# Reconstruction

David Rousseau

- Introduction
- Status and plans of the various detectors
- the next months

# Contributors to reco software

- three populations (sometimes mixed):
  - people involved so far in physics TDR
  - people involved so far in OO software
  - people involved so far in hardware
- they would have different approaches which should benefit from each other
- they are spread around the world

- importance of up to date web based documation/information
- boost productivity by encouraging use of tools (not all existing now):
  - automatic tool for code checking (coding rule)
  - debugger, profiling tools
  - event display
  - analysis tool
- need guidelines: e.g use STL, CLHEP, good C++ code example

# Introduction

- existing reconstruction code was successful (see 1000 pages of physics TDR)
- we know how to reconstruct full events with the complete detector
- detector performance is the limiting factor in most apects

# Evolution

- **now we need to provide an even better reconstruction software:**
  - **OO/C++**
  - **using databases rather than hard-wired numbers**
  - **online data based rather than GEANT3 based (e.g., ADC to GeV, time to drift distance...) (test-beam)**
  - **trigger aware**
  - **realistic (alignment/calibration, noisy/dead channels, any luminosity, real B field)**
  - **faster (and with low memory usage)(for event filter and anybody else)**
  - **robust (should survive beam halo events, cosmics and DAQ hickups)(for event filter and anybody else)**
  - **higher performances (scrab % of efficiencies and resolution)**
  - **new perfomances (provide final analysis objects (see later), $\pi^0$, energy flow using tracking information, other great ideas from combined performance groups)**
  - **...**

# Trigger

- **LVL2**
  - **LVL2 will not use offline software, however..**
  - **LVL1/LVL2 to be simulated in same framework than offline**
  - **LVL1/LVL2 information (trigger menu bits and ROI information) passed on to Event Filter (possibly to seed reconstruction) and offline (to check trigger)**
- **Event filter**
  - **Event filter will run offline software but will hopefully not write reconstruction code**
  - **performance (typically 1 s/event, not enormous memory) reliability and robustness are very important**
  - **performance to be obtained by optimizing offline code and (if necessary) running simpler algorithms**

# Calorimeters

- **Larg calorimeter:**
  - **wish to be able to run part of reconstruction on test beam (hence modular code)**
  - **first step is to understand more precisely what exists:**
    - **C++ attempt of Larg reconstruction**
    - **current code reverse engineering (=>a document is foreseen)**
  - **prepare next step=first iteration of better software**
- **Tile calorimeter**
  - **no attempt (yet) to have C++ calo reconstruction**
  - **TileCal pilot project is an important experience**
  - **a document is foreseen to describe it so that other detectors can benefit**
  - **next step is reverse engineering of existing atrecon code (do not wish to do any wrapping)**

# Trackers

- **Inner detector:**
  - **several C++ packages existing already**
  - **no real duplication since algorithms are really different**
  - **however identified commonalities should become really common code**
  - **modules with same functionnalities but different strategies should be identified to be tested individually**
  - **vertexing and conversion finding code needs wrapping or rewriting**
- **Muon system**
  - **fortran code used for TDR**
  - **complete C++ code exists but has not been tested on real events**
  - **first step is to identify objects and write interfaces**

# Event definition

- What will the end-physicist use as objects ?
- Containers: framework issue? (need homogeneity and simplicity)
- Contents:
  - Combined entities:
    - electrons/gamma/muons/taus/hadrons/jets identified for $p_T$=0 to infinity
    - 4-momenta plus typical information on these objects
    - pointer to truth (lias with Monte-Carlo group)
    - pointer to intermediate objects
  - Intermediate entities
    - clusters/tracks/jets
    - (as above)
    - pointer to semi-raw data objects
  - Semi-raw entities (=today's DIGI)
    - cells/strips/pixel
    - (as above)
    - pointer to raw objects
  - Raw entities (from online event)
- additional levels possible (e.g pixel clusters)

# Event definition

- **Data entities to be defined first.**
  - **Existing combined ntuple (=RECB banks) could be used as a starting point**
  - **Concentrate on entities which are likely to remain in all design (tracks will be tracks, clusters will be clusters)**
  - **Information to be divided in small logical chunks (e.g. helix parameters, strips lists)**
  - **Names of objects and variables should be proposed (naming conventions ?) without too much binding to history.**
- **Operation on entities to be defined next**
- **Data entities and operation = basis for interfaces**

# Code evolution

- Identified entities broadly define an event flow
  - careful however with loops (e.g. use track to refine muon but track found in muon seed, recalibrate cell wether in jet or electron)
- Algorithms between entities are (probably big) modules
- These big modules could be wrapped from fortran if needed
- Breaking up big modules in smaller modules should then be done but wrapping of the smaller modules could be dangerous (unless very small indeed)

# Future of Atrecon

- **Atrecon needs to be kept alive till something at least as good and complete exists, for**
  - **detector performance studies with new layouts**
  - **ATRIG vs offline studies for Trigger TP (spring 2000)**
  - **new physics studies**
  - **test of new C++ packages against old one in a full chain, (e.g. quality of tracking to be tested for efficiency of b-tagging or electron identification)**

# PASO

- implement into PASO fortran wrapped code (or existing C++)
- not too much work should be spent fitting things into PASO that do not want to go there
- rather use PASO to identify desired functionalities from the final framework
- PASO should then be updated (if not too much work, given short lifetime)
- TileCal pilot project also good  experience

# a slice in PASO

- output (e.g. track classes) from upstream reconstruction packages should be tested rather quickly
- proposal: PASO could also be used to implement a “slice” of final atlas software
- PASO should have the possibility to read Combined Ntuple or ZEBRA RECB bank created by existing Atrecon and the proposed track (or cluster or muon) classes could be recreated (maybe only partially)
- Then they could be used in simple algorithms ($K^0$ finding, electron/muon identification,...)
- simple analysis (e.g. $H \rightarrow$ llll reconstruction) could then be developped (in view of testing Analysis Tools)
- Ease of use would be a criterion for prefering one design over another
- Note: good way to get started with C++

# Conclusion

- ATRECON is great but...
- Migration of the code should start by
  - identify entities and variables
  - then work on algorithms
- could use PASO to:
  - implement single modules
  - test simplified slice of reconstruction