

# LAr OO Reconstruction

- People: H. Ma, S. Rajagopalan, J. Schwindling
  - + help/advise from several others
- This work started around September 1999
  - documentation of existing code, use cases, new design
- Establish an infrastructure for the LAr Reconstruction that will easily allow us to implement, test and optimize algorithms
- Extensive discussions with others
  - LArg community
  - Joint meeting with Tile group at BNL

# LAr OO Reconstruction (2)

- Framework:
  - interfaced to PASO
- Inputs
  - access to G3 digits (ZEBRA) using TestEvent
  - someday with Objectivity
- Output: (LArAnalysis Package)
  - use of HTL to print & dump contents of histograms
  - HBOOK ntuples using wrapped fortran code
- Testing:
  - barrel/endcap using single  $e/\gamma$  with  $E = 20 - 60$  GeV &&  $H \rightarrow \gamma\gamma$  events.
  - Comparisons with ATRECON
- Documentation can be accessed via
  - <http://www.cern.ch/Atlas/GROUPS/LIQUARGON/software/>

# Miscellaneous comments

- Modularity
- Separation of data and algorithm objects
- Tried to preserve similar names as in atrecon wherever possible
- Corrections/Calibration and some other pieces of the source are just as those used in atrecon (but in c++). **Not reinventing the wheel!**
- Geometry/Calibration constants are embedded as “static const” in the code (for now)

# Example: AlgorithmSelector

```
LArAlgorithmID aid;           // an instance of an algorithm identifier
LArClusterMaker* maker = new LArClusterMaker()
    maker → CreateCorrection(aid.eta_correction())
    maker → CreateCorrection(aid.phi_correction())
    maker → CreateCorrection(aid.shape_correction())
    maker → CreateBuilder(aid.SlidingWindow())
```

Each cluster maker thus defines the environment necessary to find clusters

You may make as many cluster or cell makers. Associated with each maker is a: unique builder, a set of corrections, calibration, etc.

The “maker” is then passed as a strategy to the managers

# The Managers

- LArManager contains a collection of the cell and cluster managers.
- A unique identifier for a cell or a cluster manager is being worked out
- LArCellManager contains a collection of all cells:
  - A cell can be accessed given its identifier.
  - The begin and end iterator of the list of cells can be accessed.
  - A list of identifiers corresponding to a given range can be accessed.
  - Smarter collection objects are being considered.
- LArClusterManager contains a collection of all clusters:
  - The begin and end iterator of the cluster list can be accessed
  - List of clusters sorted and/or filtered is possible

# class LArCell

friend class LArCellCorrection

public:

float ID()

float sampling()

type? status()

float energy()

float et()

float eta()

float phi()

float deltaR()

void print()

private:

Identifier m\_ID

float m\_energy

float m\_eta

float m\_phi

# class LArCluster

friend class LArClusterCorrection

public:

energy(), energy\_uncorrected(), et(), eta(), phi() *// for each cluster*

energy(), energy\_max() *// for each sample*

eta(), etamax(), etasize()

phi(), phimax(), phisize()

iterator cell\_begin(), cell\_end() *// cell list*

float qpoint(z, dz, eta), qgcld() *// pointing, depth*

void addcell(LArCell\* cell) *// adding cell to list*

void print()

# LArAnalysis Package

- Use of HTL to book/fill histograms
  - Get a print of the histogram in the output log file
  - Dump of the histogram contents
  - Use with Objectivity/Explorer will allow us to make use of other features
- We can also write out standard HBOOK ntuples. We simply call these from C++ code.
  - Extremely convenient, especially in performing comparisons with ntuples from ATRECON.

# Packages

Calorimeter/CaloRec

common to LAr/Tile

LArCalorimeter/LArRec

common to LAr

LArCalorimeter/LArRec/EM\_Rec

EM specific

LArCalorimeter/LArRec/HEC\_Rec

HEC specific

LArCalorimeter/LArRec/FCAL\_Rec

FCAL specific

TileCalorimeter/TileRec

Tile specific

- In addition, there will be packages for Detector Description, Event, ... as required.

## Next few steps

- Fix names of common code (prefix “Calo”)
- Release existing code/documentation in CVS (not tagged)
- abstract base class for Cell and Cluster objects?
- Cell/Cluster object definition
- Test with Fcal, HEC, Tile
- Ensure EM results are indeed the same as ATRECON
- Performance issues: (timing, memory,...)
- Platform support: (Linux, Solaris, HP)
- Implementation of other use cases
- Migrating to a new framework that may become available (May?)