Overview of trigger EDM and persistency

Focusing on the e/γ slice

Referring to the work of several people

Motivation:

Trigger-specific

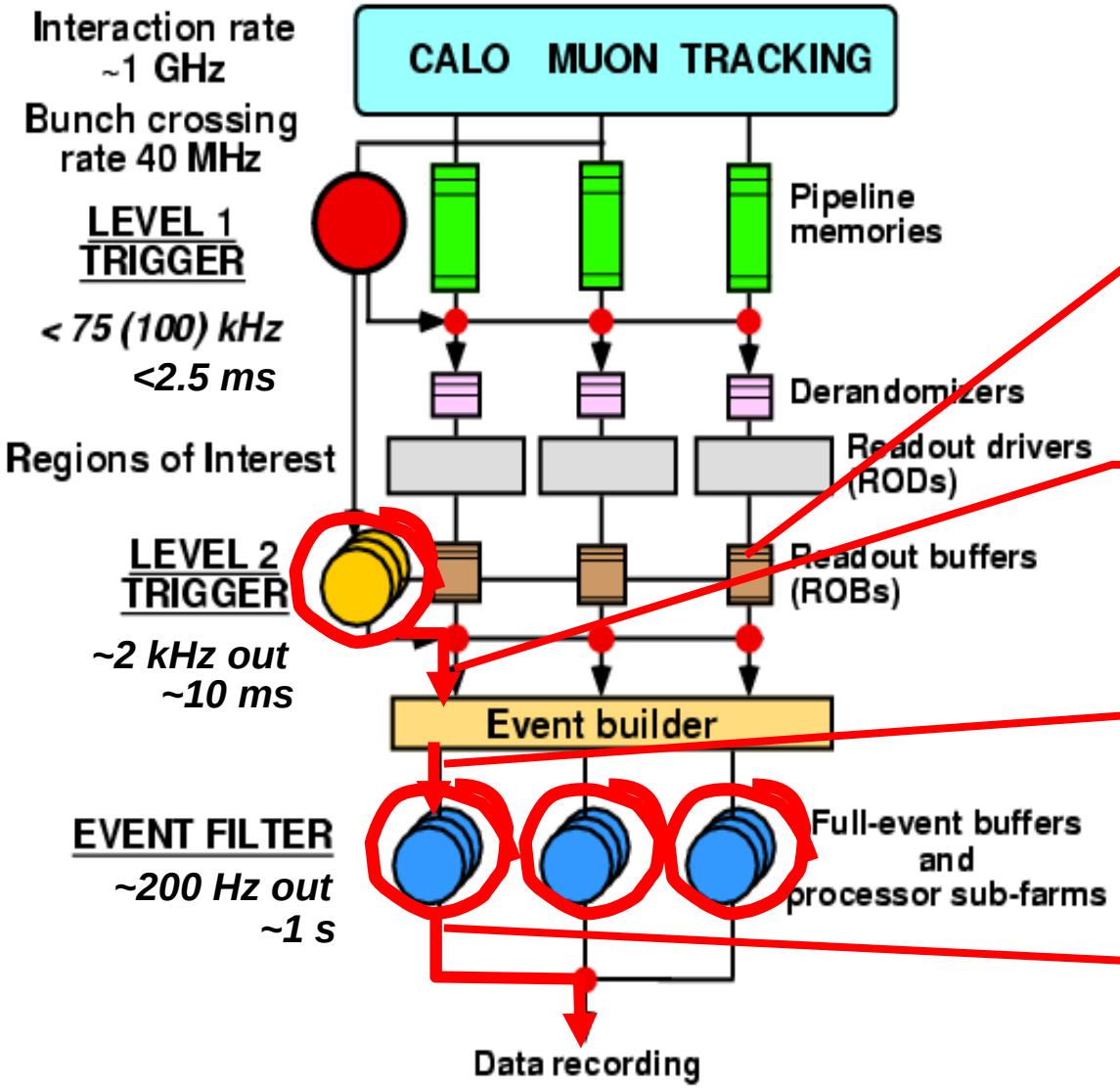
- We will need some (**redundant**) trigger information available offline for **debugging**
- Need well defined ways of communicating between LVL2 and EF (through bytestream)
- Need to make it easy to **develop/debug** Feature Extraction algorithms and **optimize** Hypothesis algorithms
- Not much information needs to be kept in normal running (but important to be able to store **a lot** of debugging information)
- Trigger slices need input from physics "users" to have optimal menus

Physics analysis

- Physics analyses need input from trigger to be **realistic** (i.e. trigger info in **AODs**)
- This may mean several different things:
  1) "Yes/No" result of hypothesis algorithms only: probably good enough for most physics analyses; would generate valuable **feedback** from physics groups
     - ☐ **See TriggerDecision below**
  2) Enough information to allow some **tuning of cuts** in hypothesis algorithms: must include navigation information; even more valuable feedback from physics groups; allow development of new trigger menus
     - ☐ **See the Serializer below**
  3) Everything (…this means running trigger from RDOs; **not feasible** for physics analysis)

We must think in terms of what we need to run over real data

Sources of event data from the trigger



TDAQ data flow

Detector data - raw event fragments

Level 2 data appended to raw event

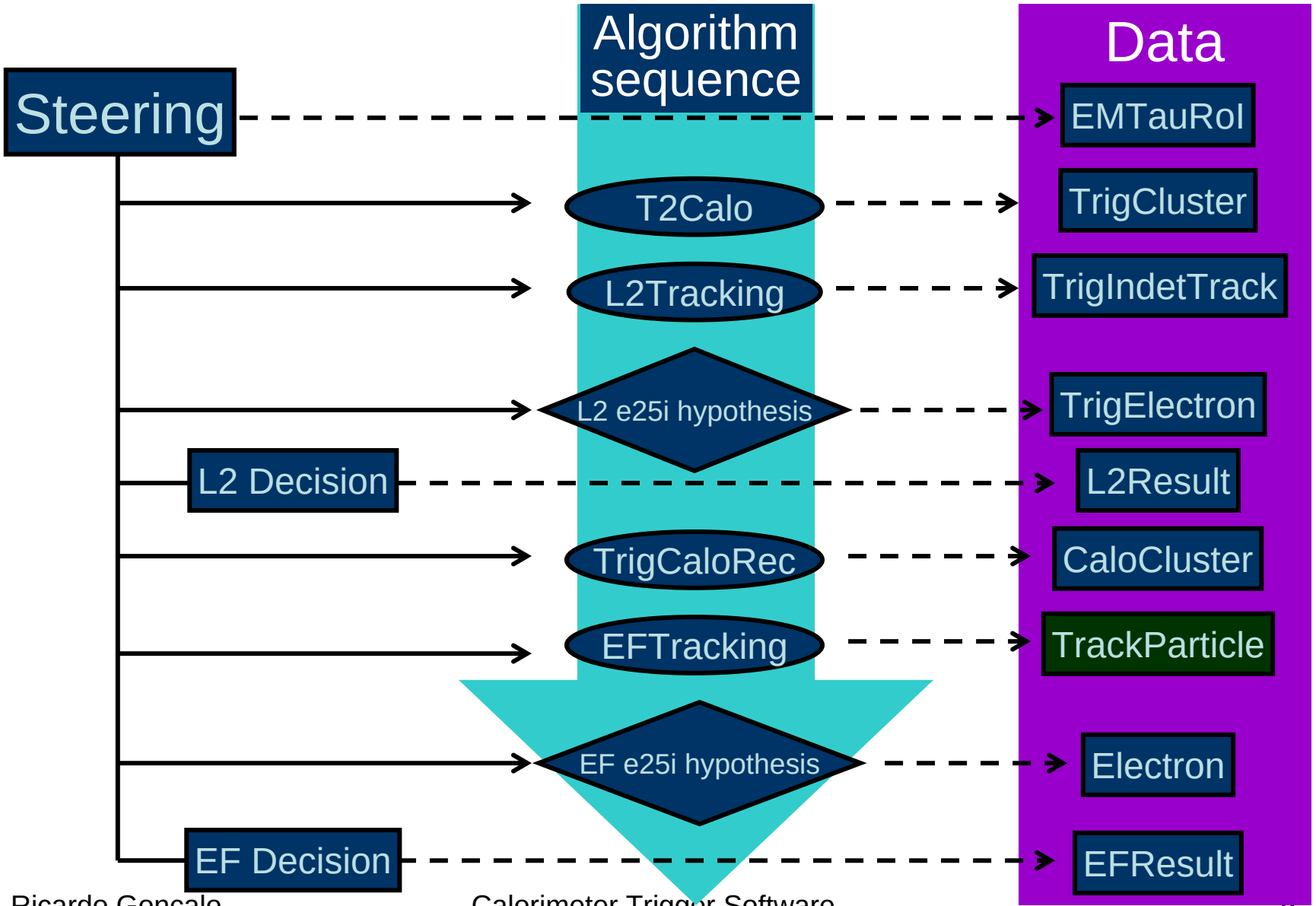
Event passed to EF, uses LVL2 data to seed

EF data appended to raw event

Existing trigger data objects

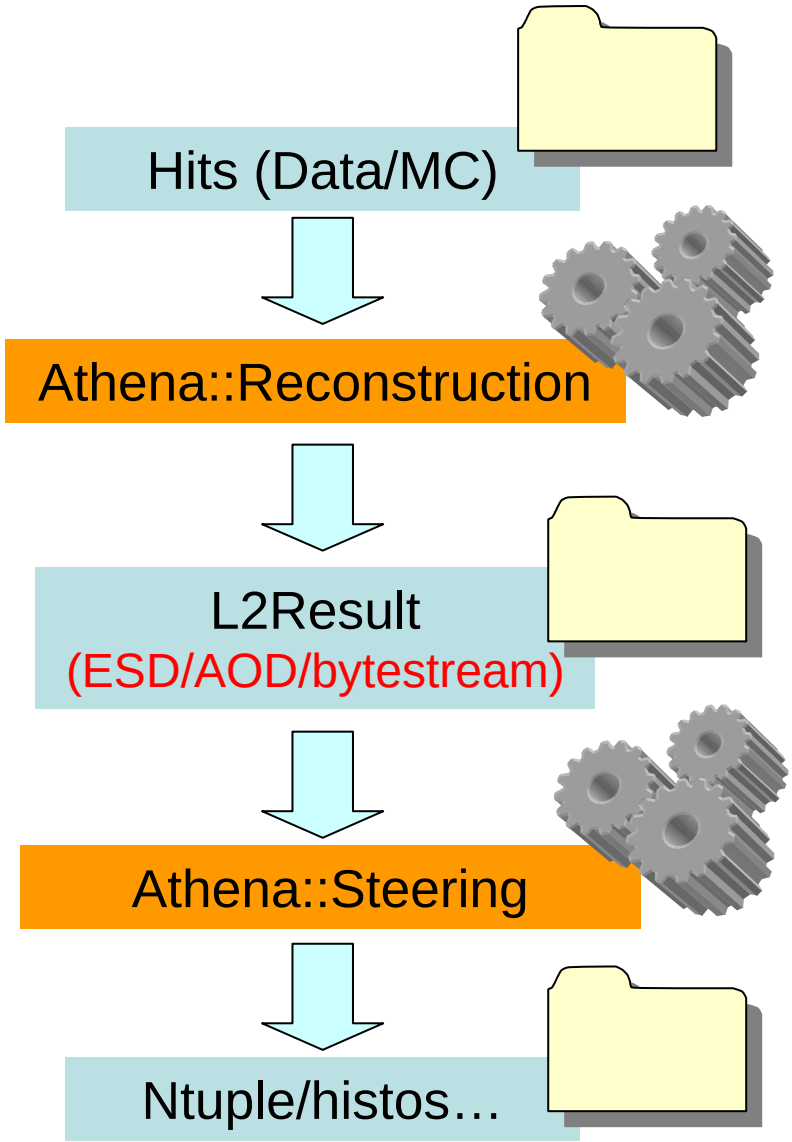
- Level 1: (See Alan's talk yesterday)
  - `TriggerTower`, `JetElement` (AOD/ESD)
  - `L1EmTauObject`, `L1JetObject`, `L1EtmissObject`
  - `EmTau_ROI`, `JetET_ROI`, `EnergySumROI` (soon in ESD/AOD)
  - `CTPDecision` is in `TriggerDecision`
- Level 2:
  - `TrigEMCluster` and `TrigTauCluster` to replace `EMShowerMinimal` for 11.0.5 (See Denis' talk today)
  - `TrigInDetTrack`
  - `TrigParticle` (`TrigElectron`)
- Event Filter: (see Cibran's talk today)
  - `CaloCluster` from `TrigCaloRec/egamma/Tau`
  - `Trk::Track` (ESD) `TrackParticle` (ESD/AOD)
  - `egamma`
  - `TriggerDecision` (should be updated at each level)

Data produced by the HLT – e/γ example



Producing and running from ESD/AOD

- New Steering functionality allows for objects with a SEAL dictionary to be **serialized** into L2Result and EFResult (see Gianluca's talk on monday)
- These are simply std::vector<int>
- These can be put in ESD/AOD
- For LVL2:
  1. When running on **simulated data**, the Steering **serializes** all relevant data objects into **L2Result** (in ESD)
  2. To do this, the Steering runs only FEX algorithms (which produce the data)
  3. When running from ESD, the Steering **de-serializes** data objects and runs only HYPO algorithms
- In this way, the hypothesis algos can be run many times over the same data objects as if running online, and the cuts **optimized**



Status of the Serializer

- Simon, Gianluca, Jiri
- Uses Reflection library to serialize classes into `L2Result` and `EFResult` in bytestream or POOL
- Works with:
  - Simple native types (**int**, **double**, **float**)
  - **Pointers** (and **NULL** pointers)
  - Follows (non-NULL) pointers
  - Classes need to have **SEAL** dictionaries (same requirement for POOL)
  - All unsupported class data members should be declared **`transient`** (in `selection.xml`)
- To do:
  - Store references to POOL objects (e.g. to write **`EFResult,`** where objects are **not** serializable) will be there from 11.0.5
  - STL containers (std::vector<>, DataVector<>)
  - Schema evolution
  - For 11.3.0 (in ~2 weeks) switch from Reflection to Reflex

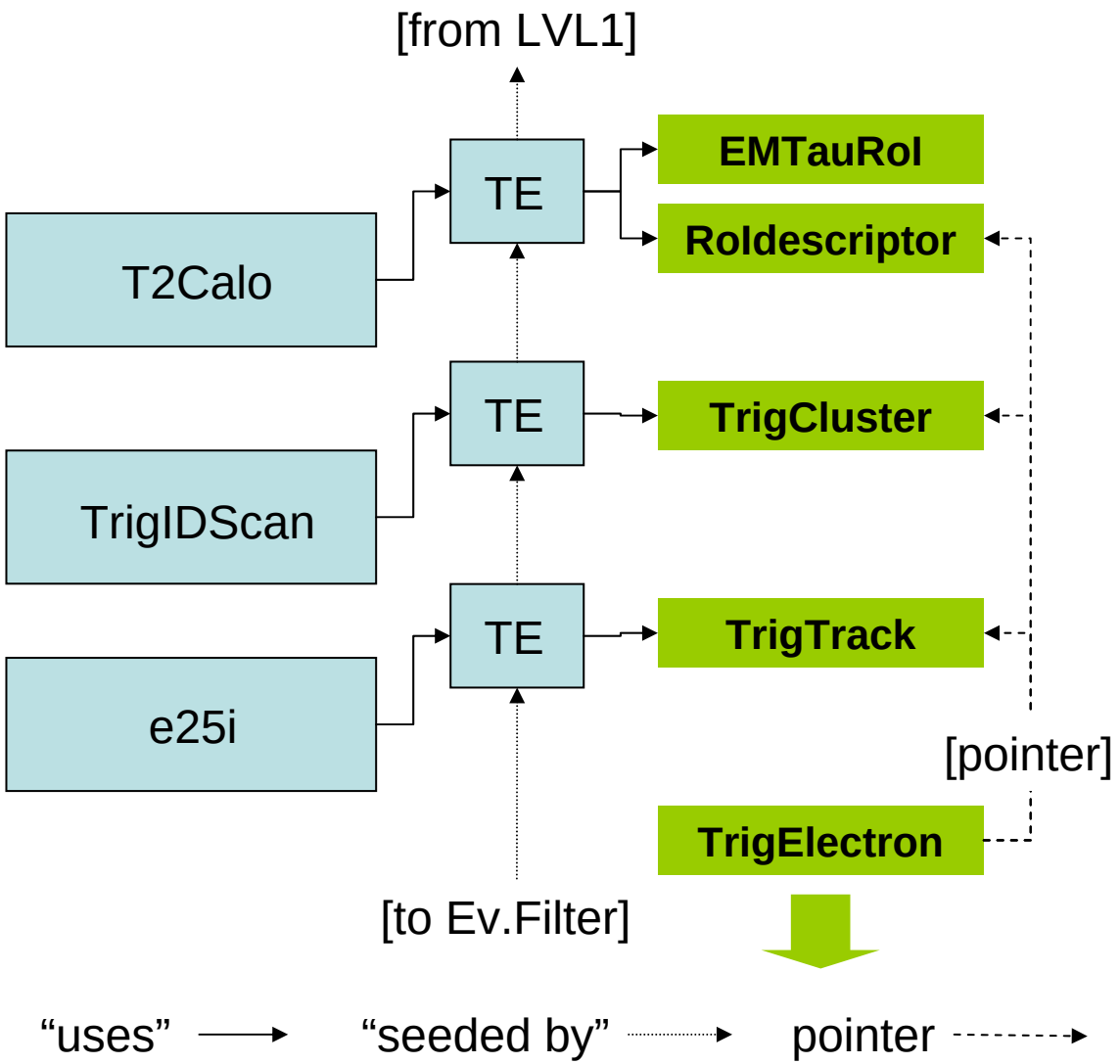
Level 2: TrigParticle

- To store the candidate object that was accepted by a signature
- Should be light, with no ElementLinks or heavy inheritance, to ease persistency
- Example:
  - **`TrigElectron`**
  - Summary data to use for **debugging** and **analysis**
  - **`TrigElectron`** data members:
    - Roi_Id
    - eta, phi
    - Z vertex
    - p_T, E_T
    - **pointer** to track
    - **pointer** to cluster

    (“best estimate” values from HLTHypoAlg)

```
class TrigElectron {
 public:
    TrigElectron();
...
    TrigTrack*   track(int i);
    TrigCluster* cluster();
    int RoI();
 private:
    int              m_roi;
    double           m_eta;
    double           m_phi;
    double           m_z_vtx;
    double           m_p_T;
    EMShowerMin*     m_cluster;
    vector<TrigTrack*> m_trk;
};
```

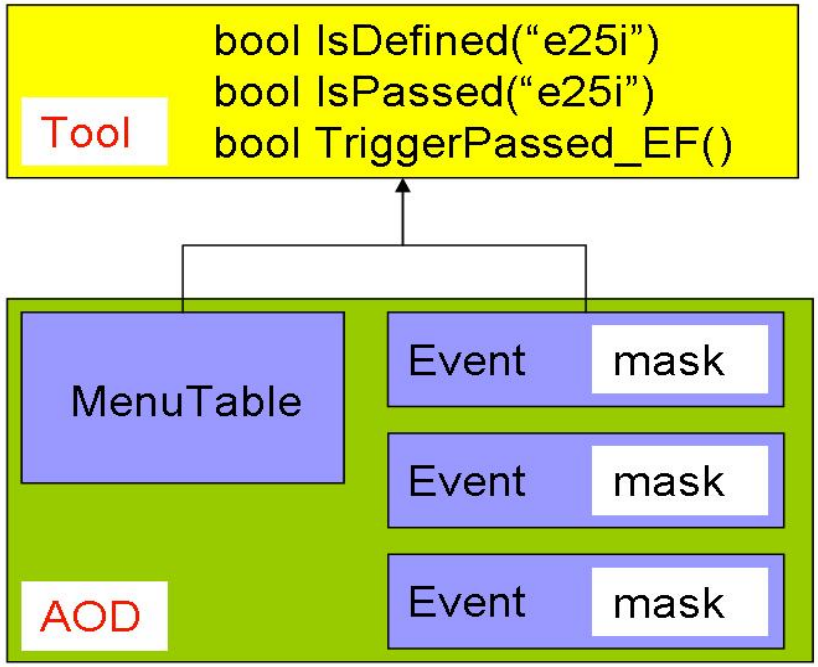
TrigParticle: status



- Only TrigElectron so far
- To do:
  - Replace EMShoweMinimal with TrigEMCluster
  - Inherit from Inavigable4Momentum
  - Test persistency with serializer
  - Write example Hypo algorithm
- Can be instantiated from ESD/AOD

New steering features made TrigParticle less important

Trigger Decision: yes/no result



- Signatures passed/failed/prescaled encoded in a **bit pattern** stored once per event
- Bit pattern **interpreted** through MenuTable (in conditions DB or in RunStore…)

- A Tool would provide the user interface to L1/L2/EF and individual signature results by interpreting bit patterns in AOD. It would give:
  - Decision bit for each signature
  - Access to trigger configuration through methods like isDefined()

TriggerDecision: status

- **Short term implementation**: while there are only a few signatures
  - Store object in AOD consisting basically of map:

    ```
    map<string label, bool accept>
    ```

  - Derive trigger decisions from Hypothesis algorithms
  - Only a few signatures - wasted AOD space by repeating labels each event is negligible

- **To do:**
- TriggerDecision has been implemented (Monika)
- Need algorithm to fill TriggerDecision for a given menu (don't need /can't have machinery to deal with generic menus for now)

Other loose subjects

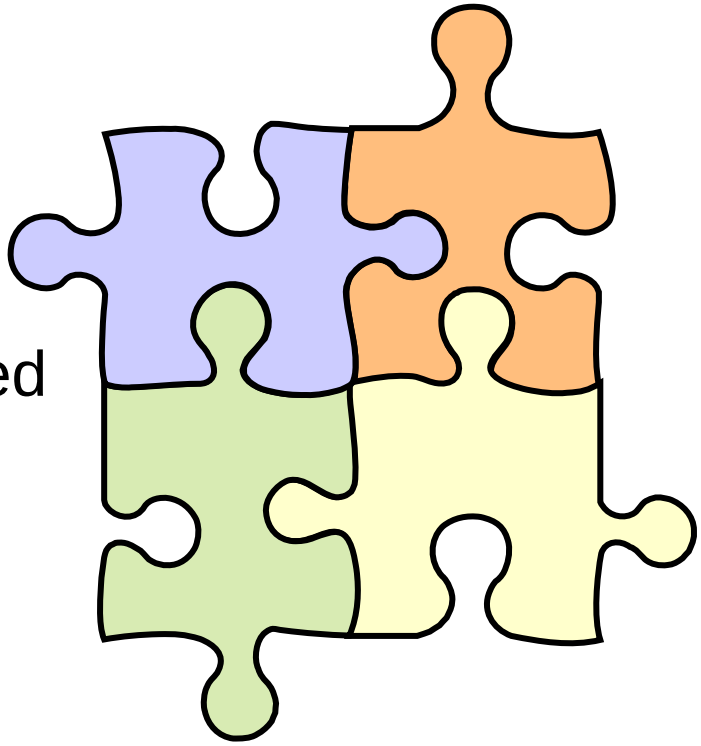
- Level 1:
  - Schema evolution: hasn't been a concern so far because trigger not run by default in productions: this should change soon
  - Thresholds passed/trigger configuration
- Level 2:
  - TrigInDetTrack truth association: under development
  - Until Serializer can deal with STL containers, using a few tricks that should be removed
- Event Filter:
  - Several classes being adapted from offline (Iwona)
    - TrackParticle-truth association
    - VxVertex
- The size of all this:
  - LVL2 objects very small compared to offline
  - EF objects **are** offline (and we're only reconstructing a fraction of the event)

Plans for release 12

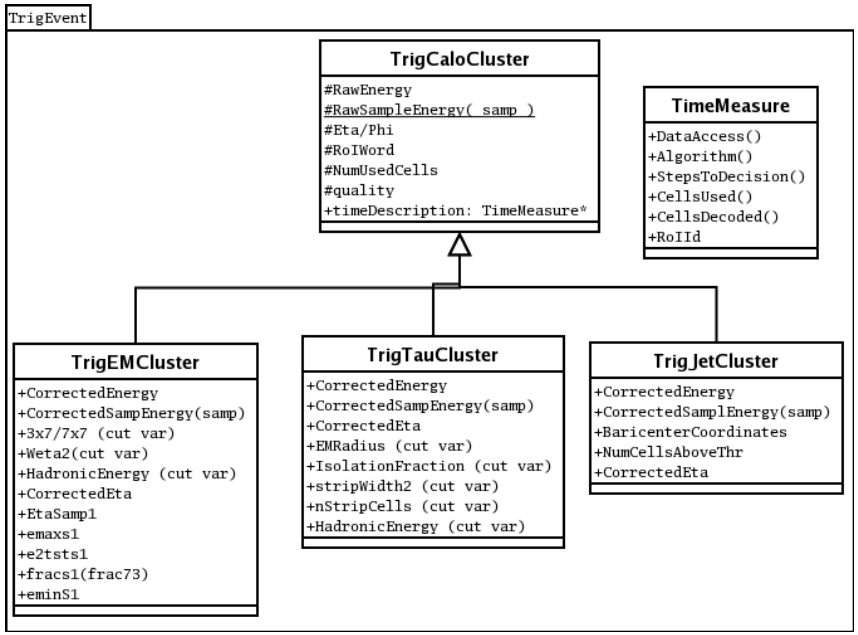
- **Wish list** (e/γ specific..):
  - New calo EDM (**`TrigEMCluster`**,..): serializable, tested
  - **`TrigInDetTracks`**: serializable, tested
  - **`TrigInDetTrack`** association: (**`TrigInDetTrackTruth`** and **`TrigInDetTrackTruthMap`**) storable in POOL and tested
  - TrigParticle (**`TrigElectron/TrigPhoton`**?): hypothesis algorithms to fill them; example menu configurations
  - **`TriggerDecision`**: configured with a default menu; filled with Steering information and stored in AOD/ESD for end users
  - Test jobs for each of the above
  - Default doTrigger=True in CSC production
- To have the above in **12.0.0** (end of March) should test it in **11.0.5** (end of this month)
- This would allow many studies that were hard to do/impossible up to now
- It would also generate lots of feedback from the physics community
- Note that with also Taus, Muons, Jets, xE_T, the value for physics analysis would increase exponentially

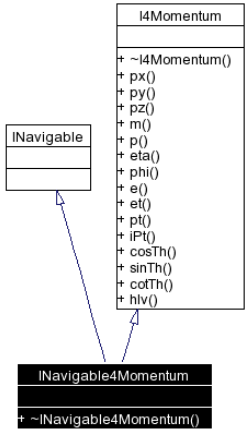
Conclusions

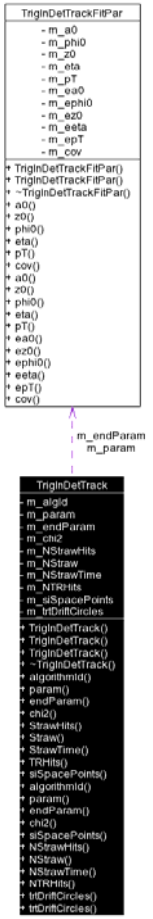
- Trigger EDM and persistency in much better shape than, say, 1 year ago
- Many different pieces in the puzzle are coming together
- Aim should be to have working, tested functionality in place for 12.0.0
- This could allow us to make the a great use of the CSC production for trigger studies



Level 2 e/γ







21 doubles and 5 int per track
Plan to optionally include space points for special trigger studies