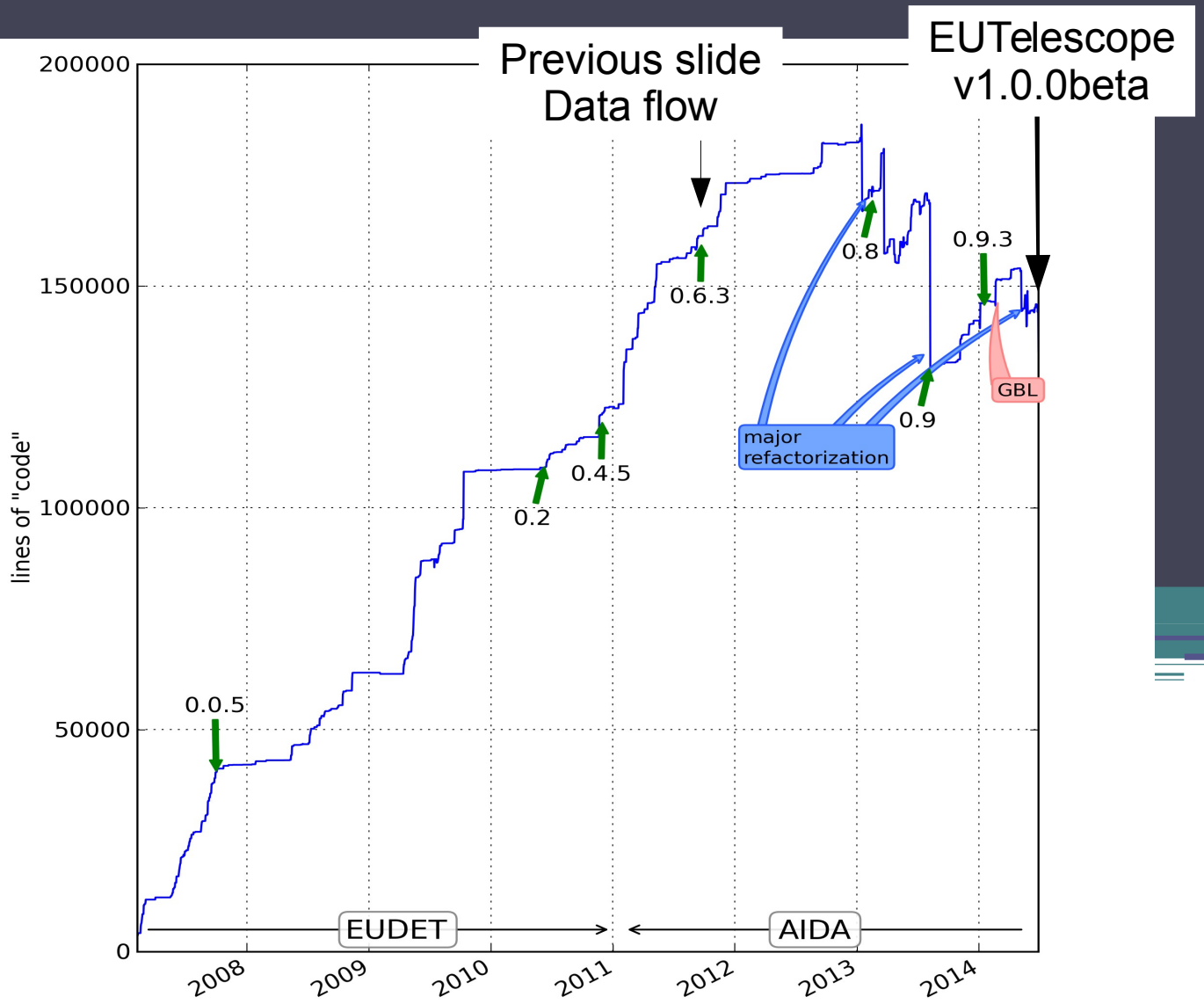
Partially still present in the ./jobsub/examples/

Issues:
1. code per processor too large/ too many parameters/ not readable any more
2. adding new functionality was becoming a challenge/ maintenance → MIP

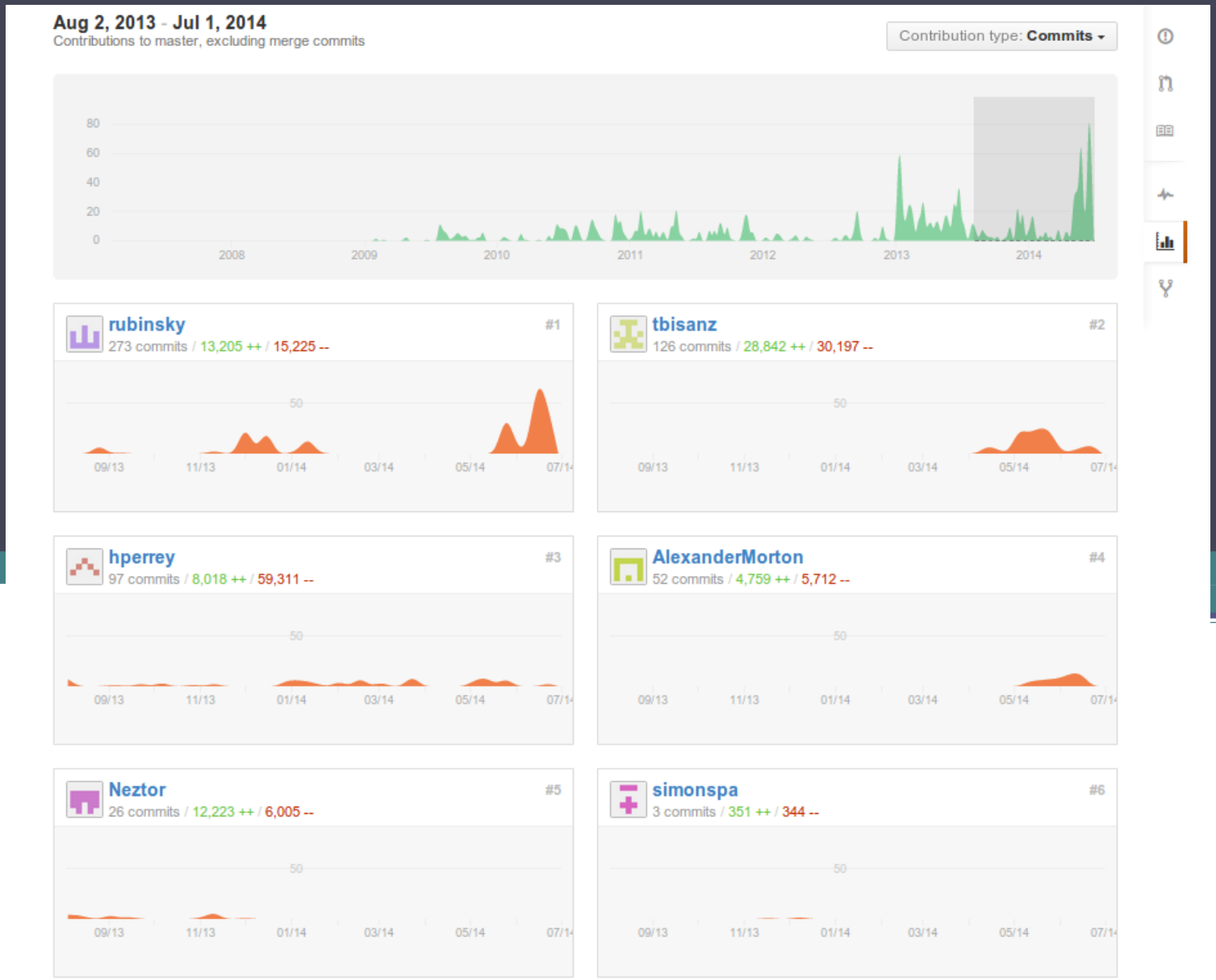
EUTelescope major refactoring over the last 2 years:
- cleanup: sorting out printout messages, resolving compiler warnings, removing unused or not documented code, naming convention(!)

New features:
– introducing new Geometry description based on TGeo, GBL+Millepede,
– revision of tracking information (track as a collection of EUTelTrackState objects)

Source code history – in lines of code



Source code history – github commits



Reconstruction Software EUTelescope

Still based on ILCSoft/Marlin framework and LCIO data format

- operation unit: data Processor to read, process, and save data in LCIO files.

- steering information is provided not via command line , or config files, but as list of Processors with certain parameters

- very useful feature of ILCSoft: all data IO is hidden

In the next few slides
– how we see the data flow should be arranged, EUTelescope v1.0.0beta tag.
See: ./jubsub/examples/datura-noDUT/
./jobsub/examples/anemone-2FEI4/

EUTelescope v1.0.0beta. Fundamental changes to the Library Architecture:

Data types:

– detectors consist of channels of EUTelGenericSparsePixel type
– short x, short y, short signal, short time
– zero suppressed, digital data stream.
– [data of all other types are reducable to this one, let us know if something is not !]

Converter:

– uses EUDAQ data decoder, puts all data into EUTelGenericSparsePixel
– build a map of hot channels (pixels)
– **EUTelNativeReader**
(Detector description classes were moved to EUDAQ!
Only EUDAQ should know what the RAW data format is)
–- outputs: data in lcio format

Hot Pixels:

– finds and maps only pixels firing above certain threshold
– **EUTelProcessorHotPixelMasker**
–- outputs: an lcio database collection (one event record)

EUTelescope v1.0.0beta. Fundamental changes to the Library Architecture:

Clustering:

– nearest neighbor search for sparsified data,

– other algorithms are nearly obsolete [pls share a good one, which is not NNS]

– **EUTelProcessorSparseClustering**

—- outputs: cluster collection(s) in a new lcio file

Filter [optional step]

– removes clusters if outside of ROI mask

—- **EUTelClusterFilter** (to be renamed → **EUTelProcessorFilterClusters**)

– marks or removes hot pixels

—- **EUTelProcessorNoisyClusterMasker & EUTelProcessorNoisyClusterRemover**

—- outputs: filtered cluster collection(s) in the same lcio file

EUTelescope v1.0.0beta. Fundamental changes to the Library Architecture:

HitMaker (for measurements)

– the name is a bit confusing, this processor returns X and Y position of the cluster in the sensor measurement frame. Actual "hit" making took place in clustering :)

– prefered: hit in local frame system (option to have hits in global frame kept)

– Geometrywize have to know the pitch and number of rows, columns. No more.

– this is the last step when one can afford to be ignorant to the layout and rotation of the detectors.

– new: Explicitly using the bit field mask for hits:

- – sensorID, [no need to guess SensorID]
- – local or global coordinate system
- – measured or fitted hit
- – contains hot/noisy pixels

EUTelescope v1.0.0beta. Fundamental changes to the Library Architecture:

EUTelGeometryTelescopeGeoDescription – derived from TGeo,

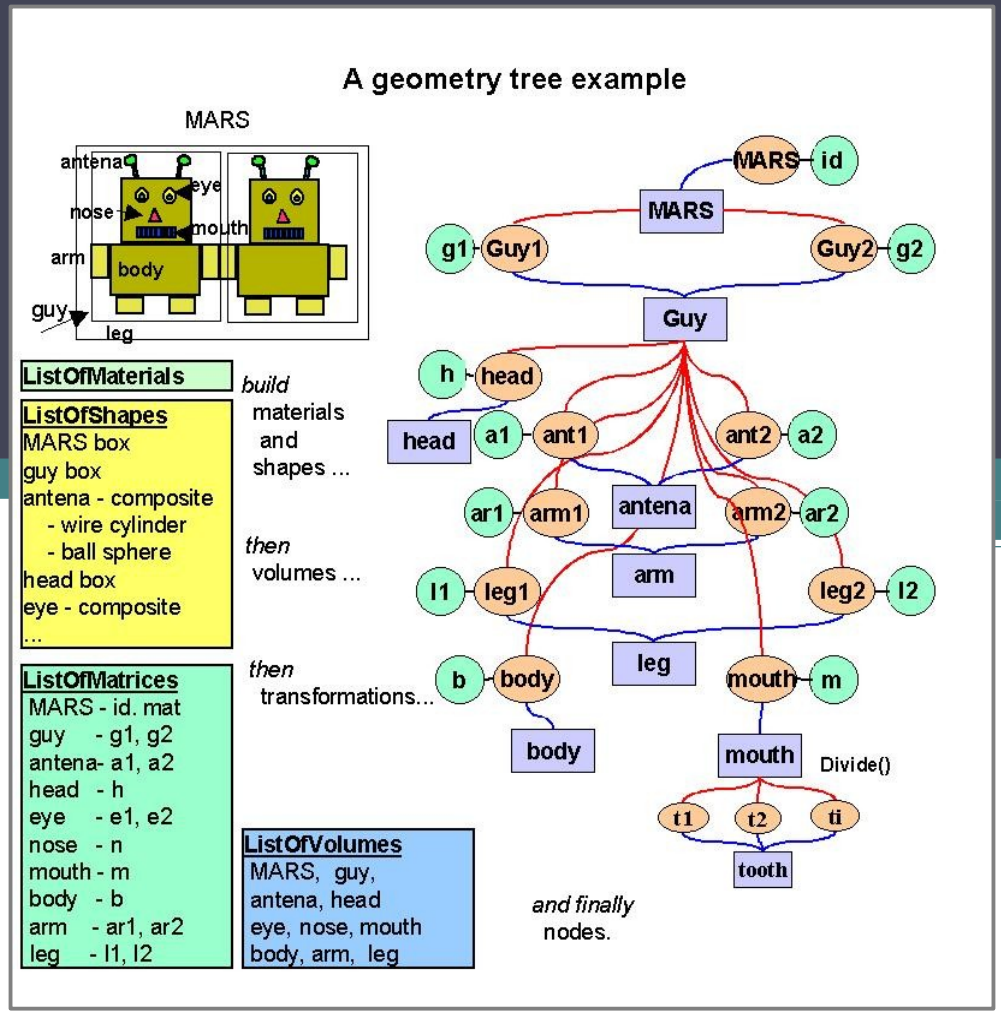
– interfaces GEAR library now, so NO calls to GEAR are allowed from the rest of the library,
– can read and write gear files, [GEAR library will become obsolete soon]
– basically a subLibrary:
    –- has to know about the "swim" direction, sensor tilts, Bfield.
    –- everything about 3D coordinates from SensoID1 to SensorID2,
    – hit coordinates transformation from Module plane to Telescope plane
– navigation methods are placed in **EUTelGeometryTelescopeGeoDescription** and **EUTelTrackStateImpl**

Hit coordinate transformation processor:
– EUTelProcessorCoordinateTransformHits
– output new hit collection, with coordinates in global/local coordinate system
    –- all geometry related methods are now hidden in
        EUTelGeometryTelescopeGeoDescription class

EUTelescope v1.0beta. Fundamental changes to the Library Architecture:

Every object is a volume, related to it's own parent volume by child-parent coordinate transformation

Can always go by the child-parent relation chain for every object, retrieve
- it's volume ID
- local and global coordinates
- vector direction



Previously used processors to do alignment and tracking are kept:

PreAlignment:
– sensor offsets from Hit correlations (**PreAlign** processor)

Alignment:
– **EUTelMille** (very dusty “spagetti” plate, being deprecated)
– with built-in straight line Pattern Recognition (fails < 3 GeV)
– can use external tracking processor for PR
– anyway Millepede needs straight line parameterisation to built input Global and Local derivatives.

Both above require ReferenceHit collection to keep track of sensor center for rotation alignment and 3D point identification (guessSensorID)

Broken line trackers:
– **EUTelTestFitter**
  – EUDET implementation of Broken Line math,
  –- can not do, tilted sensors, no Bfield

– **EUTelDafFitter**
  – implementation within EUDET
  –- can do tilted sensors, but no Bfield

Minimal use of TGeo in these processors

EUTelescope v1.0.0beta. Fundamental changes to the Library Architecture:

new data type objects:

- EUTelTrackStateImpl :: derived from LCIO::TrackStateImpl
  - it's a track parameterisation at a volume (measurement or scattering)
  - EUTel is in Cartesian system, X,Y, tx, ty, sensorID
- EUTelTrackerImpl :: derived from LCIO::TrackerImpl
  - it's a vector of EUTelTrackStateImpl

Implementation issue: EUTelTrackerImpl and EUTelTrackStateImpl can not be saved directly to LCIO files, have to rewrite the classes to derive from LCObject.

  - being done.

- all TrackState(s) are fitted hits, not required to be on sensitive plane
- the results of the PR is a EUTelTrackerImpl = track candidate vector

Pattern Recognition:

– requires a motion library, hit-to-hit, from SensoID1 to SensorID2,
– has to know about the "swim" direction, sensor tilts, Bfield.

(re)-Implemented in a KalmanFilter processor:

– **EUTelProcessorTrackingHelixTrackSearch & EUTelMagneticFieldFinder**

Tracking and Alignment

– GBL library which has a built-in interface to Millepede library
– **EUTelProcessorTrackingGBL...xxx**
    – build GBL trajectories and fits
    – outputs: TrackStates on all planes including DUTs

Currently working on:

    – can Millepede use residuals from broken line fit or it has to be parameteric?
    – track parameterisation in Bfield, with a limit B → 0.

Summary

EUTelescope has gone through the major rewrite of the reconstruction steps

Release v1.0.0.beta has been tagged, and available for testers. Please volonteer to try it out with your old and well known data!

More test beam data and use-examples are needed to test different code paths.

– everyone is welcome to contribute to make your own code better

EUTelescope v1.0beta. Fundamental changes to the Library Architecture:

A fitted point by origin and information it contains is completely different measurement point

Measurement point = cluster center, comes from the sensor design. And does not know anything about other sensors.

Fitted point – a point on a sensor surface (X,Y) with a direction vector associated to it, and a trajectory kink angle – depends on all measured points on this track candidate. Loose relation to the measurement points through their covariance matrix.

So these objects are of two completely different types:

TrackerHit: SensorID, X, Y [measurement]
TrackState: X, Y, Z, tx, ty, Curvature (~ 1/p) – can be defined for volumes without SensorID

So our new definition of a track is – it a vector of TrackStates with chi2, ndf,