Author: M. Cattaneo
Status: Draft 3
Last modified: 13th September 2000

PROPOSED LHCb coding convention: Event data model conventions

Deadline for comments on draft 1:	11th September 2000
Presentation for approval:	LHCb week in Milano
Implementation:	Gaudi release 6, Brunel v2

Introduction

This document is intended as a discussion document whose goal is to define a set of conventions to ensure that the LHCb transient event data model has a coherent structure across the different sub-detectors. Each section of the document will address one aspect.

1. Glossary

The event data is logically subdivided in sub-events that are the results of a processing step.

- *MC Event* is the output of the physics event simulation. This typically includes MC Particles and MC Vertices
- *MC Hits* are the output of the GEANT tracking step. They typically include detector entrance and exit point, energy loss etc.
- *FE* data is the simulated output of the detector front end, as seen by the hardware triggers, when this is different from *Raw* data.
- *Raw* data is the output of the digitisation step of the detector simulation, and the output of the data acquisition system for real data. For example, ADC and TDC counts.
- *Coordinates* are the output of the reconstruction program when applied to detector hits. This typically consists of hit coordinates, calorimeter clusters etc.
- *Reconstructed* data is the final output of the reconstruction program. This typically consists of tracks, particle ID, energy flow objects etc.
- *Analysis* data is the output of specialised analysis algorithms.

2. Proposal for relationship between raw data and MC Hits

During the code reviews that took place spring 2000, it became clear that various approaches are possible to allow navigation from simulated raw data back to the MonteCarlo Hits truth information. Of the approaches proposed, two seem most relevant: navigation by inheritance, and navigation by association (lookup table).

- Navigation by inheritance
  In this scheme, a MonteCarlo class (e.g. *MCCaloDigit*) inherits from the corresponding real data class (*CaloDigit*), adding to the real data class a pointer to the MonteCarlo truth information (*MCCaloSummedDeposit* in this example).

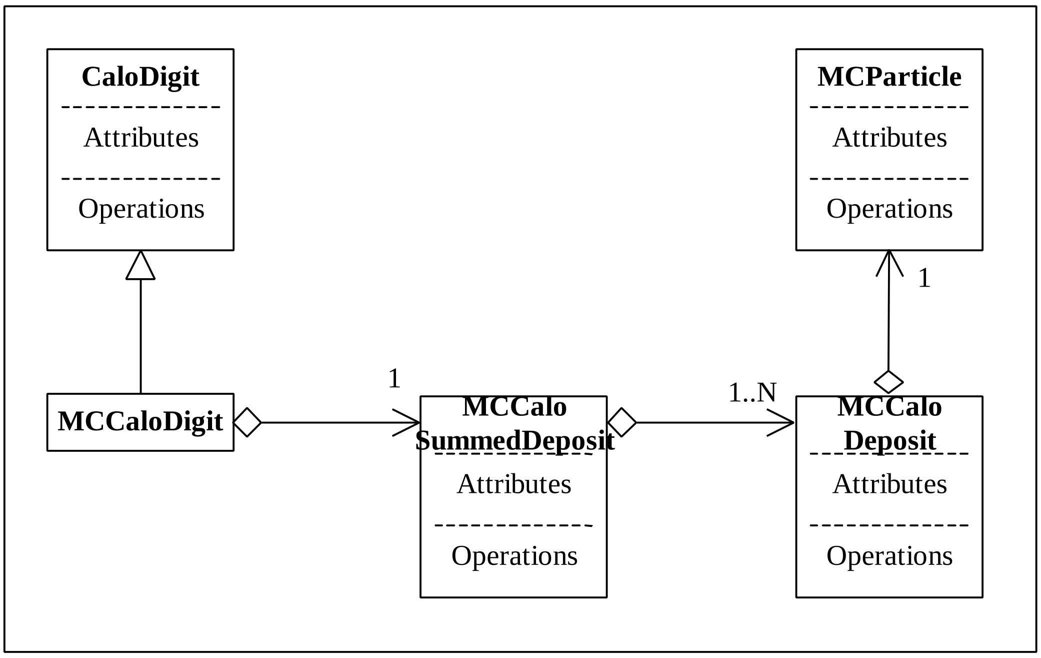
Advantages	Disadvantages
▪ Easy and fast access: navigation to MonteCarlo truth is as easy as dereferencing a pointer.	▪ The scheme is not general: it cannot work when the navigation is not deterministic, as is the case after pattern recognition.
▪ Space efficient: navigation information adds only 4 or 8 bytes.	▪ It is complex, e.g. when several truth hits lead to one digitising. It also leads to some technical problems, such as how to create a new *MCCaloDigit* from a calibration algorithm that only knows about *CaloDigit*s.
▪ Most algorithms (e.g. reconstruction) deal only with the base class: they do not see the difference between real and simulated data. To access the truth information they have to do a `dynamic_cast`.	▪ The criticism has been made that the reconstruction and analysis programs will never see real data without the MC links before data-taking begins, so it will not be possible to check that the software does not rely on MC truth. This is not a real problem: *MCCaloDigit*s could be generated in which the pointer is set to NULL.

- Navigation by association
  In this scheme, there are no explicit links between simulated digitisings (or the subsequent reconstruction output) and MC truth. The link is made via a link variable (such as the calorimeter *CellId*), or a lookup table in the case of many-to-many associations, or via some more complex algorithm in the case of non-deterministic relationships (such as reconstructed track to MC particle).

Advantages	Disadvantages
▪ The simulated raw and reconstructed data are identical to the real data. There is no risk of accessing the MonteCarlo truth "by mistake".	▪ The scheme does not use the power of inheritance. In particular, navigation is slow
▪ The scheme is general: it can work for all types of relationships.	▪ The connection path to the MC truth can become complex and will be different for different classes.

It is clear that both methods have advantages and disadvantages. In some cases (e.g. one to one relationships between digitisings and MC Hits) the inheritance mechanism is very appealing. In other cases (e.g. the rather vague relationship between reconstructed tracks and the generated particles) an associative algorithm may be the only solution.

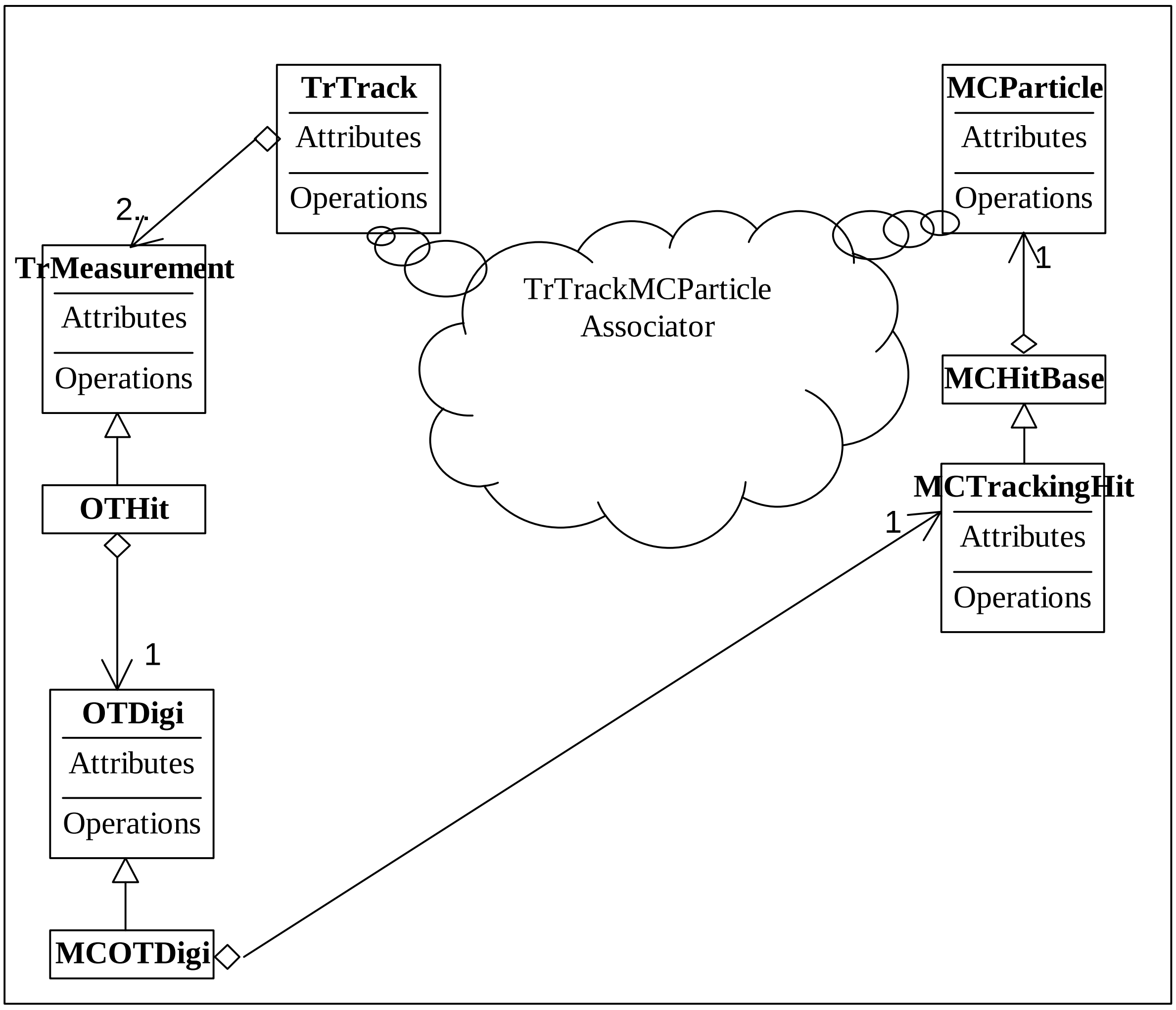
In the specific case of navigation from simulated raw data to MC Hit information, navigation by inheritance seems the most appropriate choice, as proposed by the calorimeter (see figure below) and tracking groups.



3. Proposal

For the time being we limit ourselves to the relationship between raw data and MC Hits, and propose that this be implemented using the inheritance mechanism discussed above. Note that the link is unidirectional, allowing efficient navigation from the reconstruction class to the MonteCarlo class (navigation in the opposite direction is also possible, but requires looping over all, e.g., hits to find the required one).

With this choice, it should always be possible to navigate from any reconstruction class back to the MC Truth classes. It is understood that navigation via all the intermediate classes may not be optimal in terms of I/O, so the data model could be optimised at a later stage by introducing additional navigation classes. This is illustrated in the following figure, where the "cloud" hides classes that allow direct association of *TrTrack*s to *MCParticle*s, without having to load the *OTHit*s and *OTDigi*s. Such additional navigation mechanisms could be implemented at a later stage to optimise the I/O performance of the analysis algorithms.



Proposed naming convention

When the navigation to MC truth is implemented by inheritance, as discussed above, it is proposed that all MC classes that derive from a real data base class have the same name as the base class, prefixed by "MC"

It is also necessary to agree on basic names. In the examples above, we see Calo***Digit*** and OT***Digi***. We should converge on one. *Digit* seems more natural, without too much extra typing....

Implications

The choice of inheritance has implications for the way in which such objects are created and copied. It is a requirement that a reconstruction program or a calibration task should not make a distinction between real data and MonteCarlo data, and should therefore only deal with the real data base classes. An implication of this is that when such a task needs to make a copy of these objects, it needs to do so in such a way as to preserve the MC information, even though it doesn't know that the MC information is there! This can be achieved using the *cloning* pattern. The Gaudi team should provide an example of this. For creation, the Gaudi team should provide an example factory to be used by everybody.