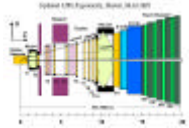
LHCb Event Data Model

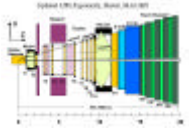
Pavel Binko
Gloria Corti
LHCb / CERN



Outline

A mix of what is designed and/or implemented both in Gaudi and in private detectors' code. It is expected to evolve (of course) …

1. Tree Structure in Stores
2. LHCb Containers and Contained Object
3. Top Level Event Structures
4. Monte Carlo Event Structures
5. Raw and Reconstructed Event Structures
6. Physics Analysis Event Structure
7. Detectors Event Structures: Velo, Tracking, RICH, Calorimeters, Muon
8. Conclusions



Tree Structure in Stores

- Objects in stores are arranged in a tree.
  - Event Root branching sub-trees

    `/Event`

    `/Event/MCEvent` **`/Event/FE (used by L1)`**

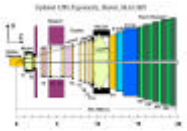
    `/Event/RawEvent`

    `/Event/RecEvent`

    `/Event/AnalEvent`
  - Defined in **`LHCbEvent/TopLevel/EventModel.h`** and **`.cpp`**

- Retrieving (storing) an identifiable object from the store is based on the knowledge (definition) of its **logical path**

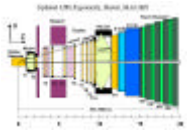
```
SmartDataPtr<MCParticleVector> particles
  ( eventSvc(), "/Event/MC/MCParticles" );
```

LHCb Containers

- Containers are the elementary unit of identification
  - Have to inherit from class DataObject
    - It is assumed that algorithms do not identify individual hits or tracks, but always containers of hits, containers of tracks, etc.
  - Are based on STL containers - implement the same interface
  - Provide containment of pointers to the physics object
    - **MCParticleVector (equal to ObjectVector<MCParticle>)**

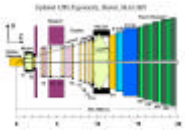
- Requirements on LHCb containers
  - Physics objects must not be contained in more than one container
    - Therefore all classes of objects, they are allowed to be contained in LHCb containers, have to inherit from the base class ContainedObject
  - Memory management of contained objects
  - Navigability in both directions
    - From the container to its object, and from the object to its container



Contained Objects

- Class ContainedObject
  - Provides the back pointer from the contained objects to its parent container
  - Provides the exclusive ownership of a physics object by a container
  - Helps the containers to provide safe memory management

- Containers together with the class ContainedObject provide extended data management
  - They manage the pointers they contain
  - In addition they manage the objects these pointers point to
    - No freely flying objects in the stores

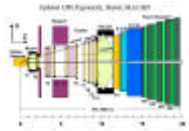


Top Level Event Structures

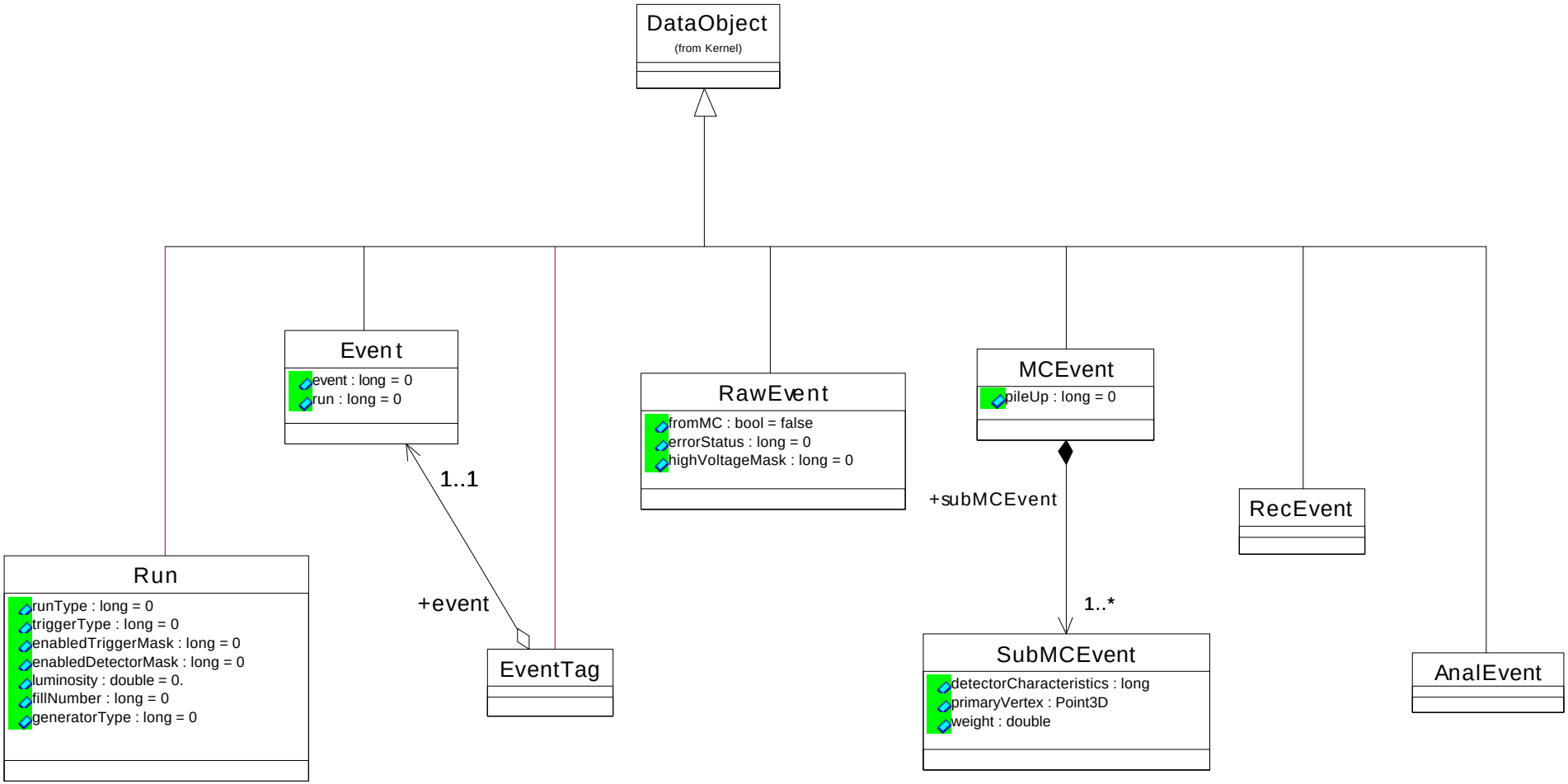
- Classes with global run and event information
  - Run, Event, EventTag, MCEvent, SubMCEvent, RawEvent, RecEvent, AnalEvent (runType, runNumber, triggerType, luminosity, fillNumber, eventNumber, timeStamp, generatorType, highVoltageMask, etc.)

- Most of the these classes have to be identifiable, therefore they have to inherit from the class DataObject

- In the directory /LHCbEvent/TopLevels there are also the LHCb container classes ObjectContainerBase, ObjectVector, ObjectList and the base class ContainedObject



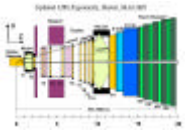
Top Level Event Diagram



Monte Carlo Event Structures

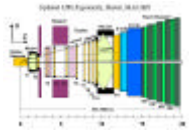
- The MonteCarlo event sub-tree contains output from the event generators and MonteCarlo tracking

- MCParticle, particles “seen” by the spectrometer (GEANT3)
  - contains the data members: fourMomentum, particleId, and references to its origin vertex (SmartRef to MCVertex) and its “decay” vertices (SmartRefVector to MCVertex)
  - **Path = “/Event/MC/MCParticles”**
- MCVertex, production and “decay” (end) vertices of MCParticles (GEANT3)
  - contains the data members: position, timeOfFlight, and references to its mother track (SmartRef to MCParticle) and its daughter tracks (SmartRefVector to MCParticle).
  - **Path = “/Event/MC/MCVertices”**

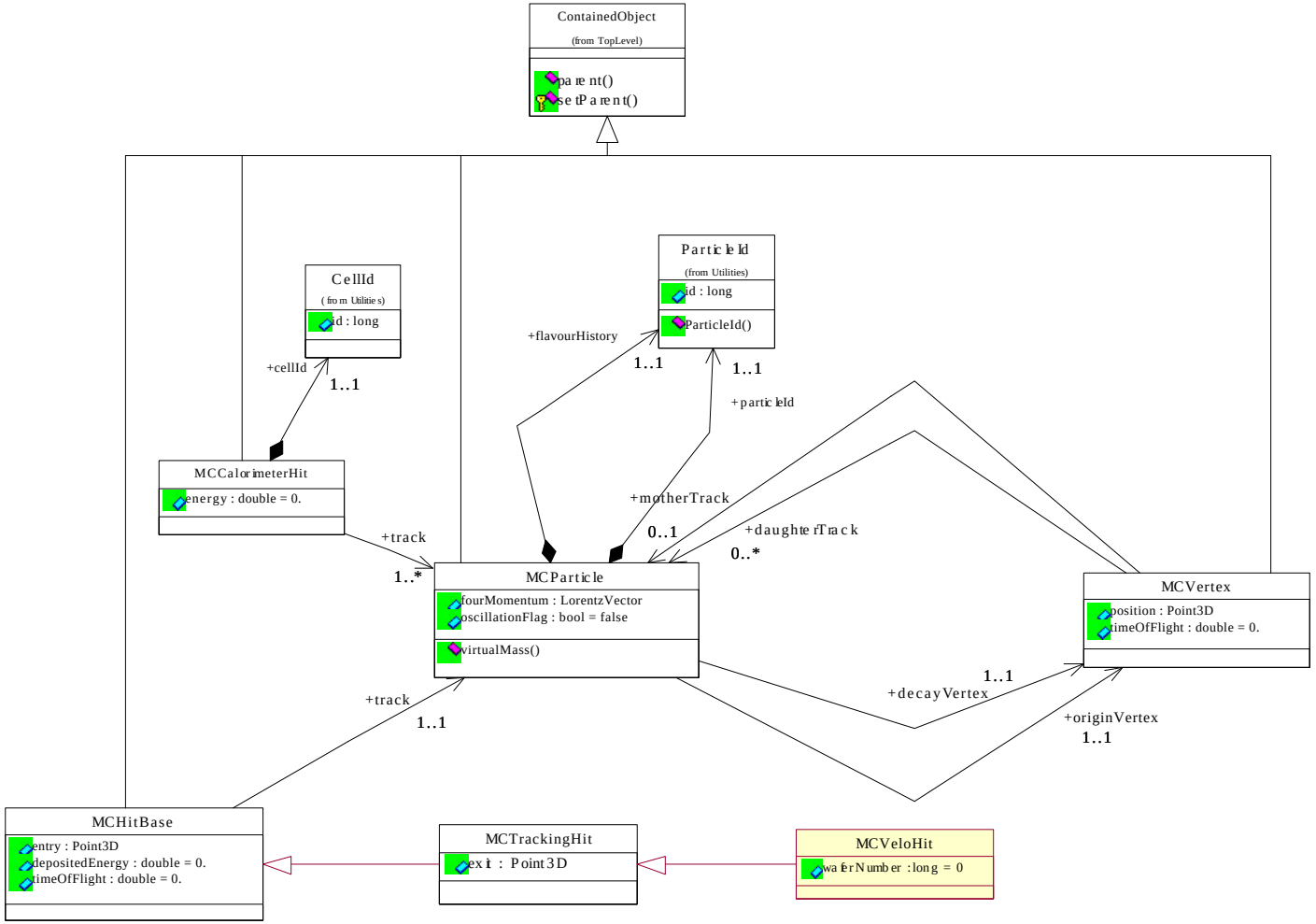


Monte Carlo Event Structures: Detectors' MC hits

- MCHitBase is a base class for many concrete hit classes
  - contains data members entry (the entry point), depositedEnergy and timeOfFlight, and the reference to MCParticle, which are common to many concrete hits, which inherit from it.
  - It is not allowed to instantiate MCHitBase, as it has no physical meaning (only the inherited classes correspond to real hits).
- MCTrackingHit is used in 2 contexts: tracker and muon detector
  - inherits from MCHitBase, in addition it contains the data member exit (the exit point).
  - **Path = "/Event/MC/MCTrackerHits", "/Event/MC/MCMuonHits"**
- For other detectors see details later

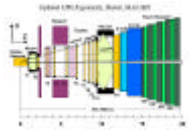


Monte Carlo Event Diagram

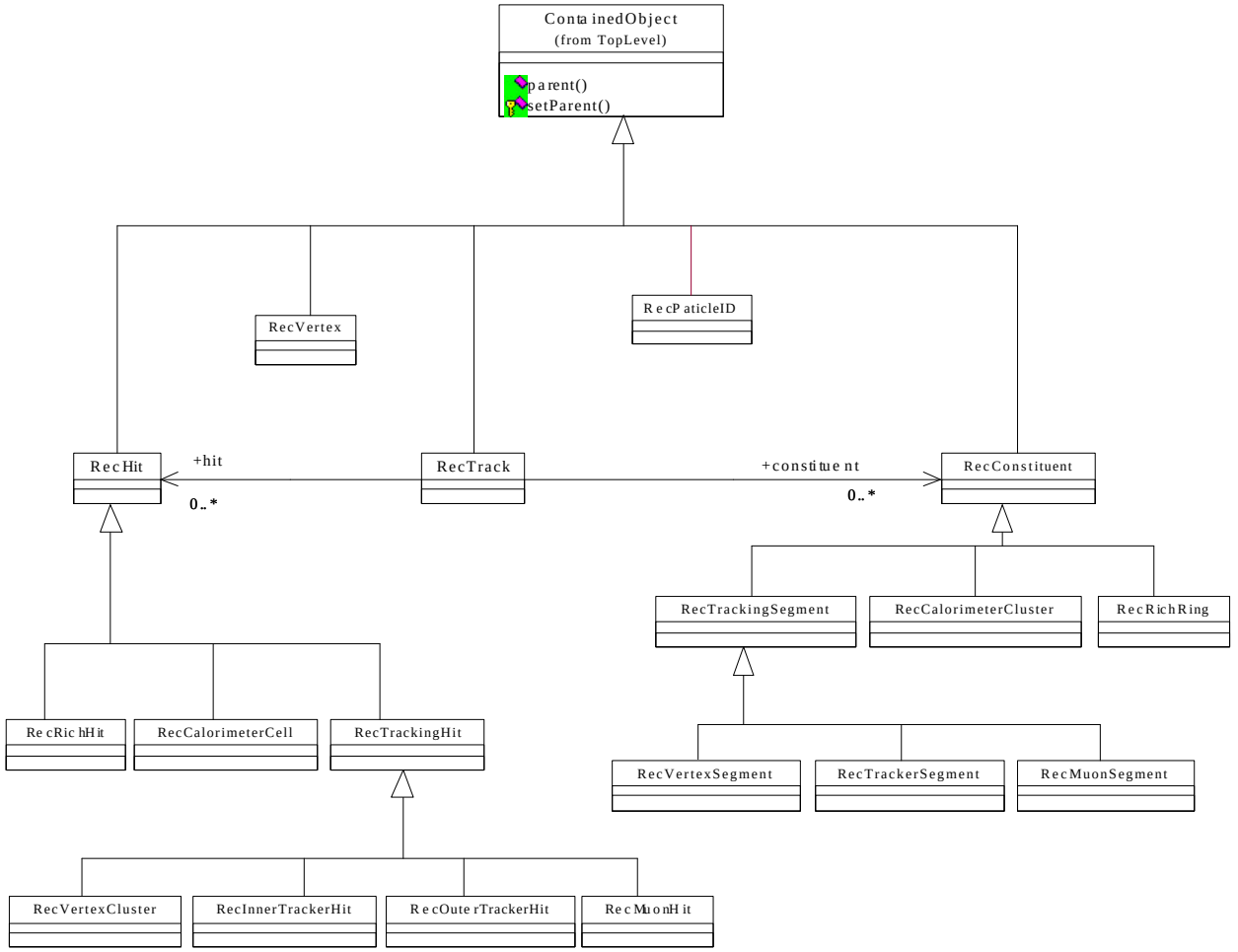


Raw & Reconstructed Event Structures

- The RAW event sub-tree contains raw data collected by DAQ and produced by simulation in same format (i.e. detector and electronics response).
- Reconstructed event sub-tree contains the output of the reconstruction
- See details later for the different sub-detectors
- Necessary to combine information from different sub-detectors in the reconstruction

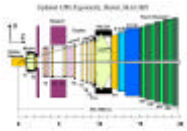


Reconstructed Event Diagram

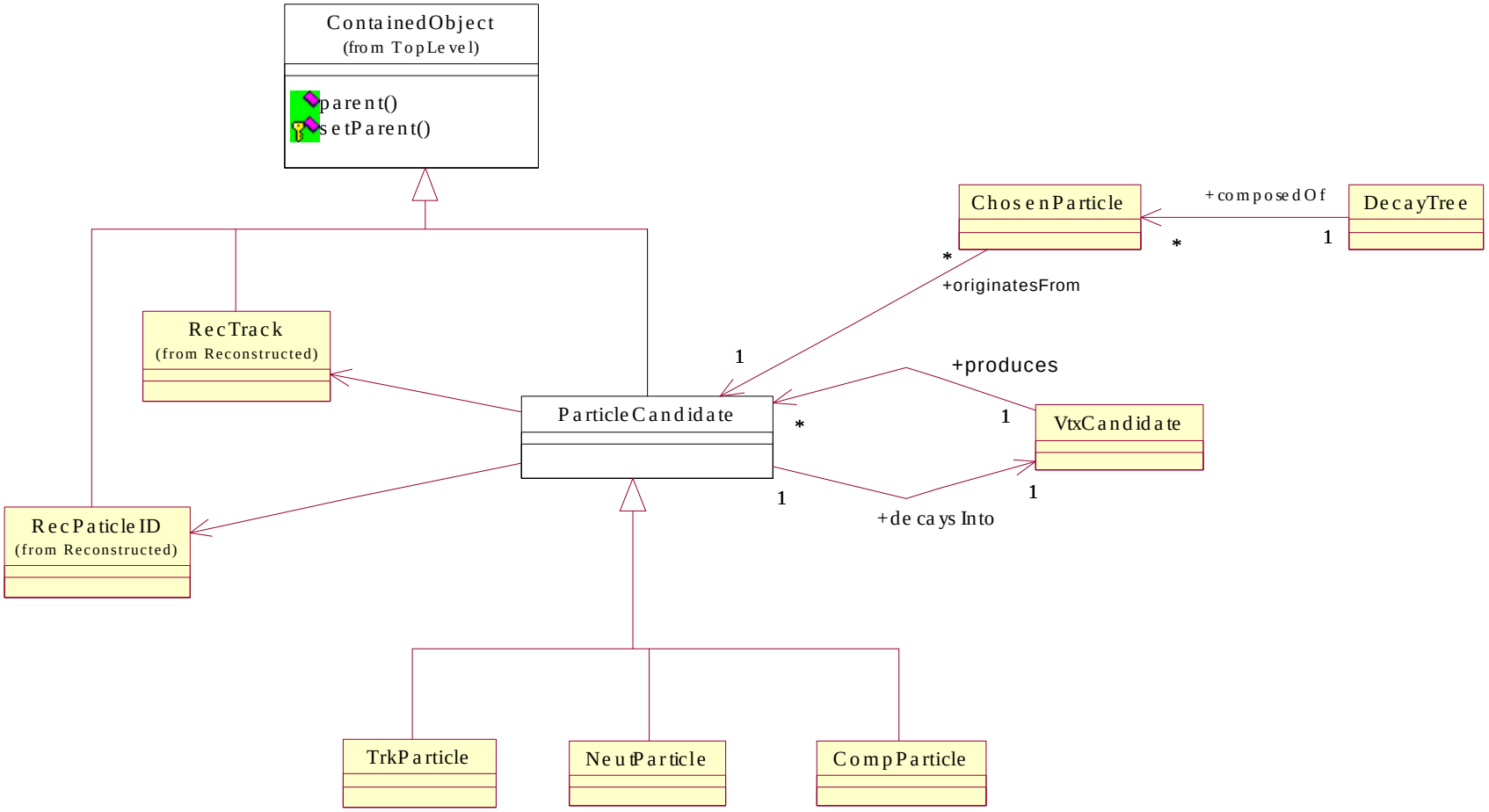


Analysis Event Structures

- Contains objects created and used during physics analysis.
- In the Gaudi release only the AxPartCandidate class, derived from the SICb AXTK bank is implemented.
- Design, partial implementation exists in private code:
  - ParticleCandidate base class with pointers to some reconstruction classes, three-momentum, measured mass for composite particles and specialized classes for different particle types (TrkParticle, NeutParticle, CompParticle), as well as reference to decay vertex
  - VtxCandidate contains position,error matrix, χ^2 and reference to particles (pointers to ParticleCandidate class) that were used to “make” the vertex
  - ChosenParticle class with pID chosen for a specific analysis, and hence four-momentum
  - DecayTree where the full decay is described with references to ChosenParticles

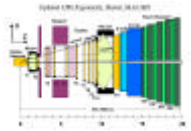


Analysis Event Diagram



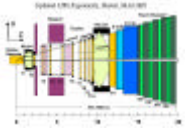
Velo / L1 Event Model

- Currently Velo/L1 SICb banks mapped to classes
- Two separate world: pure MC world, MC/real data world (Raw, Rec, FE), connected trough reference tables
- MCVeloHit (Geant3 hits)
  - inherits from MCTracking hits, in addition contains wafer method
  - **Path = "/Event/MC/Velo/Hits"**
- RawVeloHit (digitized hits)
  - sector, type (R,ϕ), stripNumber, charge, adcCount, waferNumber
  - **Path = "/Event/Raw/Velo/Hits"**
- VeloClusters (reconstructed R, ϕ clusters)
  - coordinate,station,pulseHeight,width, etc, + references to RawVeloHit
  - **Path = "/Event/Rec/Velo/RClusters", "/Event/Rec/Velo/PhiClusters"**

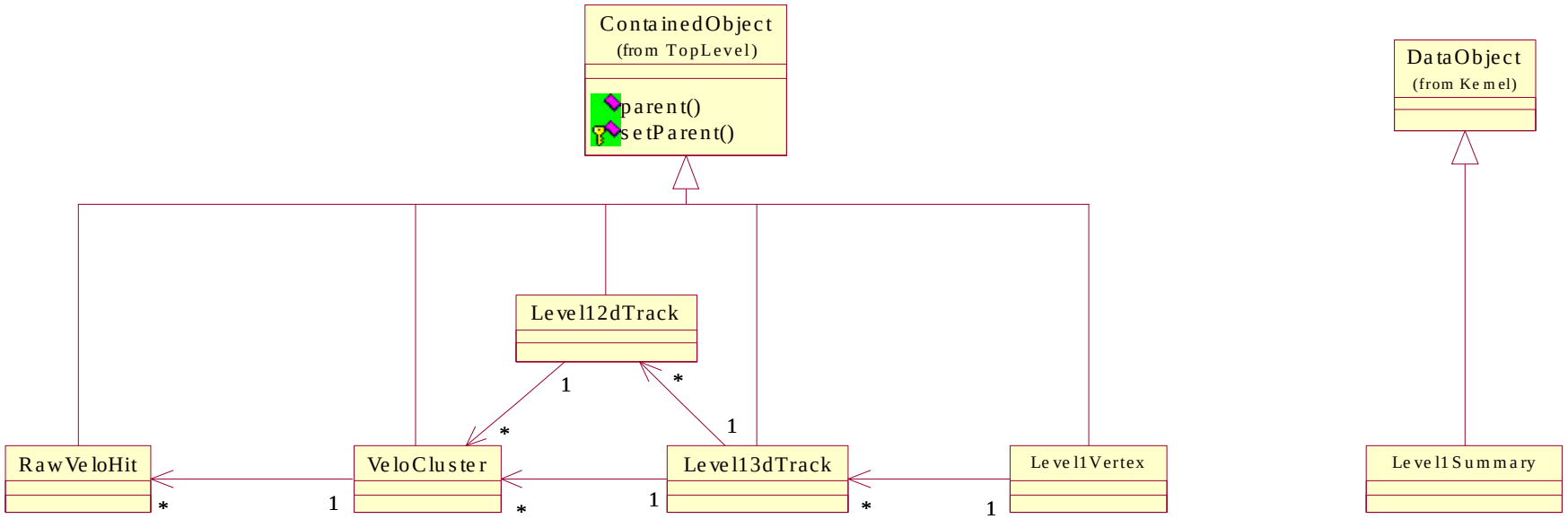


Velo / L1 Event Model

- Level12dTrack (Level1 vertex trigger 2-d tracks)
  - status, slope,radius, etc + references to VeloCluster
  - **Path = "/Event/FE/L1/2dTracks"**
- Level13dTrack (Level1 vertex trigger 2-d tracks)
  - coordStart, uvecStart, ip, etc + 2 vectors of pointers to VeloClusters and a vector of pointers to Level12dTrack
  - **Path = "/Event/FE/L1/3dTracks"**
- Level1Vertex (Level1 trigger primary and secondary vertices)
  - **Path = "/Event/FE/L1/PriVertices", "/Event/FE/L1/SecVertices"**
- Level1Summary (trigger decision, like event trigger tag)
  - decision, pileup, numLargeIPtracks, singleVtxProb, etc.
  - **Path = "/Event/FE/L1/Summary"**

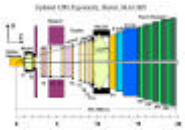


Velo / L1 Event Model Diagram



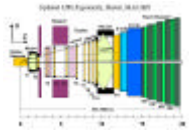
Tracking Event Model

- Data Model used by Tracking Algorithms
  - OT stands for outer tracker, IT for inner tracker
- OTDigi, ITDigi, classes with digitized raw data (i.e. TDC counts)
  - currently they are produced starting from the Gaudi Classes RawInner(Outer)TrackerMeas (copies from SICb banks)
- OTHits, ITHits, hits produced by the subdetectors reconstruction, input for tracking
- OTHitsOnTrack, ITHitsOnTrack, measurements assigned to tracks
  - inherits from base abstract class TrMeasurement

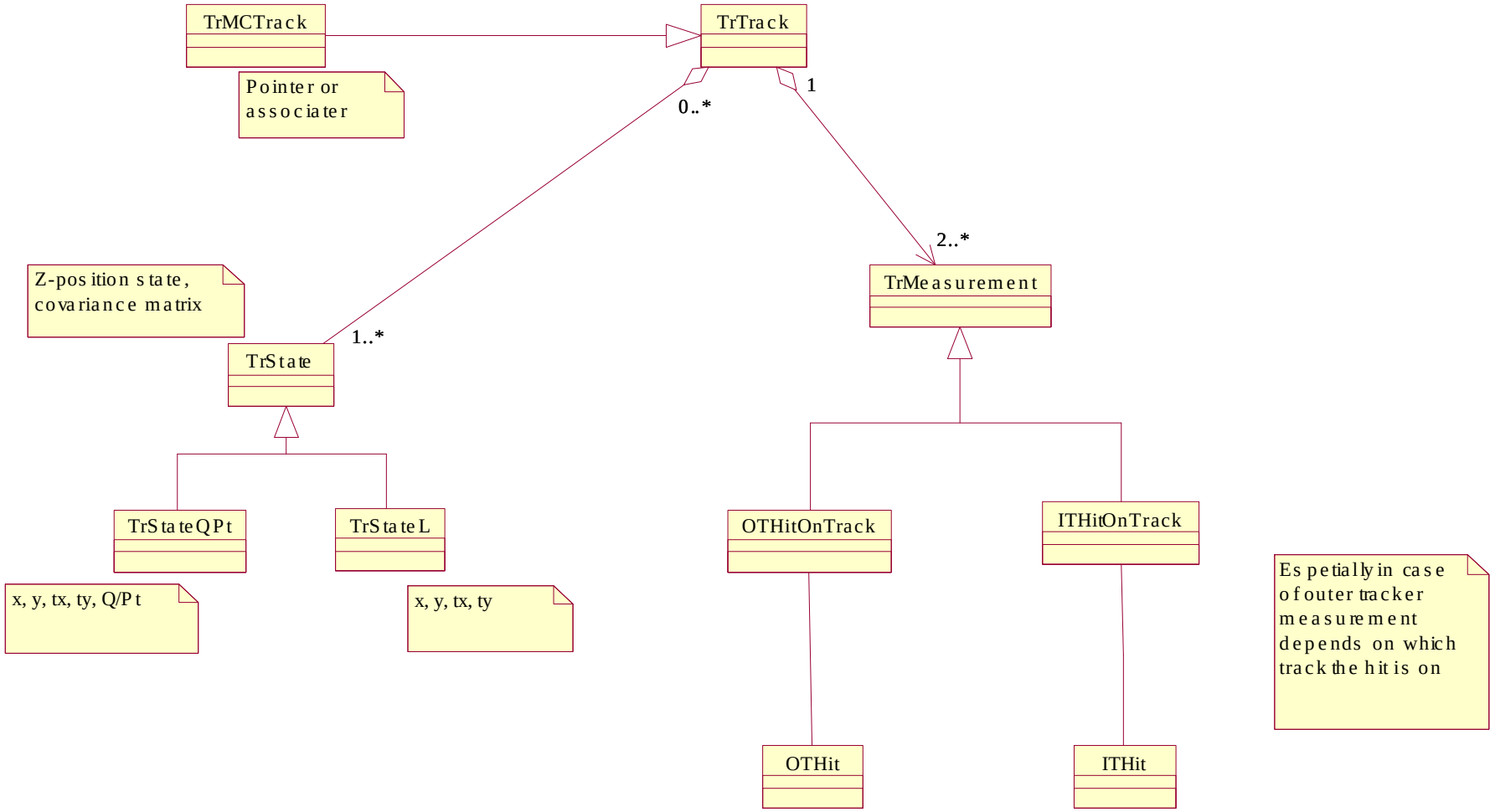


Tracking Event Model (2)

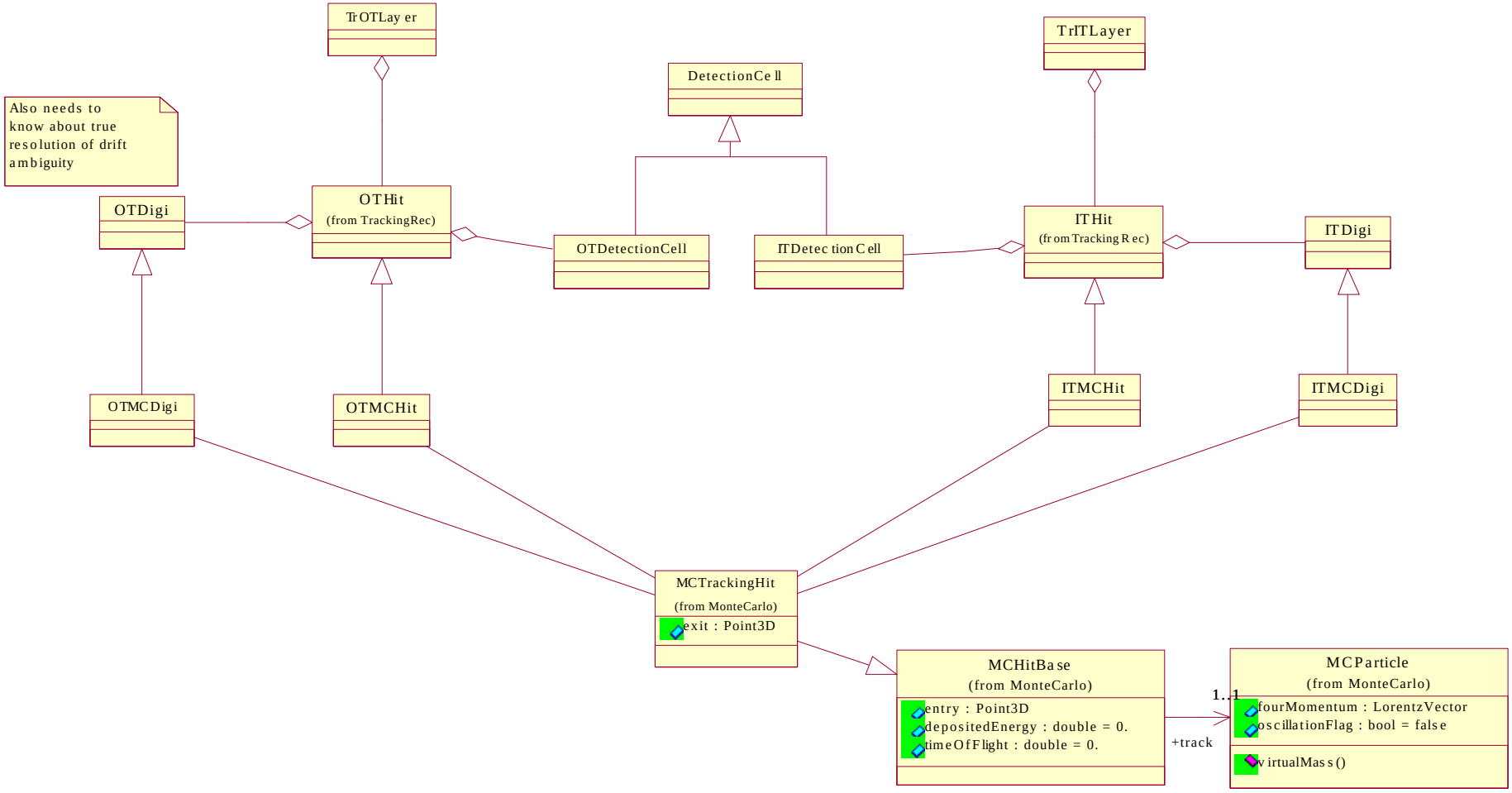
- TrState, a snapshot of a track
  - track parameters and covariance matrix at a given z position on its trajectory
- TrTrack, contains information about the track accumulated during tracking
  - list of pointers to TrMeasurements and to TrState
  - charge, χ^2 , particleType
- Connections with the MC world are done through inheritance for hits (ex. ITMCHit, ITMCDigi), inheritance with pointer or associators for tracks



Tracking Data Model for Tracks



Tracking Data Model for Hits

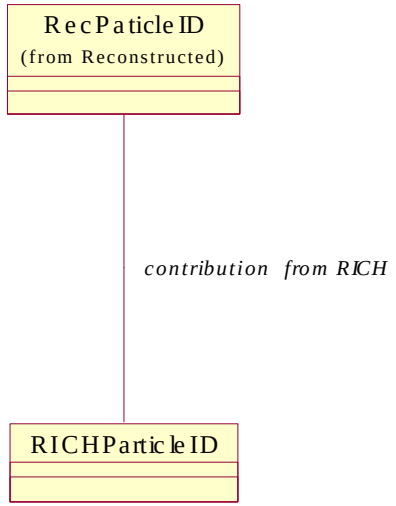


RICH Event Model

- At the moment the RICH Algorithms have their own private event and communicates with the LHCbEvent via an Event Adapter.

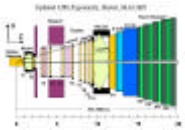
- Different adapters for the different purposes: simulation, reconstruction with simulation tracks, reconstruction

- The Event adapter creates RICH input objects (Track and TrackSegment) from objects in the LHCbEvent transient store (i.e. MCParticle, TrTrack) and deposit its result in the store with a pointer to the original object so that it can then be used by other algorithms (ex. of RICHParticleID in RecEvent)

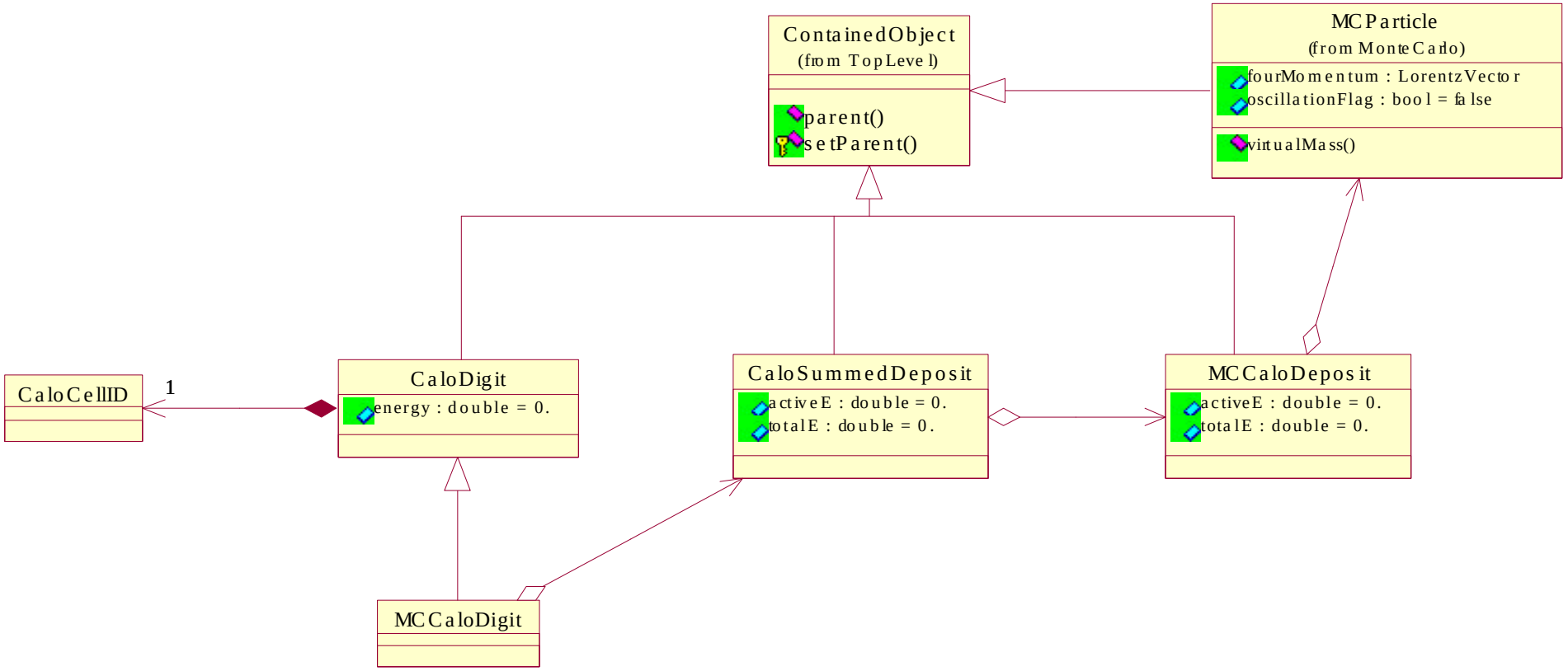


Calorimeters Event Model

- Simulation, Digitization, Calibration and Clusterization Algorithm for the Calorimeters “communicate” through the data flow from one algorithm to the other
- MCCaloDeposit, energy deposited by a MC particle
- MCCaloSummedDeposit, energy deposited in the Calorimeter active material
- CaloDigit, energy deposition in a given Calorimeter cell
- MCCaloDigit, inherits from CaloDigit + references to MC true info (MCCaloSummedDeposit and MCCaloDeposit)
- CaloCluster, reconstructed clusters (x,y,E, etc.)
  - Same class holds MC and real Data (references to CaloDigit)
- Simulation produces a sequence of MCCaloDigit, that the Digitization either updates or uses to create a new sequence. The Calibration and Clusterization treat the sequence of MCCaloDigits as a CaloDigit sequence (dynamic casting is necessary when checking against MC truth)



Calorimeters MC and Raw Data Model

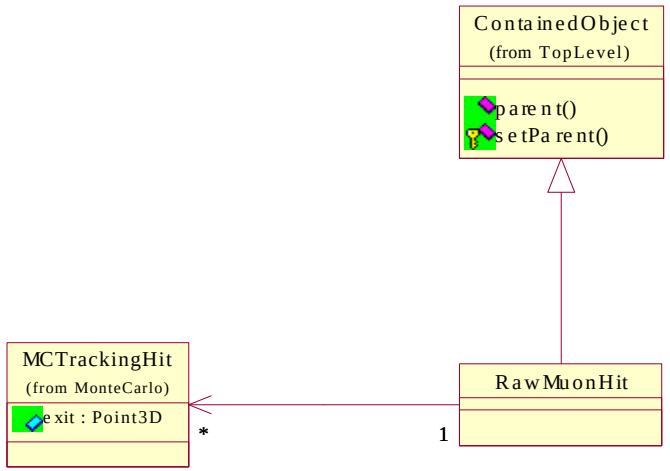


Muon Event Model

- A simple digitization algorithm is designed and implemented within the Gaudi framework. Simple Event Model related to digitization algorithm (input/output data)

- MCTrackingHit are used as input(/Event/MC/MCMuonHits)

- RawMuonHit are produced (stored in /Event/Raw)
  - full PadID (station, chamber, pad) and time stamp + vector of pointers to MCTrackingHit



Conclusions

- The Event Model in the sub-detectors mostly specialized to their algorithms. But data objects in the event store are how the sub-detector algorithms exchange information.
- How can we have a coherent global Event Model?
- The Data Model from the sub-detectors should be integrated in the LHCbEvent Model (after testing, of course...). Maybe some sub-detector data classes could be common base classes or be specialized in other sub-detectors...
- Need for concrete (specialized) containers. What are the common requirements. Necessary to collect information: workshop?