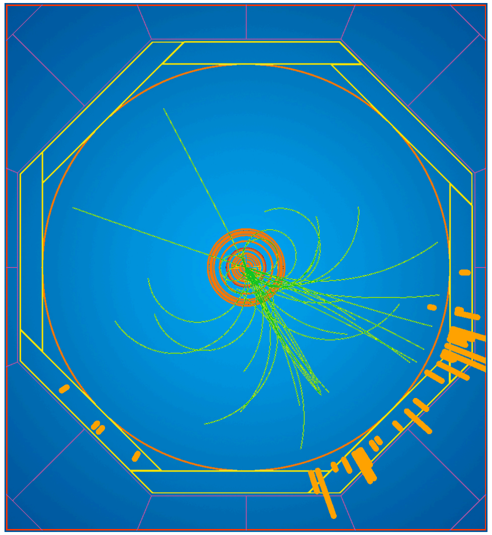
ILD Tracking – Framework Status

Steve Aplin

DESY

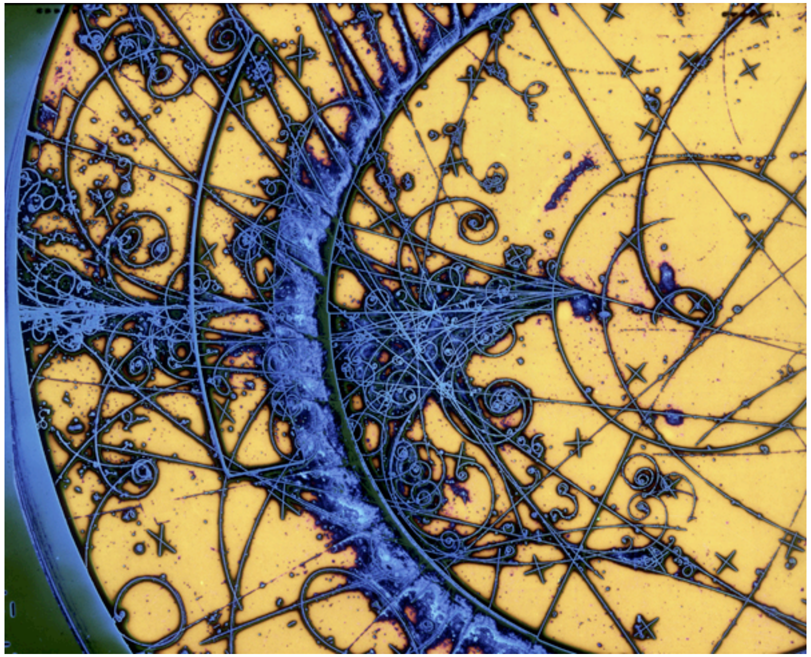
ILD Software Meeting

20th July 2011

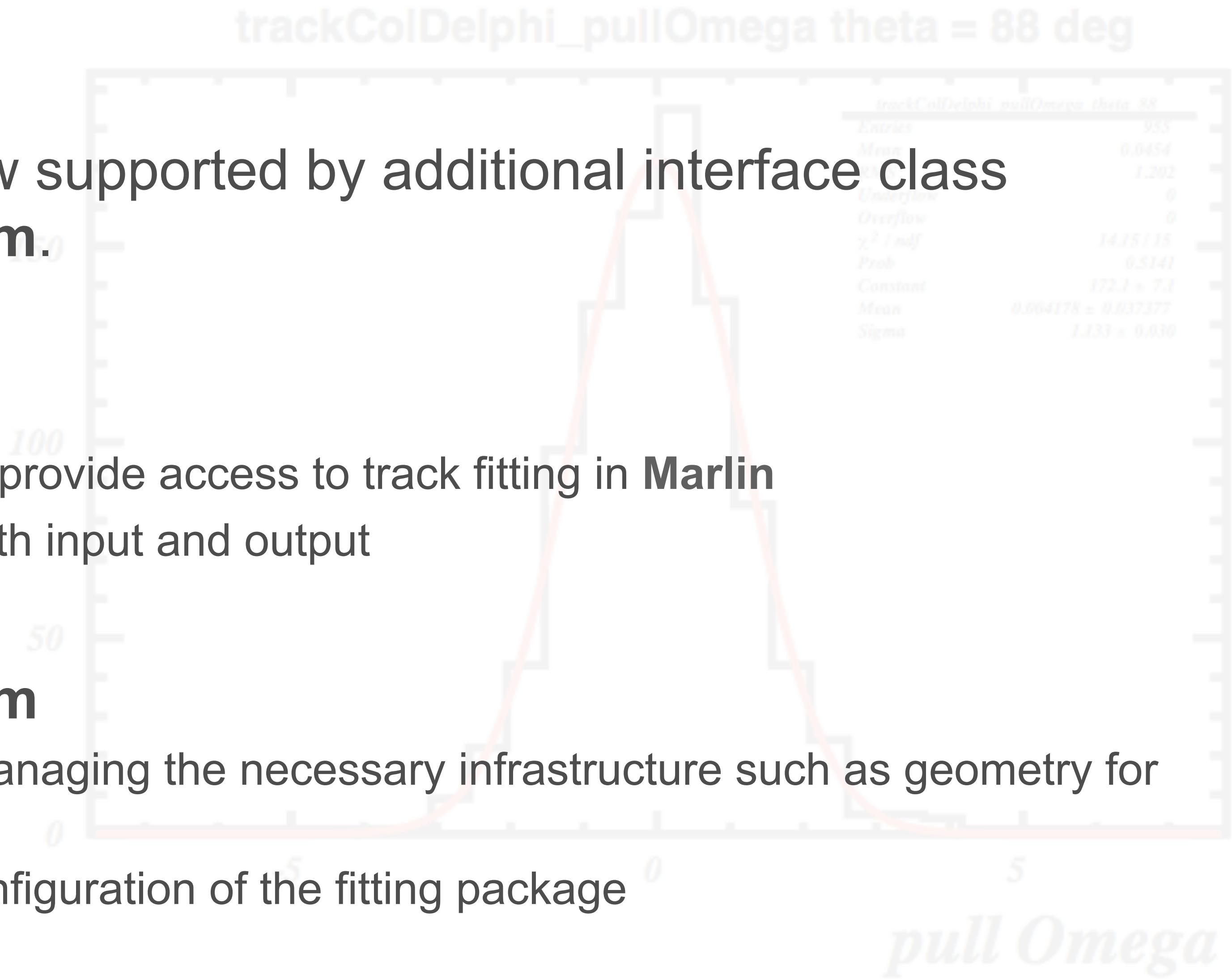


- Current Status

- Plans



IMarlinTrack and IMarlinTrkSystem

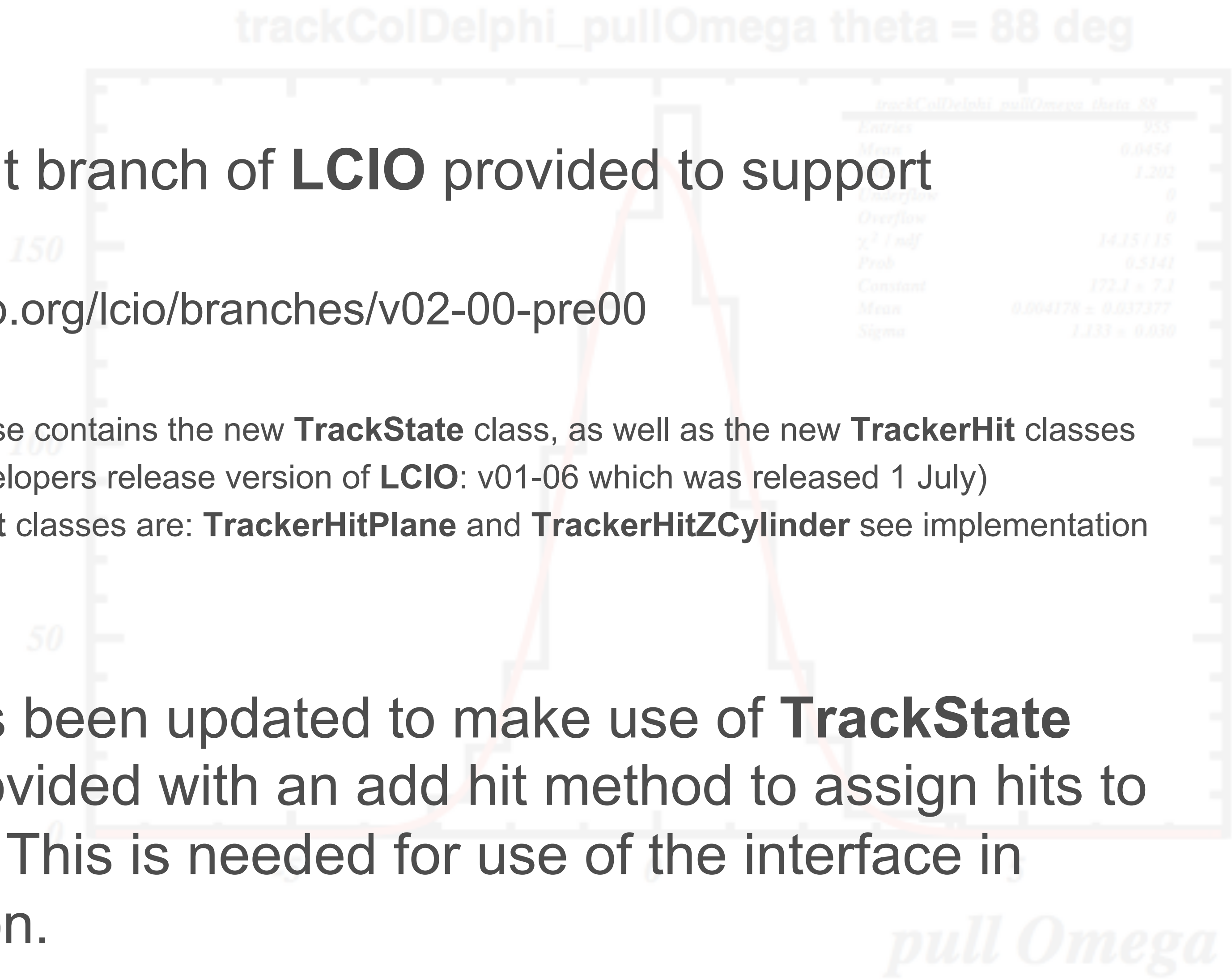


- **IMarlinTrack** now supported by additional interface class **IMarlinTrkSystem**.

- **IMarlinTrack**
  - Interface class to provide access to track fitting in **Marlin**
  - Uses **LCIO** for both input and output

- **IMarlinTrkSystem**
  - responsible for managing the necessary infrastructure such as geometry for the track fitting
  - controlling the configuration of the fitting package

IMarlinTrack and IMarlinTrkSystem



- New development branch of **LCIO** provided to support development:

  svn://svn.freehep.org/lcio/branches/v02-00-pre00

  - This development release contains the new **TrackState** class, as well as the new **TrackerHit** classes
  - (note this is not the developers release version of **LCIO**: v01-06 which was released 1 July)
  - The two new **TrackerHit** classes are: **TrackerHitPlane** and **TrackerHitZCylinder** see implementation for details.

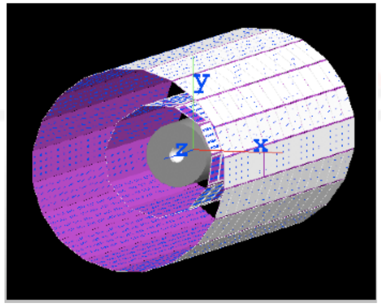
- **IMarlinTrack** has been updated to make use of **TrackState** and has been provided with an add hit method to assign hits to the current track. This is needed for use of the interface in pattern recognition.

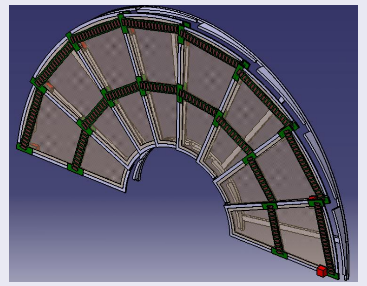
Navigation

- Welcome developments regarding the treatment of bounded and rotated planes have been added to **KalTest**, see later talk by Daisuke.

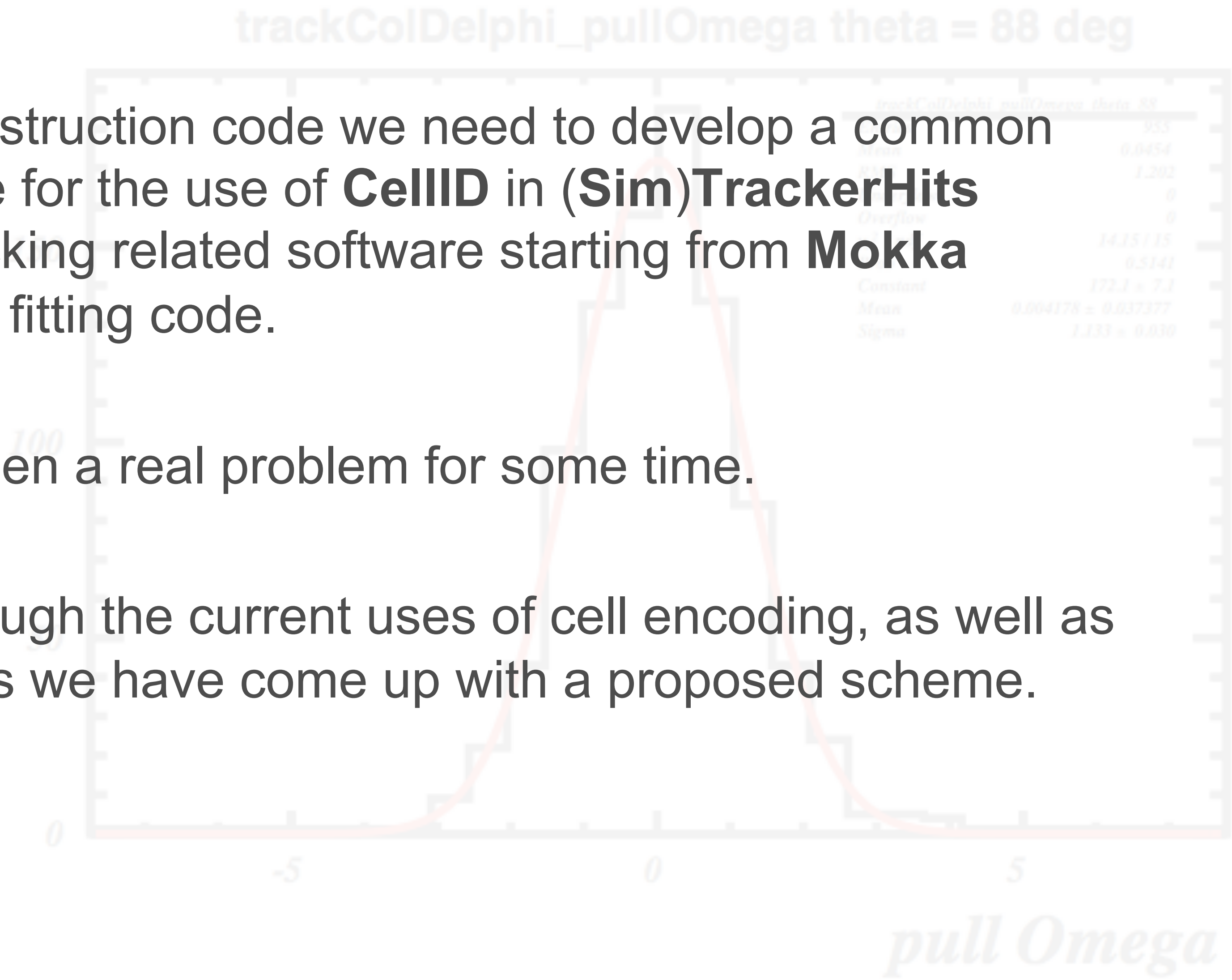
- This means that in principle we can treat both the **VXD**, **SIT**, **SET** and **FTD** designs correctly. Not yet clear for the **ETD**.

- We will have to study the performance to determine if any simplification or optimisation is still needed.





Cell ID Numbering

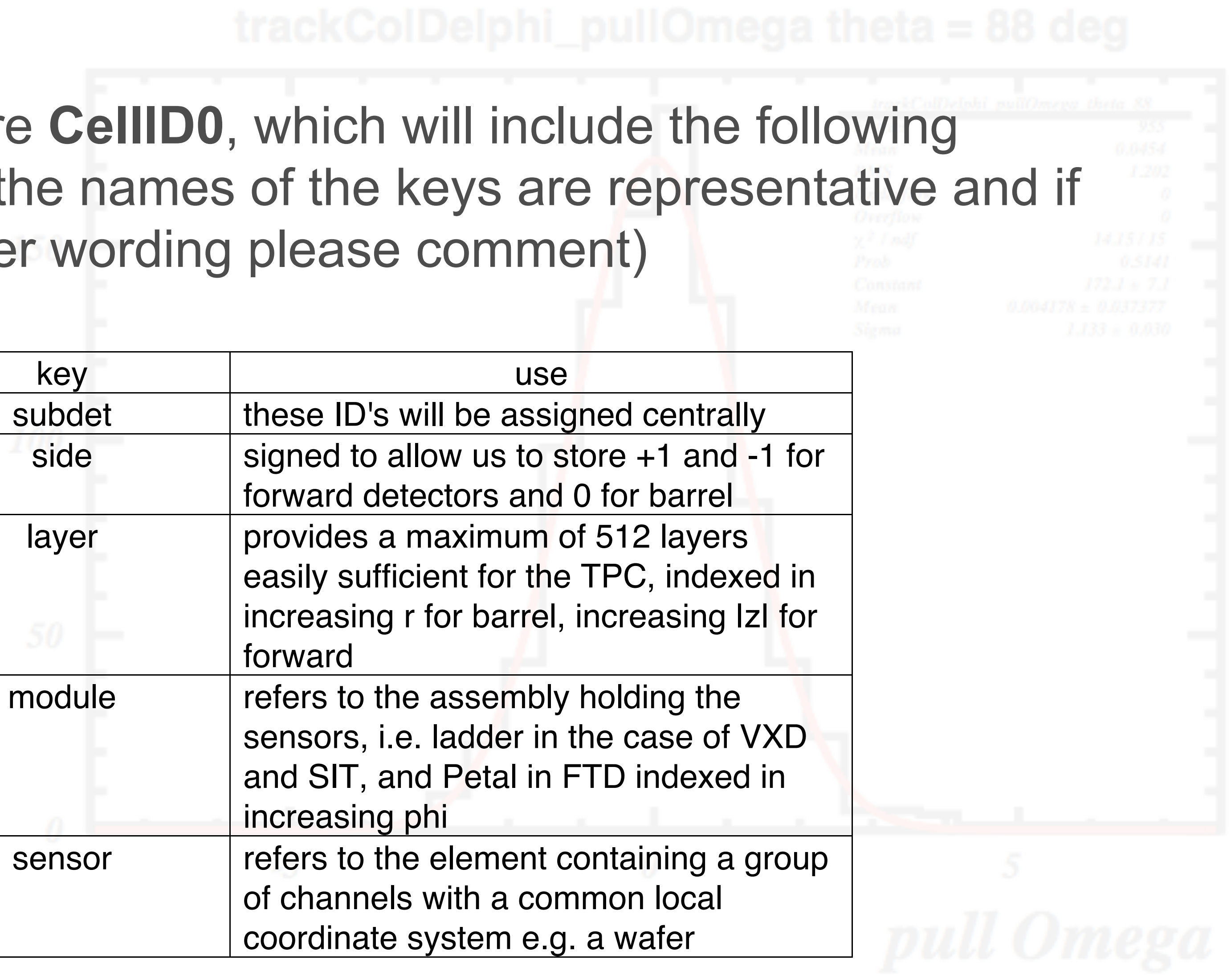


- For the DBD reconstruction code we need to develop a common numbering scheme for the use of **CellID** in (**Sim**)**TrackerHits** throughout the tracking related software starting from **Mokka** through to the final fitting code.

- Frankly this has been a real problem for some time.

- Having looked through the current uses of cell encoding, as well as several discussions we have come up with a proposed scheme.

Cell ID Numbering



- We will always store **CellID0**, which will include the following mandatory fields: (the names of the keys are representative and if you can find a better wording please comment)

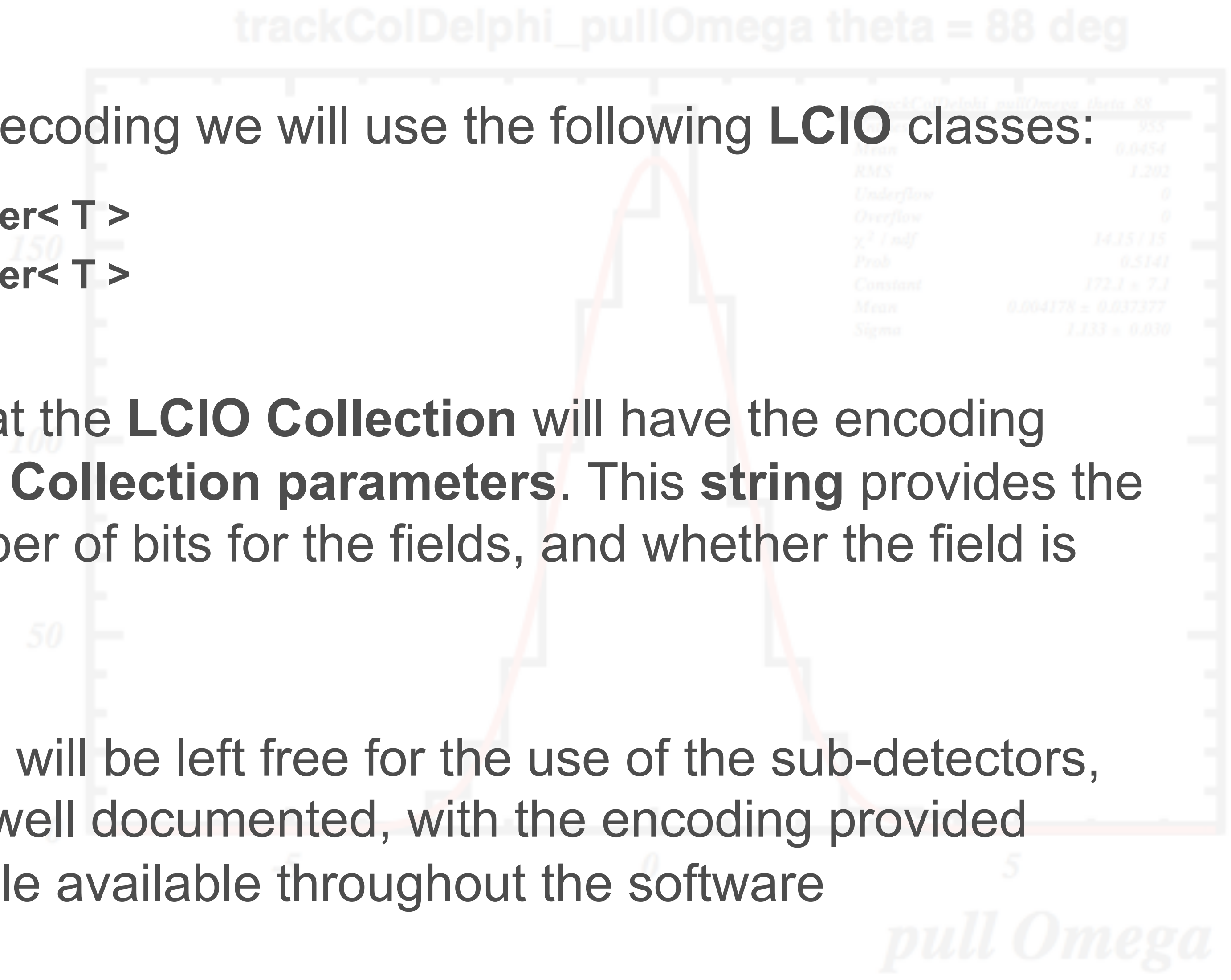
nbits	key	use
5	subdet	these ID's will be assigned centrally
2	side	signed to allow us to store +1 and -1 for forward detectors and 0 for barrel
9	layer	provides a maximum of 512 layers easily sufficient for the TPC, indexed in increasing r for barrel, increasing lzl for forward
8	module	refers to the assembly holding the sensors, i.e. ladder in the case of VXD and SIT, and Petal in FTD indexed in increasing phi
8	sensor	refers to the element containing a group of channels with a common local coordinate system e.g. a wafer

Cell ID Numbering

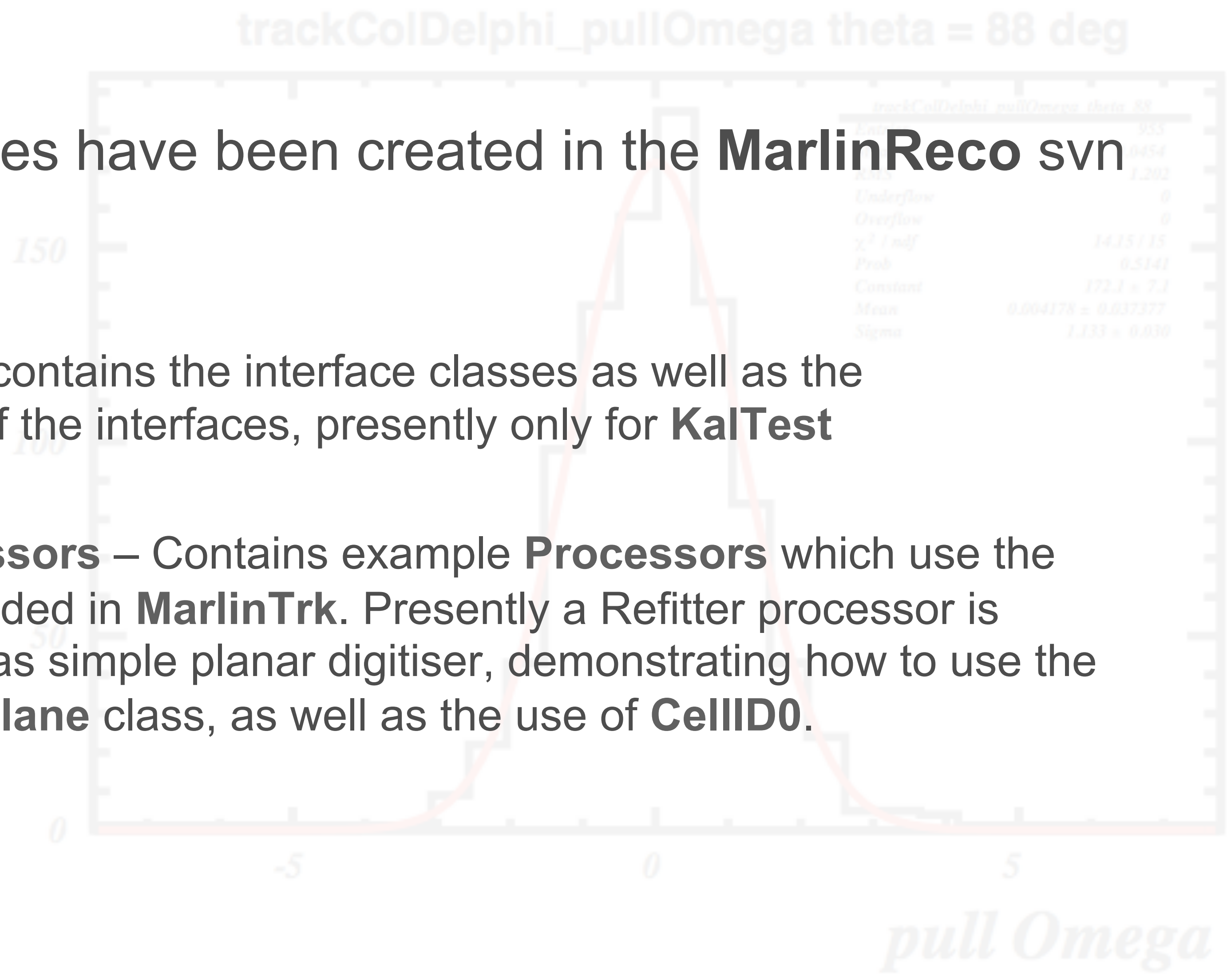
- For the encoding/decoding we will use the following **LCIO** classes:
  - **UTIL::CellIDDecoder< T >**
  - **UTIL::CellIDEncoder< T >**

- This will ensure that the **LCIO Collection** will have the encoding string added to the **Collection parameters**. This **string** provides the **keys** and the number of bits for the fields, and whether the field is signed.

- The use of **CellID1** will be left free for the use of the sub-detectors, but this should be well documented, with the encoding provided through a header file available throughout the software infrastructure.

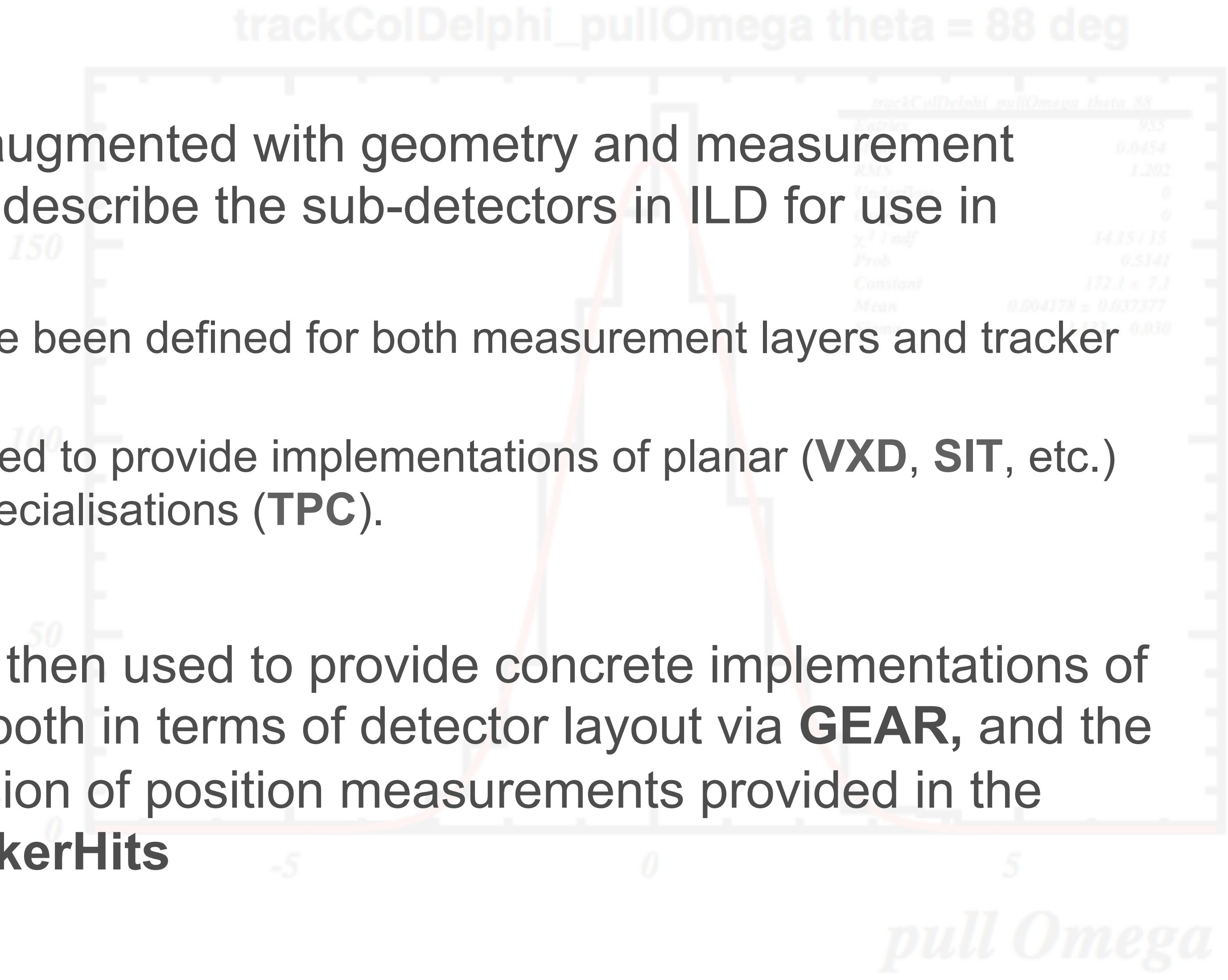


Marlin and KalTest



- Two new packages have been created in the **MarlinReco** svn repository:
  - **MarlinTrk** – this contains the interface classes as well as the implementation of the interfaces, presently only for **KalTest**
  - **MarlinTrkProcessors** – Contains example **Processors** which use the functionality provided in **MarlinTrk**. Presently a Refitter processor is provided as well as simple planar digitiser, demonstrating how to use the new **TrackerHitPlane** class, as well as the use of **CellID0**.

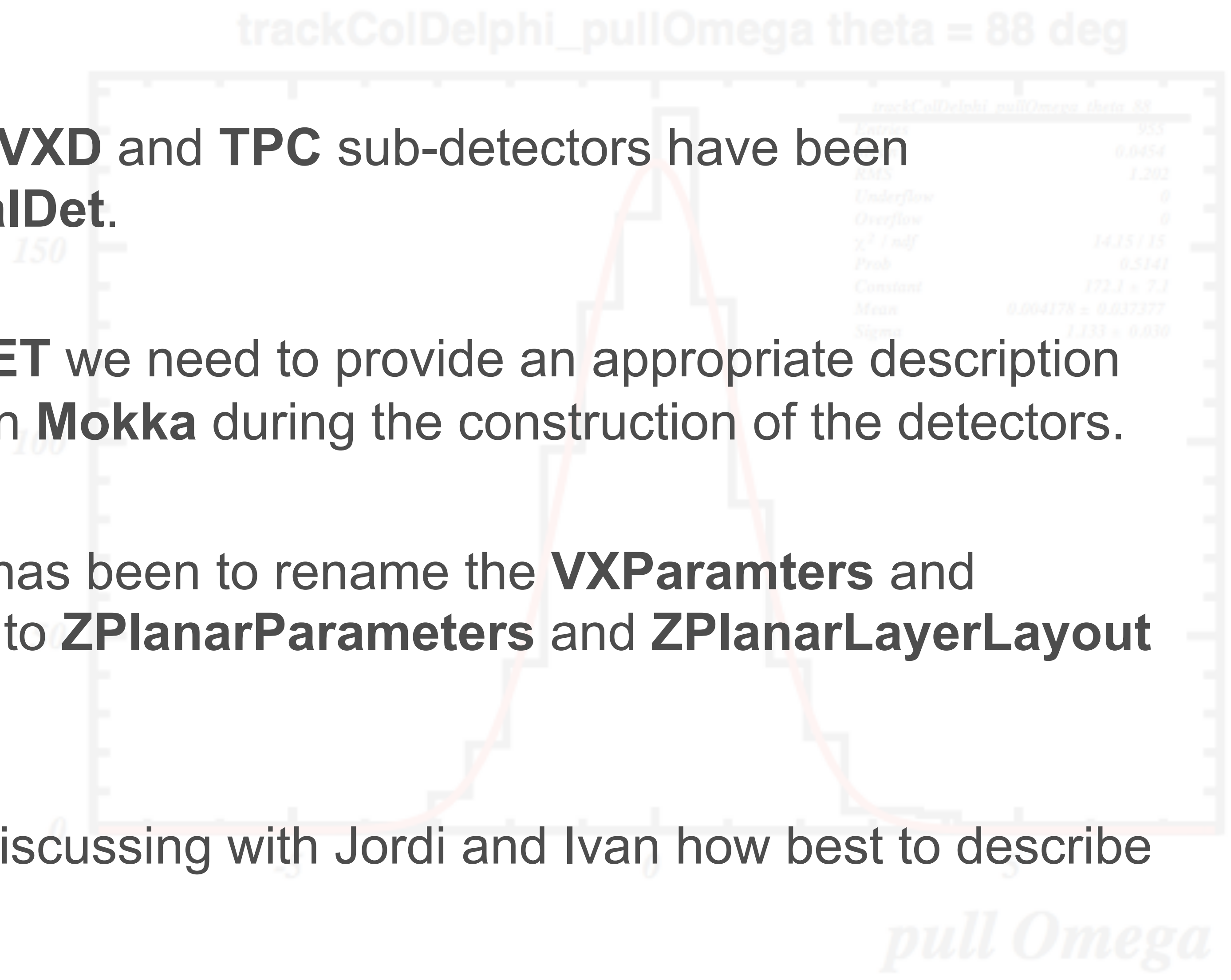
Marlin and KalTest



- **KalDet** has been augmented with geometry and measurement classes needed to describe the sub-detectors in ILD for use in **KalTest**
  - base classes have been defined for both measurement layers and tracker hits.
  - these are then used to provide implementations of planar (**VXD**, **SIT**, etc.) and cylindrical specialisations (**TPC**).

- These classes are then used to provide concrete implementations of the sub-detectors both in terms of detector layout via **GEAR,** and the necessary conversion of position measurements provided in the form of **LCIO TrackerHits**

Marlin and KalTest



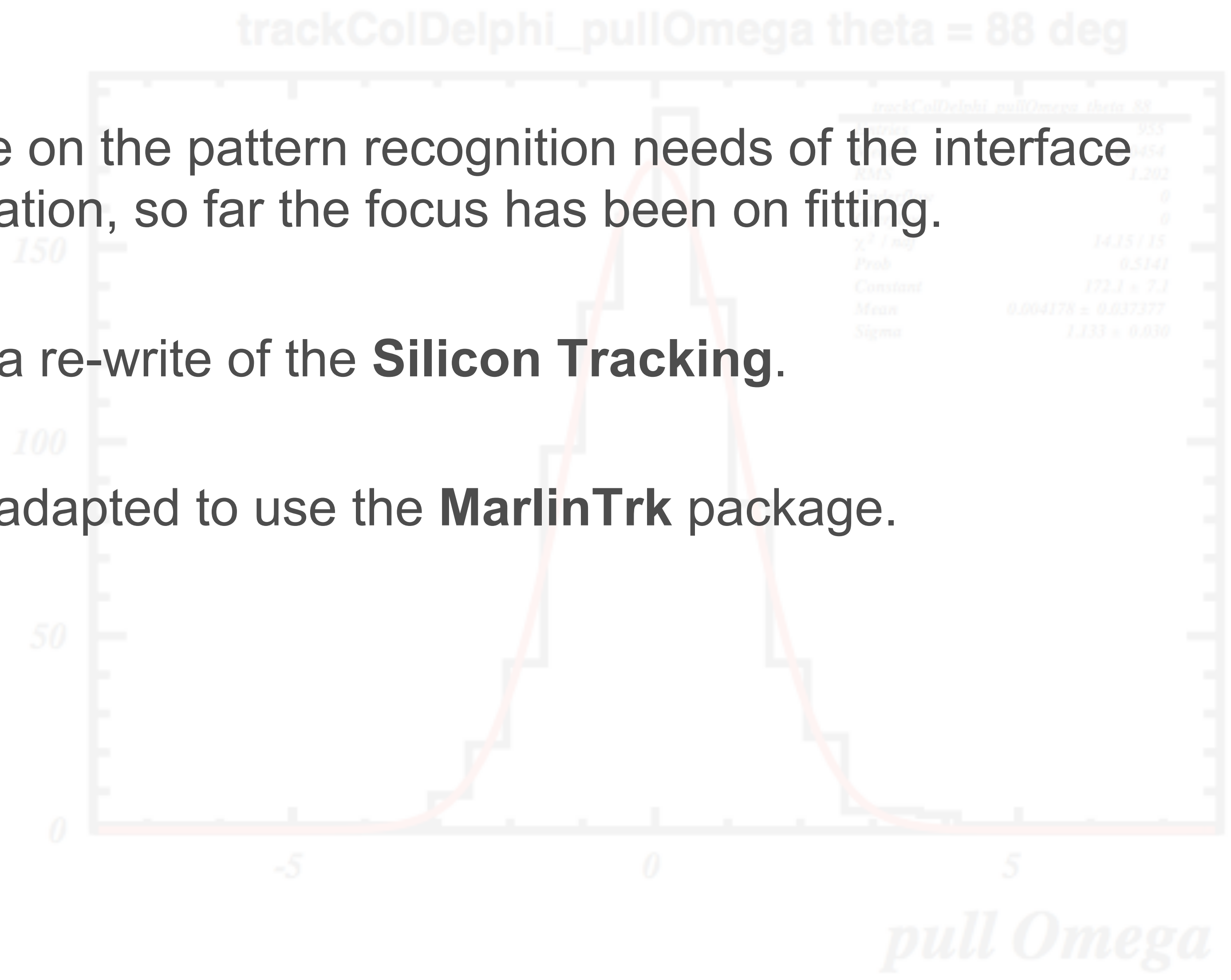
- Currently only the **VXD** and **TPC** sub-detectors have been implemented in **KalDet**.

- For the **SIT** and **SET** we need to provide an appropriate description and write this out in **Mokka** during the construction of the detectors.

- A first shot at this has been to rename the **VXParamters** and **VXDLayerLayout** to **ZPlanarParameters** and **ZPlanarLayerLayout** in **GEAR**.

- We are currently discussing with Jordi and Ivan how best to describe the **FTD**.

Plans

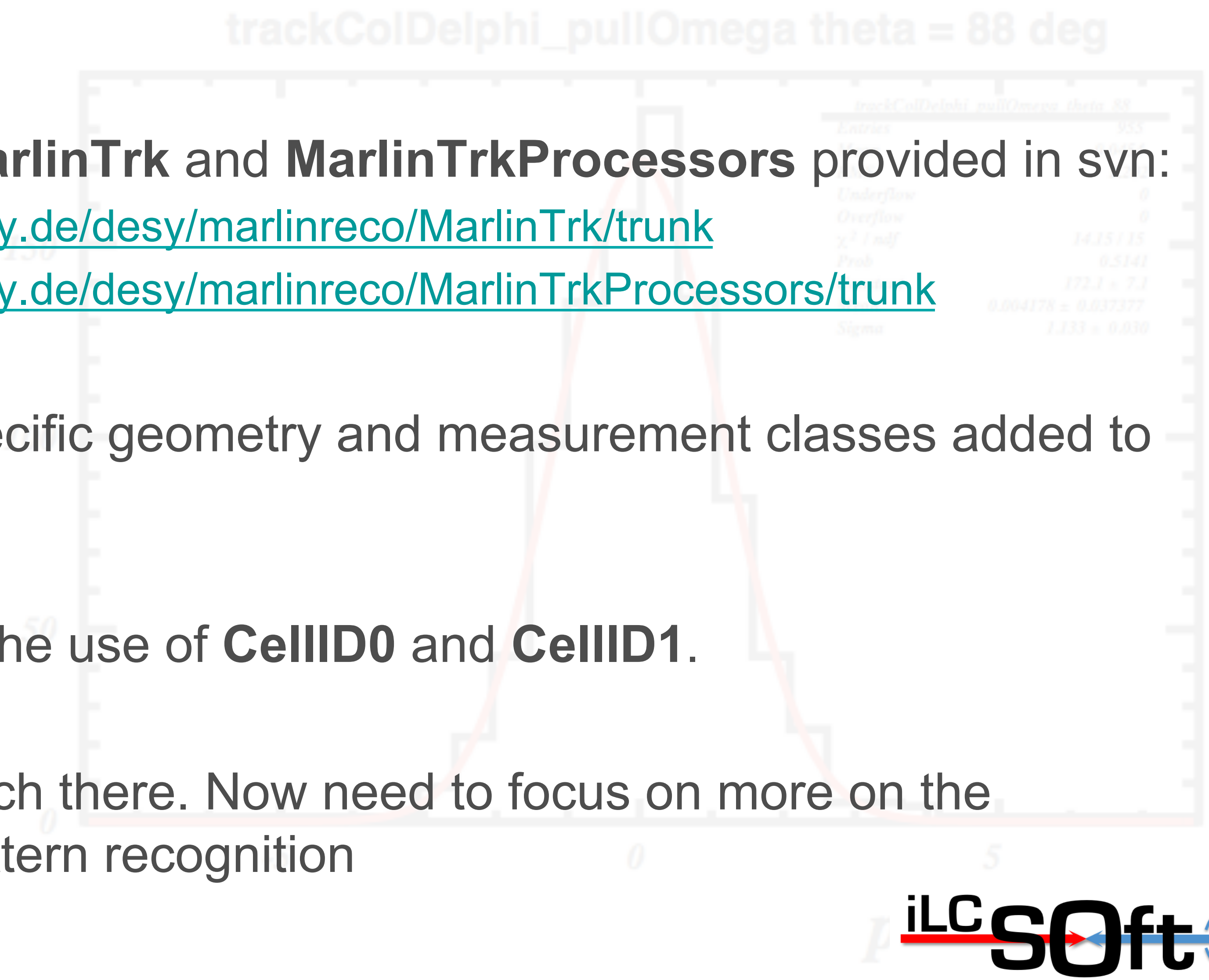


- Start to focus more on the pattern recognition needs of the interface and it’s implementation, so far the focus has been on fitting.

- This will start with a re-write of the **Silicon Tracking**.

- **Clupatra** is being adapted to use the **MarlinTrk** package.

Summary



- First version for **MarlinTrk** and **MarlinTrkProcessors** provided in svn:
  - https://svnsrv.desy.de/desy/marlinreco/MarlinTrk/trunk
  - https://svnsrv.desy.de/desy/marlinreco/MarlinTrkProcessors/trunk
- Addition of ILD specific geometry and measurement classes added to **KalDet**.
- Need to agree on the use of **CellID0** and **CellID1**.
- Fitting is pretty much there. Now need to focus on more on the functionality for pattern recognition