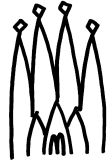




Issues identified in sub-detector OO software reviews

Calorimeters:	18th February
Tracking:	24th March
Rich:	31st March



Structure of this session

- ◆ **Brief overview the three reviews**

- ◆ **Presentation of issues raised**
 - **Pros and cons of different approaches**
 - **Some solutions discussed in the reviews**

- ◆ **Open discussion**
 - **Can we agree on common approaches?**
 - **Essential for homogeneity and maintainability of LHCb simulation, reconstruction, analysis environments**
 - **How do we foster exchange of information between sub-detectors?**
 - **Generic solutions to common problems**
 - **Sharing of good design ideas**



Calorimeters

- ◆ **Discussion document:**
 - **Detailed data model design**
 - Including class definitions (header files)
 - **Algorithm descriptions**
 - Including code examples and use cases

- ◆ **Issues raised**
 - **Connection to Monte Carlo data**
 - **Fast access to contained objects**
 - Using cell ID as index
 - **Coding styles**
 - Use of STL algorithms and function classes (functors)
 - Compactness of code vs. Maintainability/Readability



Tracking

- ◆ **Discussion document**
 - Procedural description of algorithms
 - Data Model
 - No implementation details
- ◆ **Issues raised**
 - Design driven by existing needs
 - Interaction with other detectors (RICH, VELO...)
 - Tracks, tracking Hits
 - Connection to MC truth
 - Sorting of contained objects
 - Algorithms vs. Tools vs. Services
 - Detector geometry (see this afternoon)
 - Complete material description in XML
 - Synchronisation of XML and CDF descriptions
 - Granularity of detection cell



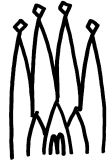
RICH

◆ Discussion document:

- Status of standalone program, adapted to GAUDI
- Detailed use case analysis
- Detailed detector and event model design
- Architecture
 - Adapter, Strategy, Monitor

◆ Issues

- Is design driven by chosen algorithm (global likelihood)?
 - More use cases to be considered
- Simulation during reconstruction
 - Needed for detection efficiency calculation
- Connection to other detectors (sequencing, updating of data)
- Connection to MC truth
- Sharing data between sub-algorithms



Connection to MC truth: two approaches

◆ Inheritance

MCCaloDigit **isA** CaloDigit, IT/OTMCDigi **isAn** IT/OTDigi

☺ **Fast and easy access to MC information (dynamic cast)**

☺ Space efficient (4 or 8 bytes)

☹ Needs discipline (can easily be abused)

☺ **Reconstruction code is the same, using real data class**

☹ But cannot be tested on real data until it comes

? **How to create or copy MC object without using MC class?**

e.g. `new CaloDigit` in calibration code

◆ Indirect association

MCCaloDigit **hasA** CellID, CaloDigit **hasA** CellID

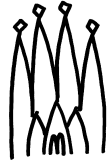
☹ **Slow(er), more complex access**

☺ **Clear separation between MC and data**

☹ But can still be abused...

■ **Only option for objects created by pattern recognition**

● Reconstructed and truth tracks sharing hits



Connection to MC truth: tools

◆ Associators

■ Encapsulate details of association

- Can be simple dynamic cast, simple or complex navigation, majority logic, etc.
- Data model can evolve without affecting analysis code

◆ Monitors

■ Algorithms that monitor performance of code

■ May know about existence of MonteCarlo

- Can use associators to make data / MC comparisons



Connection to MC truth: Recommendations

- ◆ **Do not infect reconstruction code with knowledge of Monte Carlo**
 - Use only real data classes in reconstruction code
 - No MC header files in Brunel code!
 - Use Monitors to make comparisons

- ◆ **Do not infect data/MC comparisons with implementation details**
 - Use Associators to encapsulate data/MC connection

- ◆ **Choose association method most suitable to your use**
 - Inheritance OK for DIGIs, Hits etc.
 - Provided problem of new can be solved (a virtual clone method?)



Examples of interactions between sub-detectors

◆ Definition of common base classes

IT/OTHitOnTrack **isA** TrMeasurement

- Tracking code deals with TrMeasurement
- How will VELO fit into this scheme?

◆ Working with shared classes

- What is a track? Who can update it?

◆ Sequencing of algorithms

- Tracking needs particle ID, RICH needs tracks

◆ Definition of responsibilities

- Primary vertex: VELO? Tracking? Somebody else?
- Who (and how) finds tracks in VELO?

◆ **NEED forum for discussion between sub-detectors**



Ownership of data

◆ A use case:

- Tracker finds tracks, gives ownership to transient store
- RICH takes these tracks, finds particle ID
- How can RICH attach particle ID to tracks it does not own?
 - Update track's pointer to PID info
 - Breaks rule that cannot update data on transient store
 - Save a new copy of the tracks with links to PID info
 - Safe, but proliferation of duplicate information
 - Save PID info, with link to corresponding track
 - Safe, but very inefficient for further tracking and analysis
- Based on PID, tracking wants to remove some tracks
 - RICH may still be pointing to these tracks!
 - Update a track quality flag?

Updating of pointers/flags probably OK

Deletion of data items in the store NOT OK



Adapters

- ◆ **Data items on transient store are simple**
 - **Cannot answer complex or specialised questions**
 - e.g. Tracks know only about states and measurements
 - **Different sub-detectors may need to ask different questions**
 - e.g. RICH asks tracks how many photons they will generate in a radiator

- ◆ **Adapters:**
 - **allow private view of the data**
 - e.g. RICH algorithm accesses only RICH tracks. These answer RICH specific questions. Generic track questions are forwarded to the Tracking tracks by adapters.
 - **shield algorithms from different data sources**
 - e.g. use same RICH algorithm for truth tracks or reconstructed tracks, just changing the adapter (c.f. converter)

- ◆ **Nice idea, but beware of making adapted objects too complex, compromising modularity**



Access patterns to contained objects

- ◆ **Use case 1:**
 - Clustering of ECAL requires asking for energy deposit in a given cell
 - How can `ObjectVector<CaloDigit>` be indexed by `CaloCellID`?
 - Could be done by specialising `ObjectVector` with `[]` operator accepting `CaloCellID` as index
- ◆ **Use case 2:**
 - Track finding algorithms require re-ordering of clusters according to a given quality factor
 - Cannot reorder in the data store
 - Can be done by sorting local copy of pointers to clusters
- ◆ **Are there any general solutions?**
- ◆ **Could adaptors have a role?**



Sub-algorithms vs. Tools (vs. Services)

- ◆ **Need to pass data to (and between) sub-algorithms**
 - GAUDI architecture favours publishing such data on transient store
 - Does not mean it will be made persistent!
 - Alternative is that context is passed to sub-algorithm via message
 - e.g. `Evaluate(my_event, my_detector)`
 - Couples algorithms, does not allow them to run independently
 - Some sub-algorithms need to be called several times per event
 - e.g. Track extrapolator, Kalman filter
- ◆ **New concept: "Tools"**
 - Take and configure a "tool" from a "toolbox" svc. at initialisation
 - One or more instances per algorithm (same as sub-algorithms)
 - Use tool when needed
 - By passing data with arguments, not through data store
 - Mechanism will be provided by Gaudi
- ◆ **Services are global to the application**
 - e.g. TransportSvc



DISCUSSION

- ◆ **Can we agree on common approaches?**
 - Connection to MC truth
 - Updating of transient event data
 - Access patterns
 - Use of tools, sub-algorithms, services

- ◆ **How can we exchange information between sub-detectors?**
 - Design of transient event data, common base classes
 - Sequencing of algorithms
 - Sharing of good design ideas

- ◆ **Other issues I haven't thought of.....**