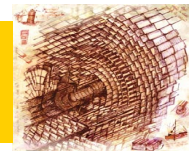
Test-Beam Software

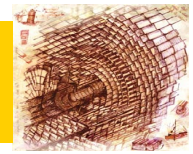
Vitaliano Ciulli

Goals



- **Test-Beam software development**
- **Test-Beam data analysis**
- **Simulation**
- **Design and operation of future test-beam**

ApvAnalysis

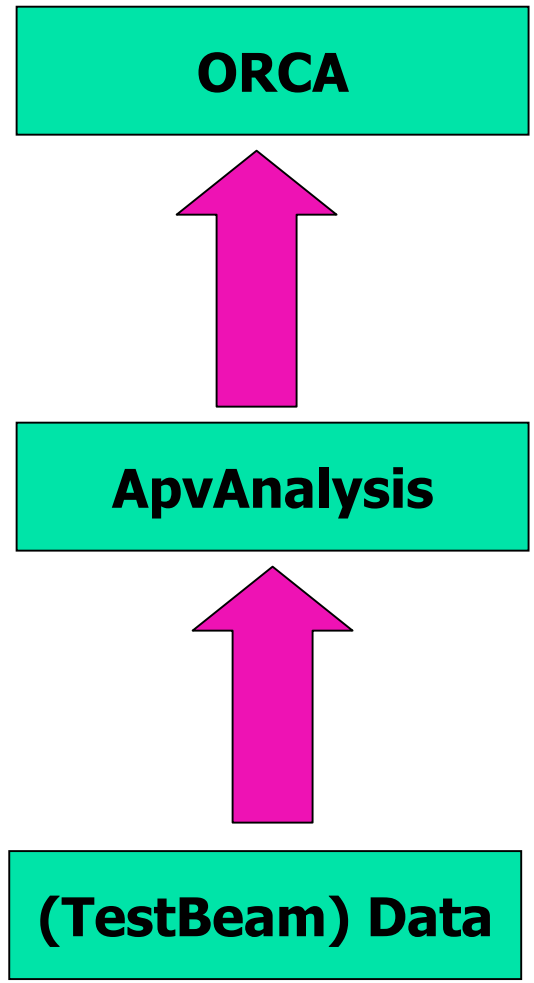


- **ApvAnalysis is a framework designed to study the various algorithms to be implemented in the APVs and in the FEDs.**
- **It has been designed having in mind:**
  - **Modularity: each single part can be easily changed/tested without touching the rest**
  - **Integration with ORCA on one side and the existing TestBeam software on the other**
  - **Even if concrete implementations are present, they are thought only as working examples and will be replaced**

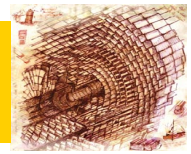
As an example we want to implement TT6 algorithms

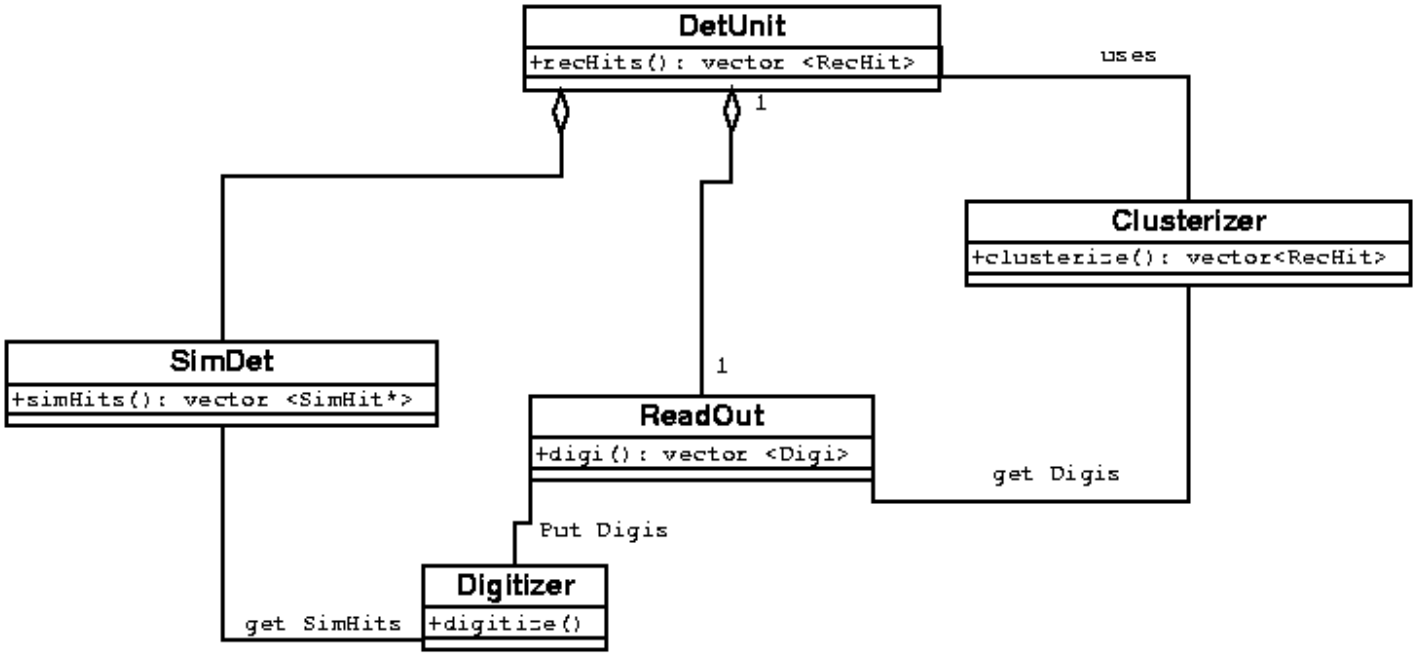
The data path

- Different types of data can be introduced in the framework:
  - Already zero suppressed
  - Raw
  - Common mode subtracted
- These data can be passed to ORCA and treated as regular digis; thus RecHits can be built and the whole reconstruction framework can be used:
  - Tracking
  - (Mis)Alignment



From ApvAnalysis to ORCA



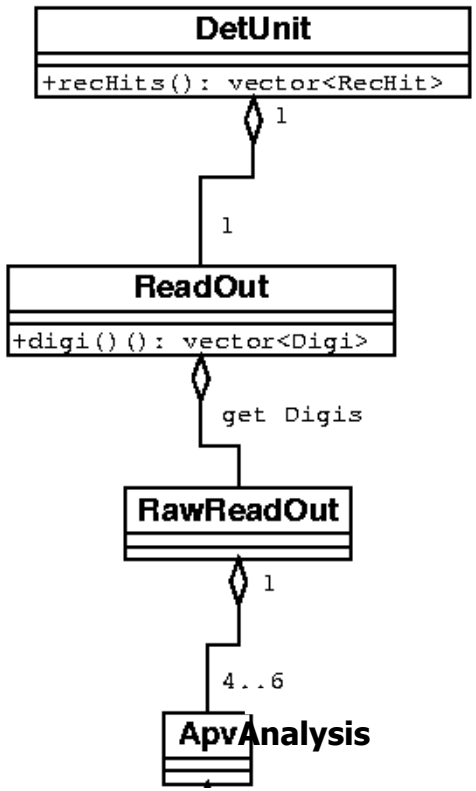


Flow for simulated data

The current granularity at ORCA level is the DetUnit, which is connected to a SimDet.

When Digis are asked, the ReadOut unit obtains them through the Digitizer.

From ApvAnalysis to ORCA (2)



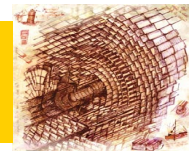
When digis are instead asked in data runs, the ReadOut unit connects to a RawReadout unit which in turn has from 4 to 6 APV connected. Thus, the granularity level is now the APV instead of the DetUnit.

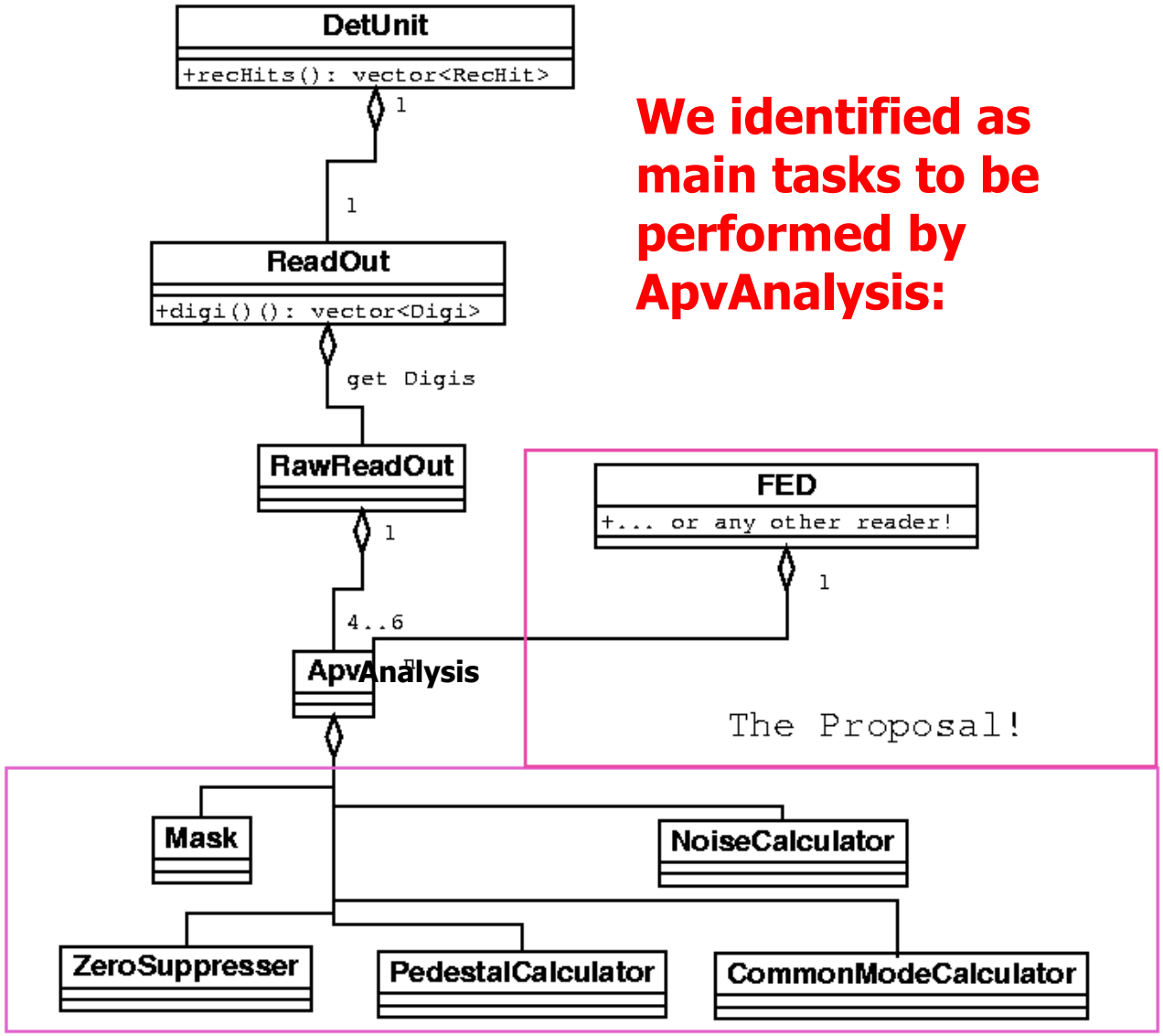
This finer granularity should be switched off by default, for speed reasons.

ApvAnalysis should return directly digis, to make the rest of the reconstruction transparent.

This part is not coded yet, but well understood on paper (Teddy)

From data to ApvAnalysis

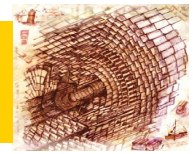




We identified as main tasks to be performed by ApvAnalysis:

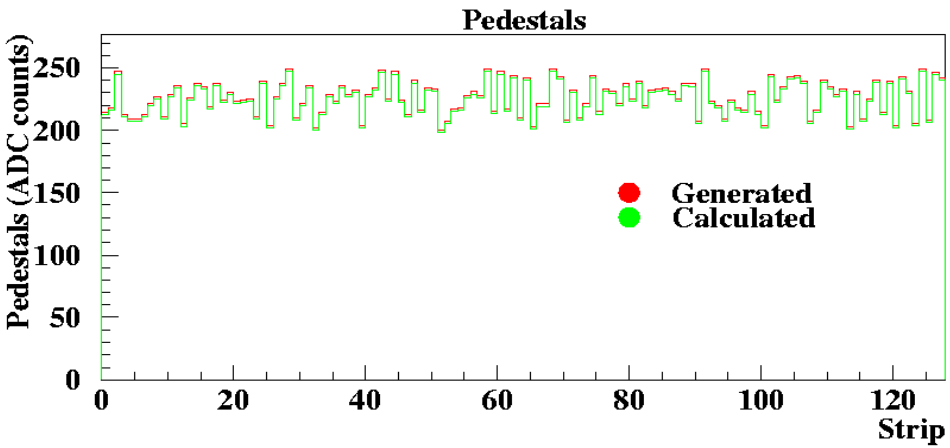
- **Access to data (tape, COBRA, FED …)**
- **Calculate/subtract pedestals and CM**
- **Mask dead/noisy strips**
- **Calculate the strip noise**
- **Zero suppress the results**
- **Produce digis**

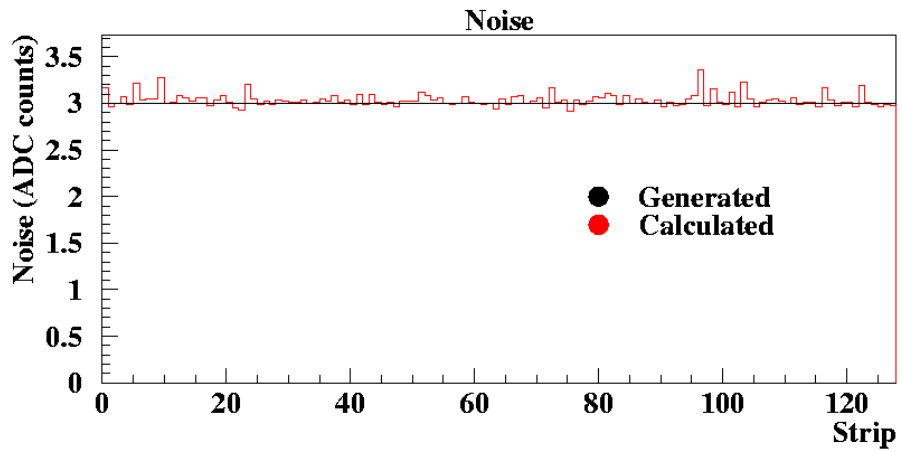
Test Programs

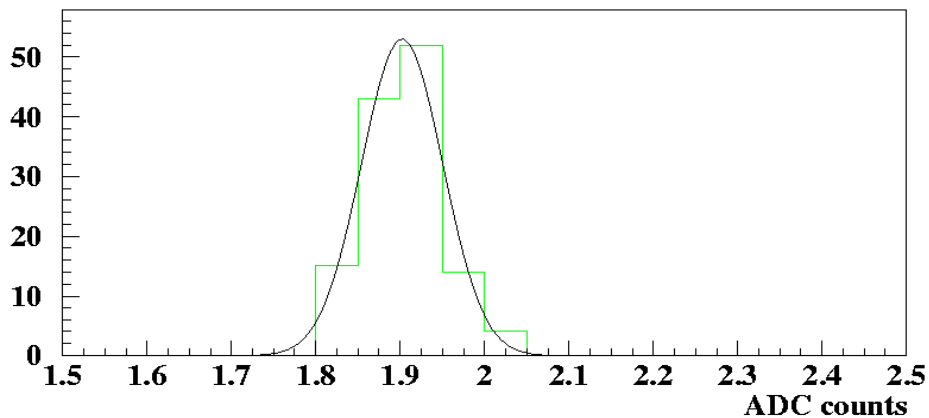


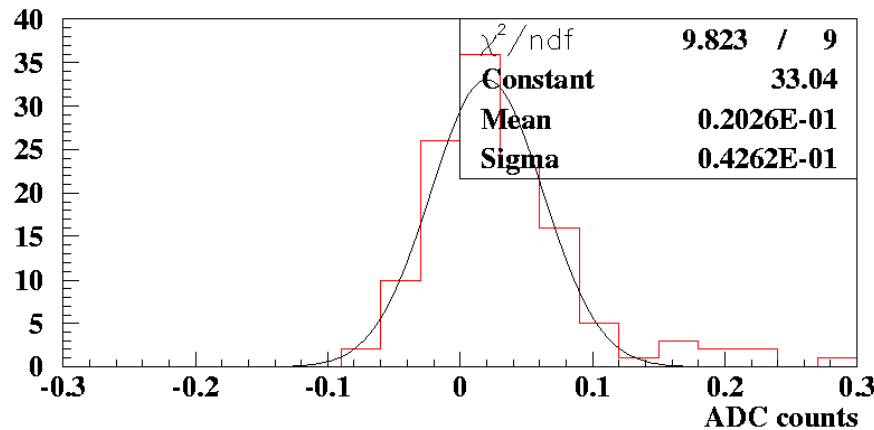
- **Skeleton.cpp**
  - **Uses the RandomEventReader to simulate a calibration run followed by a real data taking run (single APV) with single strips and clusters fired. The agreement between the various component of the raw signal (pedestals, signal, signal) is checked.**
- **SkeletonZebra.cpp**
  - **Uses the data from October X5 TestBeam. A ntuple is filled with the signal at various stages, from raw to zero suppressed signal.**

Some plots with the RandomApvEventReader





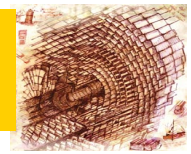




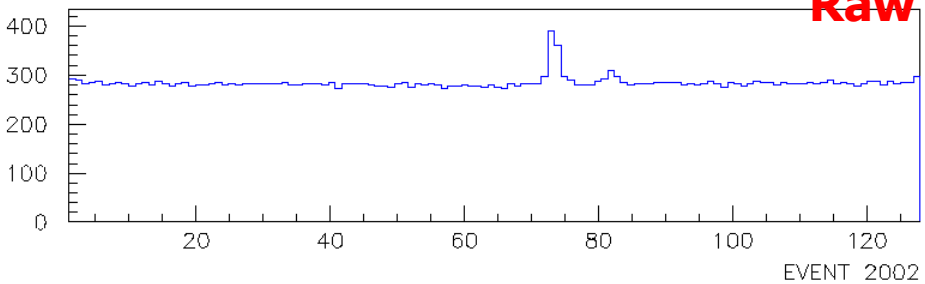
Bias due to a not clear separation between pedestal and CM…

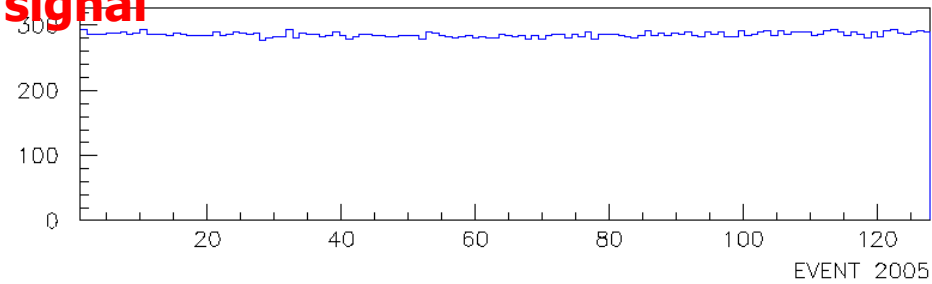
… but the noise is centered in 0!

Some plots from the TestBeam

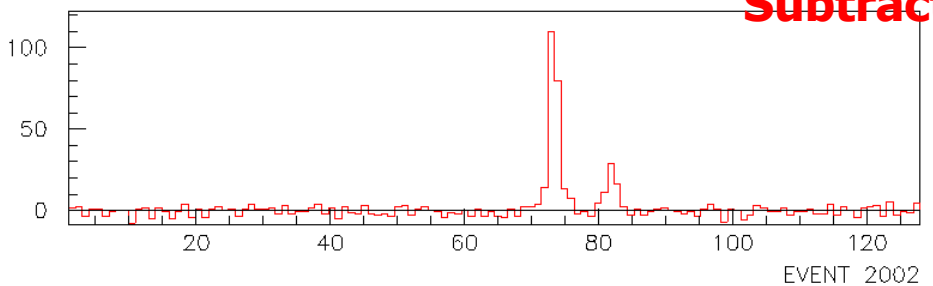


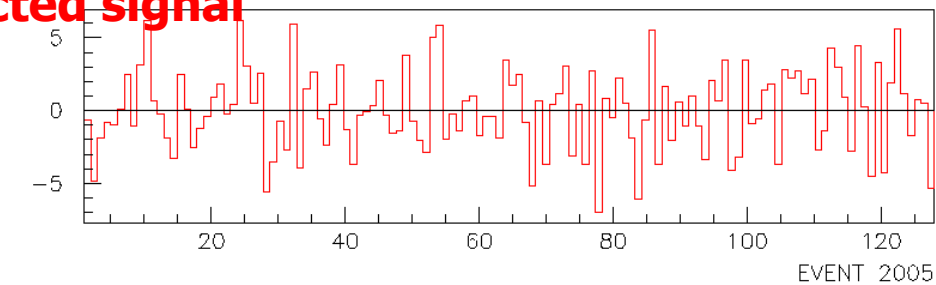
Raw signal



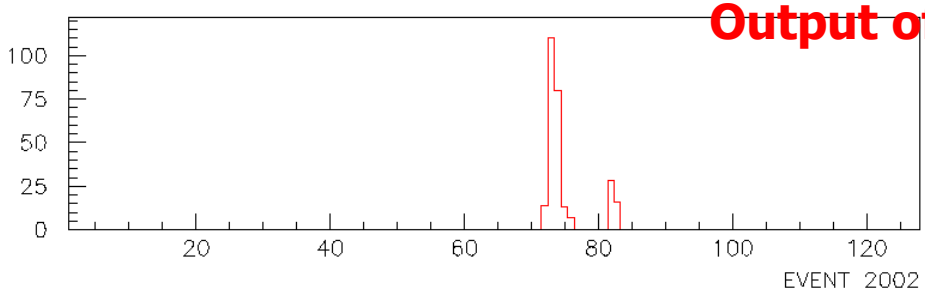


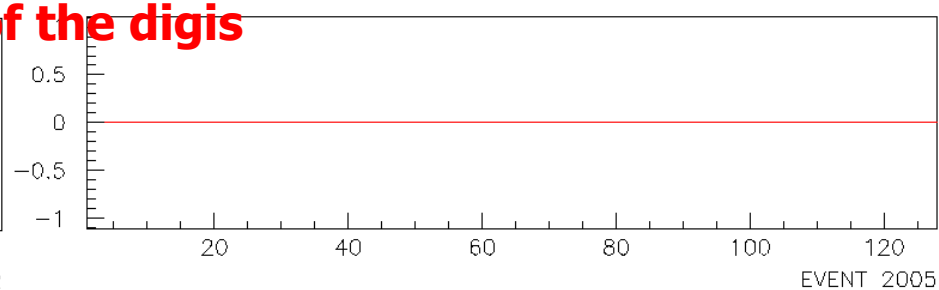
Subtracted signal





Output of the digis

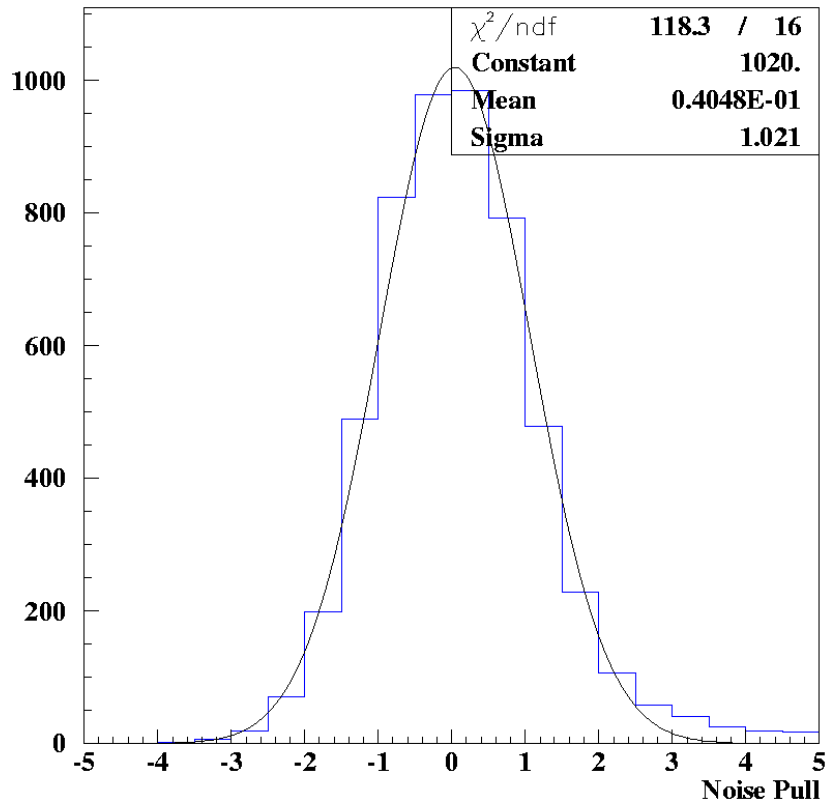




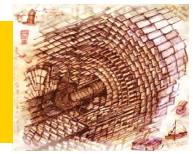
Some plots from the TestBeam

The noise pull is really good till at least 3 sigmas

The noise is really gaussian (ad shown by Roberto)

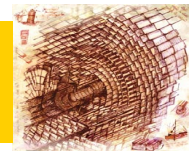


Conclusions on software



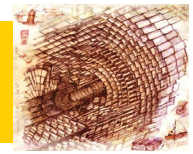
- The framework EXIST (more details will be given at the next b-tau meeting)
- Even if not ideal, the concrete classes should be enough reasonable to start some analysis on real data
- Mostly important, people are encouraged to start and port their algorithms to the framework (TT6?).
- Tracking?

Analysis



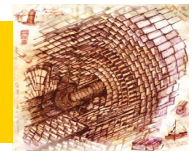
- Results from PISA group (already shown during last Tracker week). No update.
  - Cluster multiplicity increase to be understood
  - Looking at the module 1 (not-inverted) may help
  - Access to the APV parameters database needed
- Leonello is working on APV channel-by-channel gain differences
- Simone has started to resume Firenze analysis

Simulation



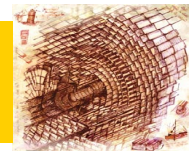
Next talk

Future Test-Beam



- Some ideas talking around (very preliminary)
  - Single module
  - Single Layer (DS)
  - Layer 1 + Layer 3
  - Tracker slice
- To be understood what we can learn from each of them, if we need telescopes, etc…

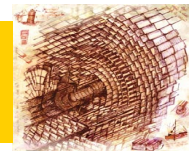
Conclusion



- Software already in a good stage. We need to start implementing algorithms
- Tracking still not possible in ORCA. Waiting for RawReadOut
- Analysis is still in early stage. Many things to understand in data.

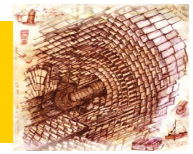
To help finding documentation and exchange results a web page has been created
http://cern.ch/ciulli/tbeam/testbeam.htm

Abstract classes (framework)



- **ApvAnalysis: the class which owns all the others, can talk to the EventReader**
- **ApvEventReader: the only source of data. Can be a Zebra file reader, COBRA, a random number generator, an ASCII reader.**
- **TkPedestalCalculator: computes and subtracts pedestals.**
- **TkCommonModeCalculator: computes and subtracts event by event CM.**
- **TkApvMask: masks strips if dead/noisy/whatever.**
- **TkNoiseCalculator: computes the strip by strip noise.**
- **TkZeroSuppresser: given the pedestal subtracted signal, the mask and the noise, suppresses the strips to lower data volume.**

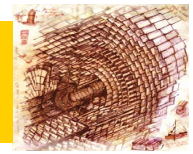
Concrete implementations



To prove the functionality of the package, concrete implementations were inherited from the abstract interfaces. They are all working and more or less reasonable, but are intended only as examples.

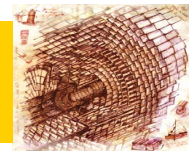
- **ZebraReaderAdapter : public ApvEventReader**
  - **reads the data from last test beam. It is an interface to Gabriella's ZebraReader.**
- **RandomApvEventReader : public ApvEventReader**
  - **generates random raw data, simulating pedestals, noise, signal of single strips and clusters with reasonable distributions.**

Concrete implementations (2)



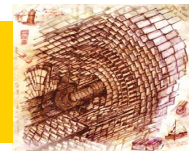
- **ReferencePedestalCalculator : public TkApvPedestalCalculator**
  - **Accumulates statistics for N events, and then averages them to allow pedestal subtraction.**
- **ReferencePedestalCalculatorWithSignalRejection : public TkApvPedestalCalculator**
  - **Same as above, but tries to exclude from the pedestal calculation strips containing real signal by smoothing the response in time.**

Concrete implementations (3)



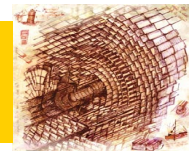
- **ReferenceCommonModeCalculator: public TkCommonModeCalculator**
  - **Averages the answer on the 128 strips and allows CM subtraction.**
- **ReferenceCommonModeCalculatorWithSignalRejection: public TkCommonModeCalculator**
  - **Same as above, but skips strips with a calculated CM compatible with coming from real signal.**

Concrete implementations (4)



- **ReferenceNoiseCalculator: public TkNoiseCalculator**
  - **Accumulates statistics to allow an estimate of the strip by strip noise.**
- **ReferenceNoiseCalculatorWithSignalRejection : public TkCommonModeCalculator**
  - **Same as above, but skips the strips that in one event have an estimated noise bigger than 3 times the previous estimate.**

Concrete implementations (5)



- **ReferenceApvMask : publicTkApvMask**
  - **Masks strips which do not show a visible noise as dead (quite arbitrary, I must admit …)**
- **ReferenceZeroSuppresser : public TkZeroSuppresser**
  - **Returns only the strips which have S/N greater than 2**
- **ReferenceClusterZeroSuppresser : public TkZeroSuppresser**
  - **Returns only the strips which have S/N greater than 5 if isolated, greater than 2 if in a cluster.**

Some results with the RandomApvEventReader

Event by event CM

Event by event noise

```
*-----------------------------------------*
*             NEW EVENT 10197
*-----------------------------------------*
CommonMode generated with RandFlat::shoot(-3, 3) = -1.87979
Noise generated with RandGauss::shoot(0., 3.)
Generated Cluster on strip 46 size = 46.2764
===> Generated Cluster on strip 46, cluster width 3
-----: ZeroSuppresser returned 3 active strips
Strip: 45 Signal 16.0713
GENERATED SIGNAL CORRECTLY FOUND: Strip: 46 Signal 49.095
Strip: 47 Signal 21.2094
```

Cluster injection

Zero suppresser output