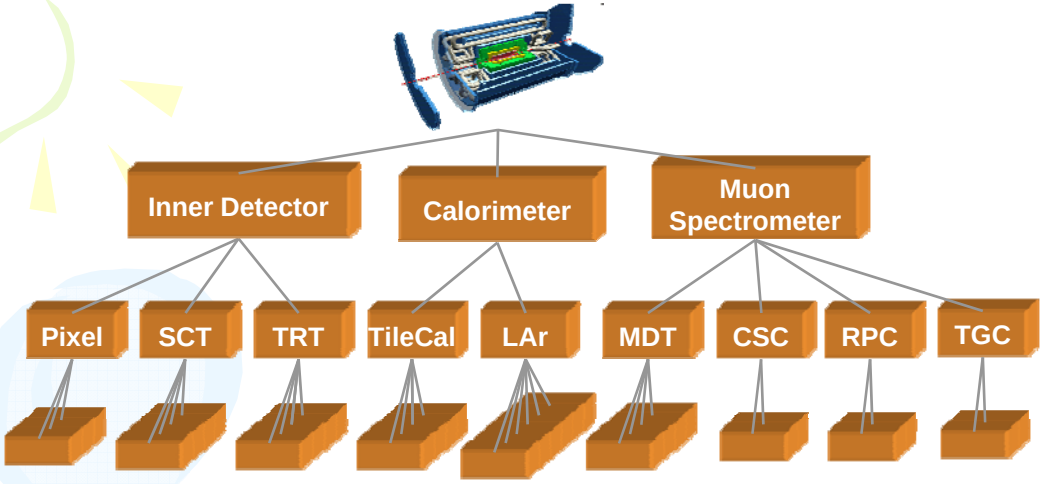
A software framework for Data Quality Monitoring in ATLAS

S.Kolos, **A.Corso-Radu** *University of California, Irvine*,
M.Hauschild *CERN*,
B.Kehoe, H.Hadavand *Southern Methodist University*

Outline

- The ATLAS Data Quality challenge
- Data Quality Framework
  - Design & Implementation
  - Experience during ATLAS commissioning
- Conclusion

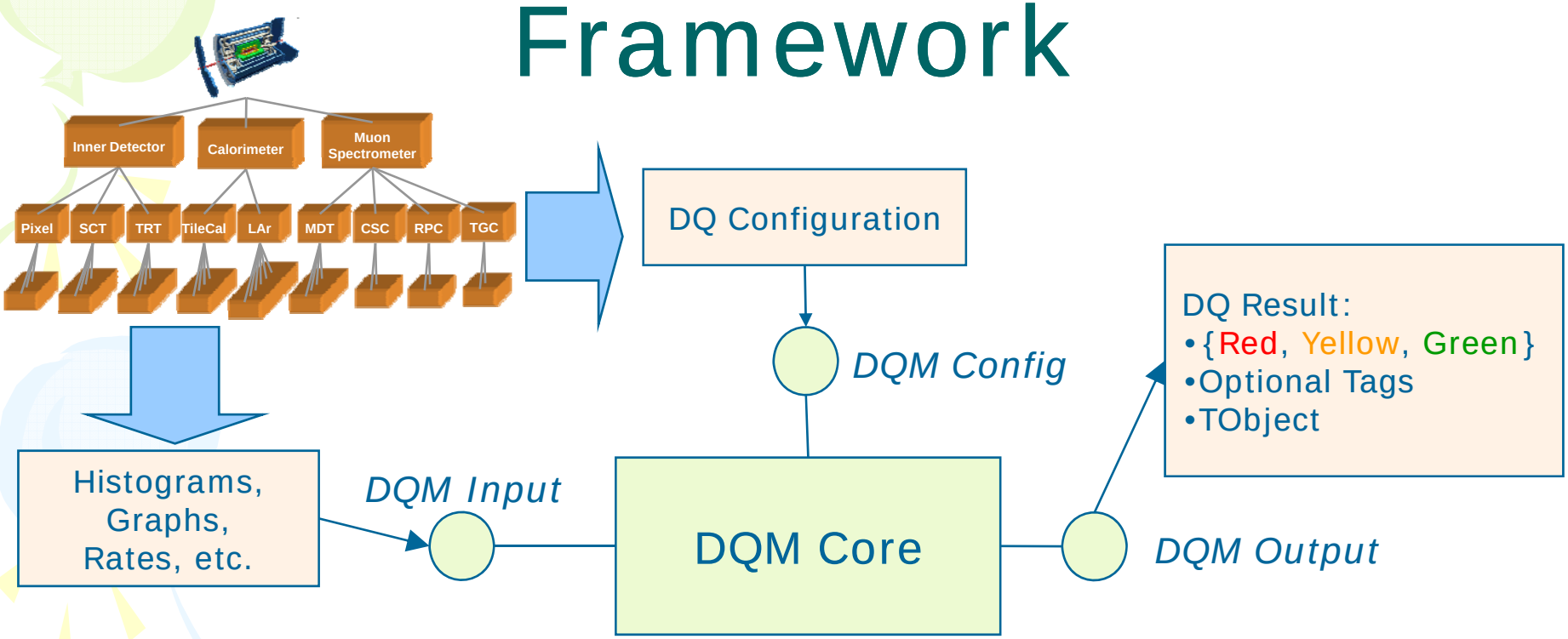
ATLAS DQ Challenge



- 3 main detectors
- 10 sub-detectors + LVL1 Trigger
- 33 sub-systems
- Each sub-system can be used independently
- 140M channels
- Bunch crossing rate 40MHz
- 3 Trigger levels reducing rate by 10^5

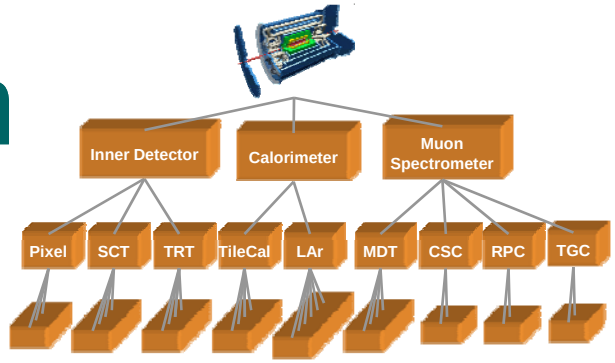
- The Data Quality Challenge:
  - Fast and reliable detection of DQ problems during data taking and reconstruction

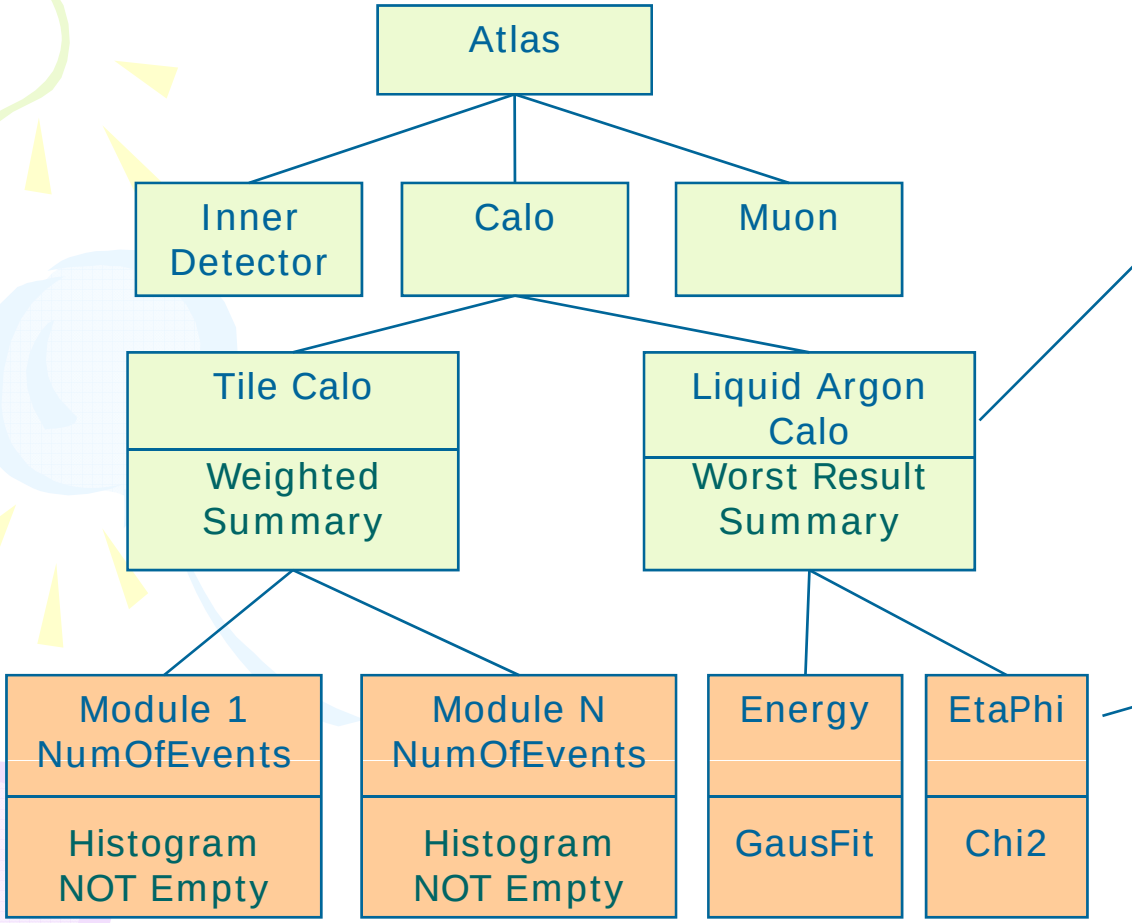
The Model of Data Quality Framework



- DQM Core:
  - Read DQ Configuration
  - Algorithm execution is data-driven:
    - Execute DQ algorithms as soon as input data becomes available
  - Produce DQ Result

DQ Configuration





DQ Region:
- Produces summary result for given sub-system
- It is based on the children results

DQ Parameter:
- Produces DQ result for a given quantity
- Analyses input data with specific algorithm

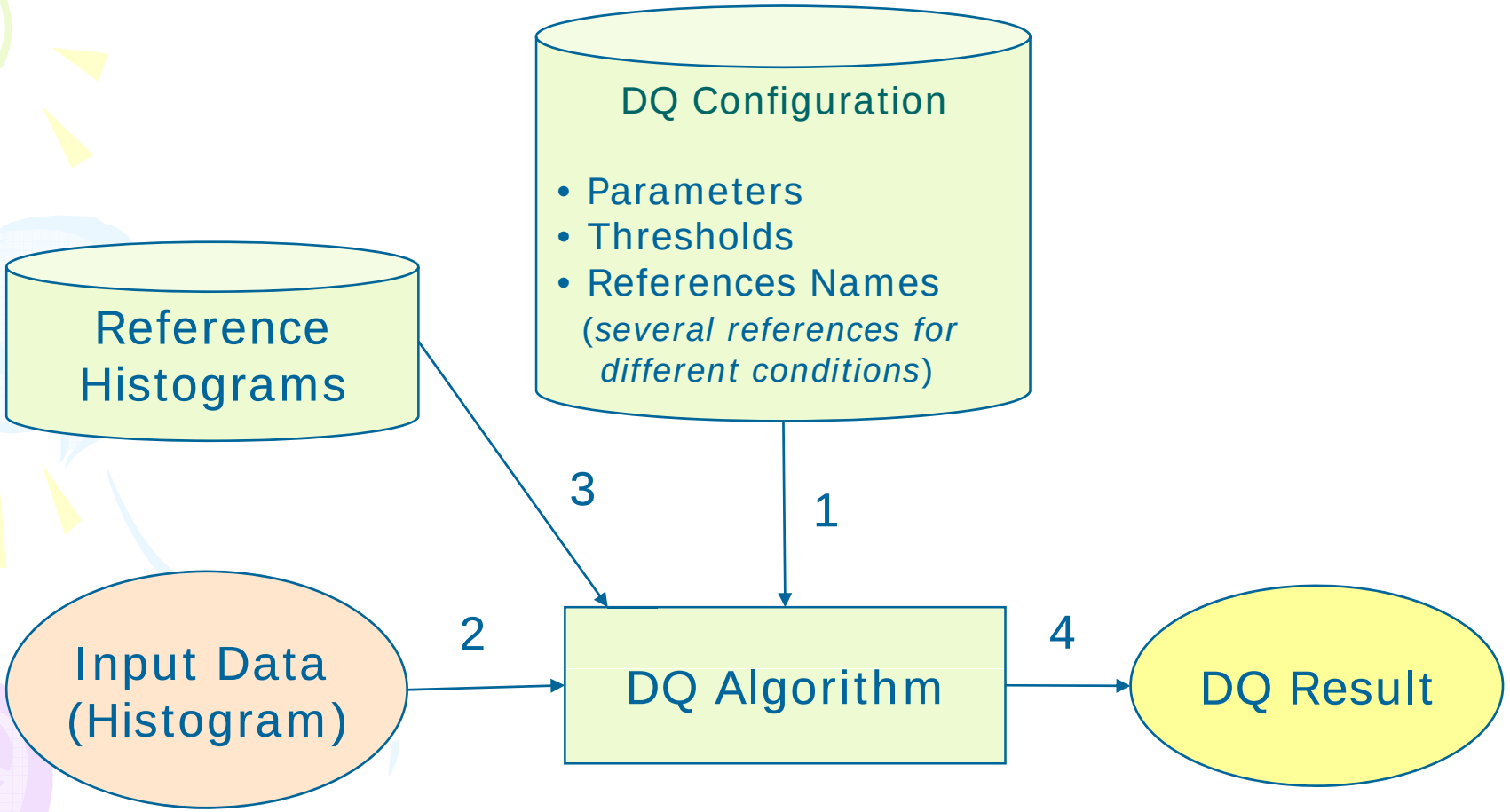
DQ Algorithm

- C++ function which analyses histogram and produce the DQ Result for the corresponding DQ Parameter
- Special type of DQ Algorithms is SummaryMaker:
  - Produces DQ Result for a DQ Region
  - This result is based on the DQ Results of the Region's children
- Custom algorithms can be easily plugged in by:
  - Inheriting from the Algorithm base class
  - Implementing two abstract methods: clone and execute

Data Quality Algorithms

- DQMF provides shared algorithms repository:
  - contains a set of predefined algorithms:
    - Reference Histogram comparison
      - *Chi2, Kolmogorov, Bin comparison (using ROOT)*
    - Fits:
      - *Gaus, Landau, Pol1 (using ROOT)*
    - Bin Threshold:
      <, >, <=< >=
    - Basic Histogram checks
      - *All_Bins_Filled, Histogram_Not_Empty, No_UnderFlows, No_OverFlows*
    - Basic Stat Checks
      - *CheckHisto_Mean, CheckHisto_RMS*
    - Etc.
  - Contains 2 summary makers:
    - Worst result
    - Weighted result
- Repository is filled up by ATLAS sub-system experts:
  - Share experience between sub-systems
  - Avoid duplication of work

DQ Algorithm Data Flow

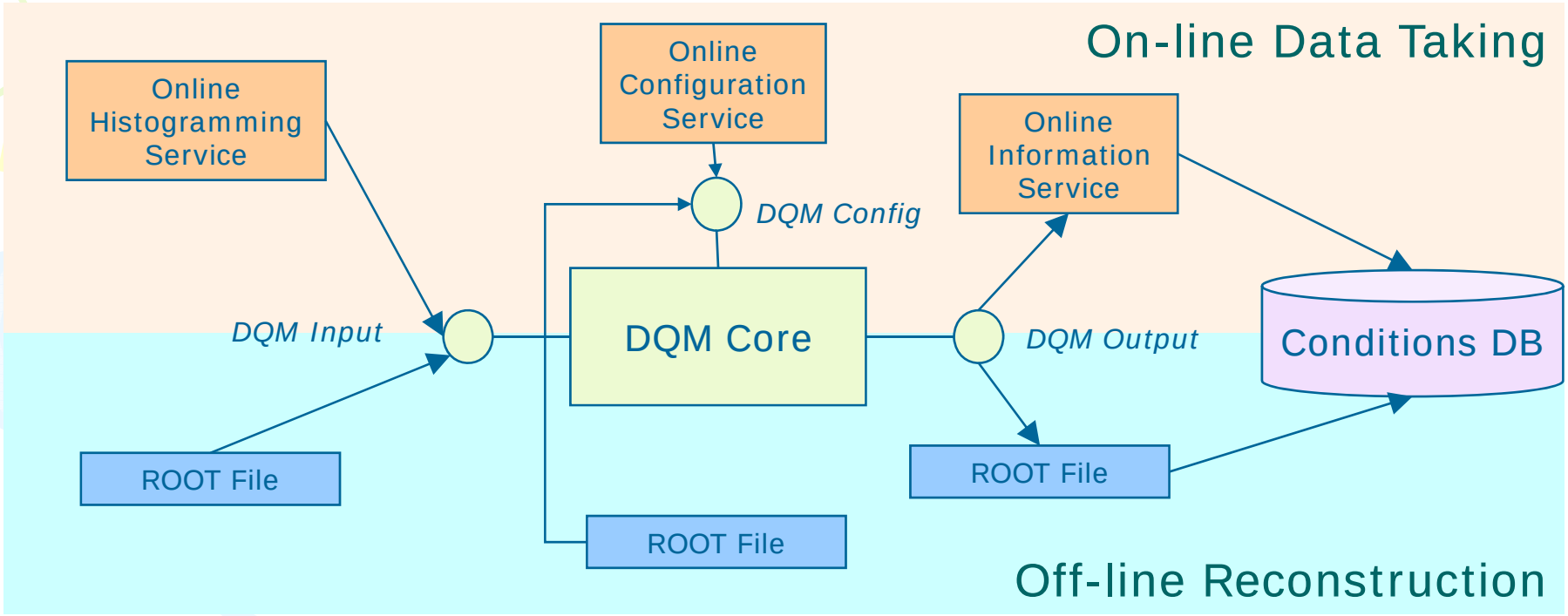


DQM Workbench

- DQMF provides set of Root macros which can be used to run DQ Algorithms in Root shell:
  - For algorithms development and debugging
  - For preparing configuration

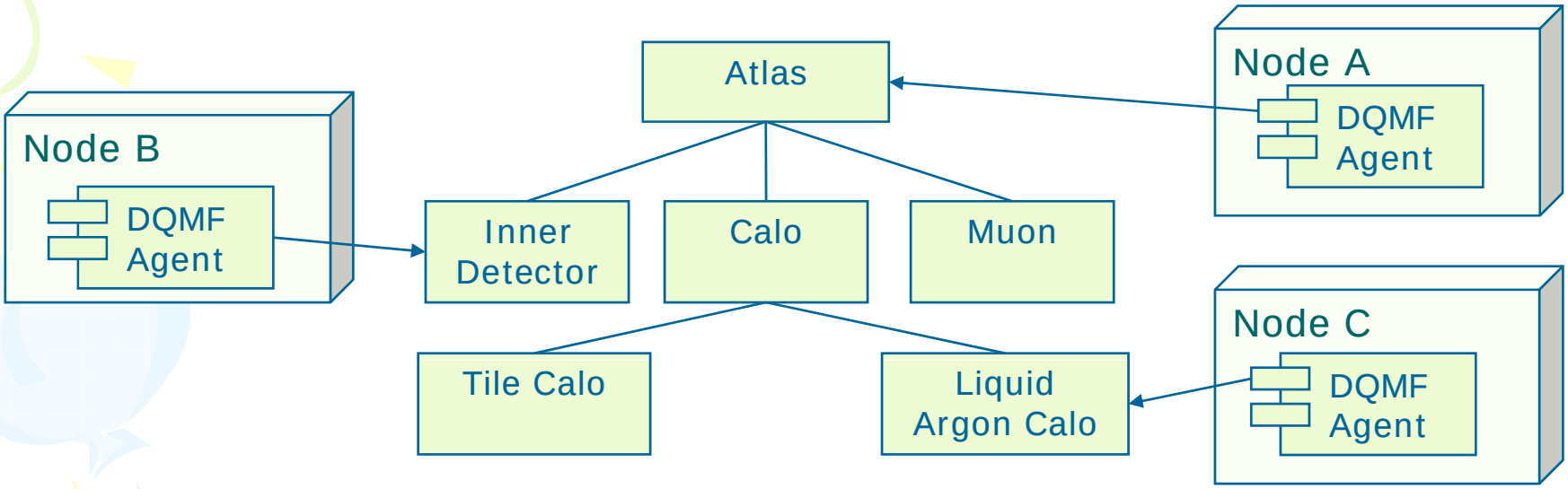
```
root [0] .x Workbench.C
root [1] .x TestBinContentCompAlgorithm.C
Number of bins 2 Sigma away from reference is 16
Green threshold: 33 bin(s);   Red threshold : 1 bin(s)
Result 2
```

On-line vs. Off-line



- DQMF is used for DQ assessment during the online data taking as well as during the off-line reconstruction

On-line DQMF Scalability

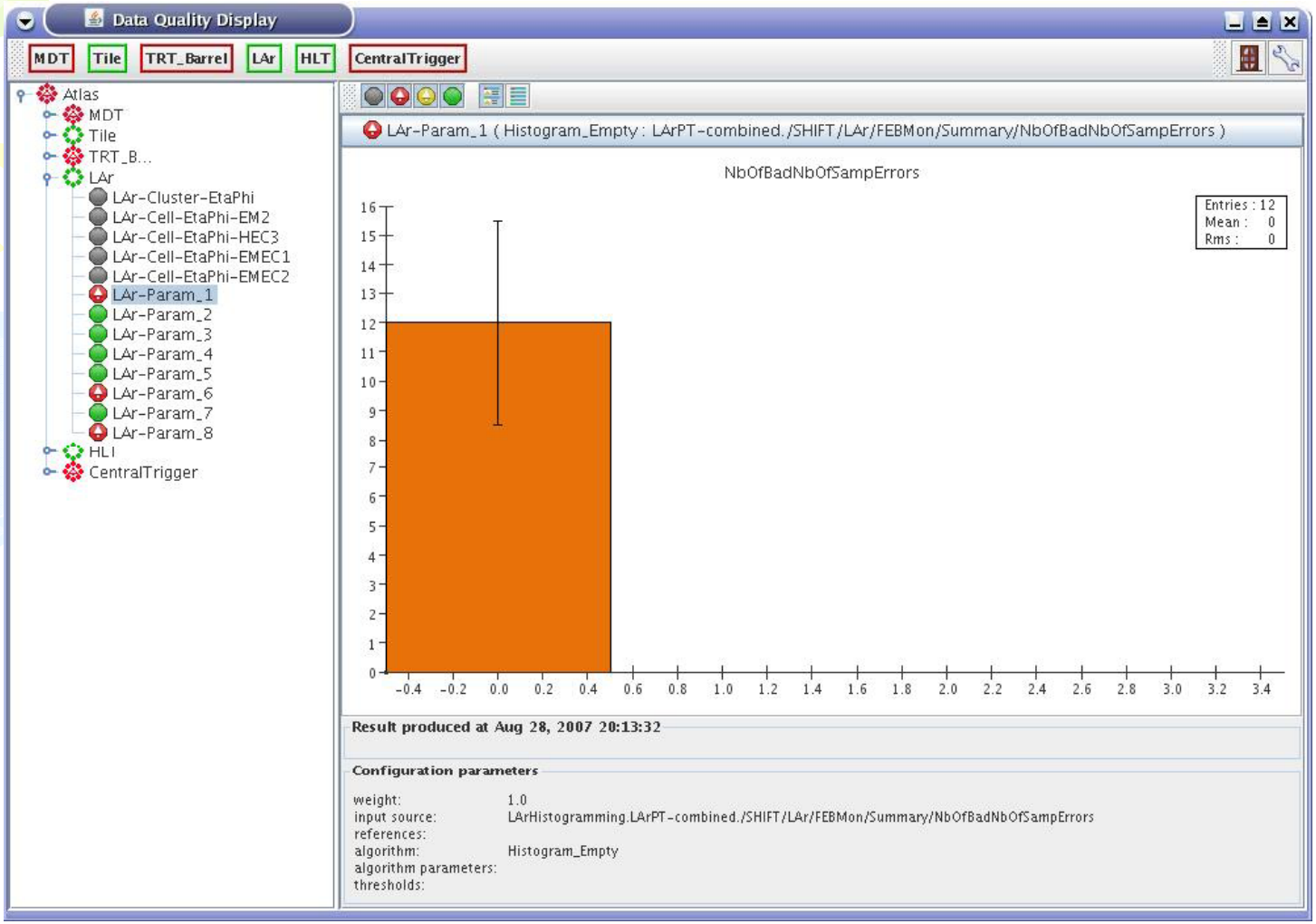


- Online environment implies high performance requirements for the DQ Assessment
- The DQMF functionality is provided by a process called DQMF Agent
  - Any DQRegion can be processed by individual DQMF Agent on a separate node

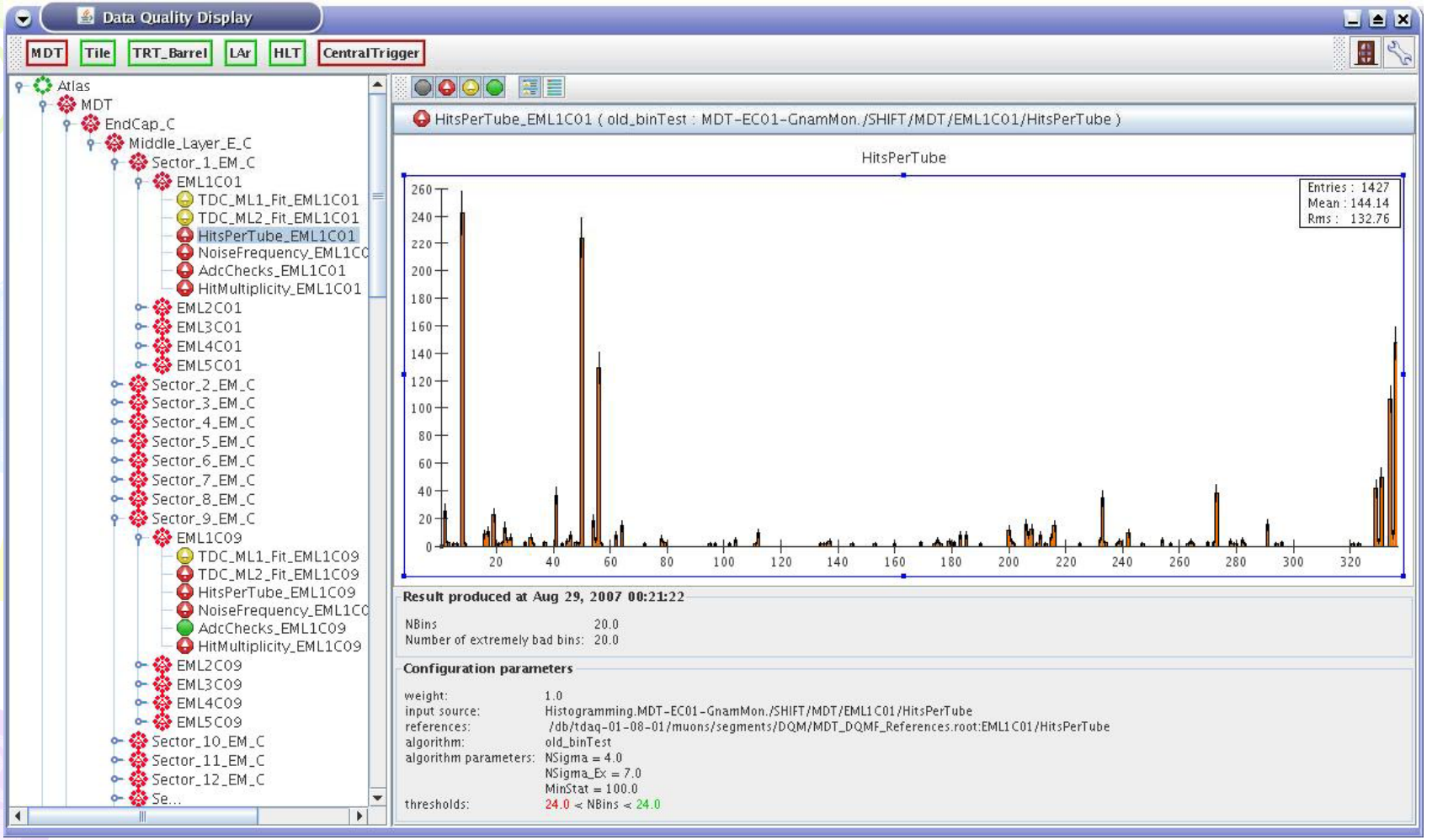
DQMF at the ATLAS Commissioning

- Online DQMF configuration for the last Commissioning run includes ~2000 histograms:
  - Signals distributions, energy distributions, noise distributions, Trigger rate, etc.
  - This configuration has been handled by a single DQMF agent process
- Lessons learned:
  - The JAIDA which we were using for the DQMF display prototype does not fulfill our performance & scalability requirements
  - The final version will be implemented using ROOT/Qt/C++

Online DQMF Display (1/2)



Online DQMF Display (2/2)



Summary & Conclusion

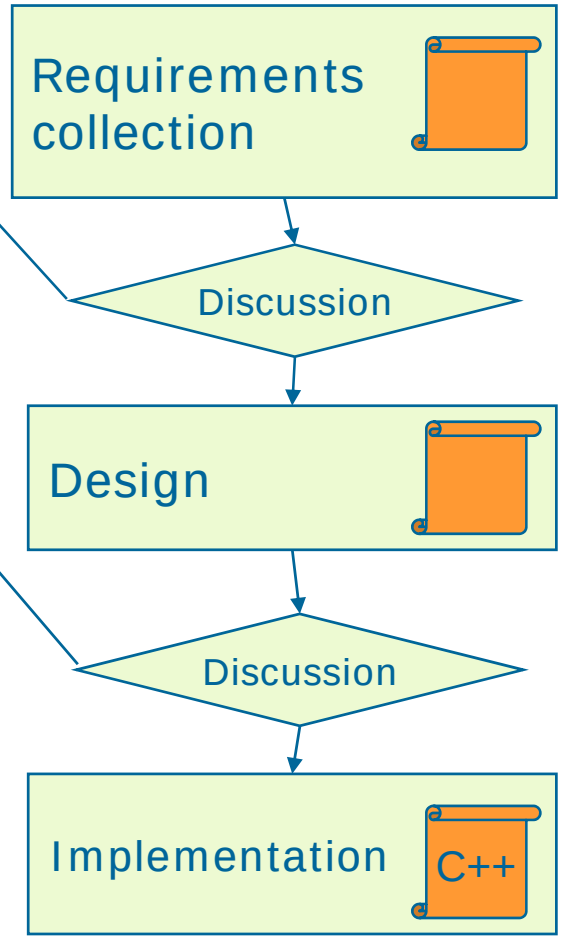
- DQMF is framework for DQ analysis which is:
  - Efficient – implemented in C++ using ROOT
  - Flexible - fully configurable via simple interface
  - Simple – no code development is necessary if standard algorithms can be used
  - Scalable – can be naturally distributed over any number of machines
- It is being successfully used now in both On-line and Off-line environments for the ATLAS commissioning

Backup

Software development process

At this stage the scope of the framework has been extended to the Off-line reconstruction

Design has been influenced by the feedback to the Requirements document



- The standard software development model has been found extremely useful