Definition of input objects

Requires the definition of a minimal common set of input objects from which reconstruction should start. This depends on the AlgTool type (e.g. track combination, track tagging, ...), as discussed in the following.

A common point to be discussed is how muon tracks should be accessed. Algorithms currently work on TrackParticles, but:

- this means that that code could be rerun at AOD level; anyway if the algorithms are then navigating to the corresponding track, this would mean accessing information at ESD level; so, AOD level operability is just apparent and could be misleading;
- is there any algorithm using information from the TrackParticle which is not accessible from the Track?

In the following I'll suppose that the use of TrackParticles can be dropped; if this is not the case (at least in the short term), the interfaces can be extended with corresponding methods using TrackParticles instead of Tracks.

Definition of output objects

Requires the definition of a minimal common set of output objects.

In this case, all AlgTool types will probably need to save the reconstructed muon candidates as collections of:

- Trk::Track;
- Rec::TrackParticle;
- Analysis::Muon.

This common step, containing no algorithmic intelligence, could be performed by a common tool, to avoid code duplications.
Common tools should also be adopted e.g. for producing a combined track starting from a couple of ID+MS tracks, to ensure that all the information attached to the input tracks is uniformly copied to the output track.

Some algorithms may need to save extra output which is not strictly needed by a generic muon reconstruction chain (e.g. the pair of tracks out of which a combined tracks was built). This could be considered "debug" information and its saving to StoreGate should be configurable (important for EF operation).

Muon tracks combination

Interfaces for Muon Track Combination

```
TrackCollection* MTC::find(TrackCollection* idTracks, TrackCollection* msTracks)
Track* MTC::find(Track* idTrack, Track* msTrack)
```

The first method performs the matching between ID and MS tracks and then calls the second method once per combined track.

Interfaces for Muon Track Matching

The matching between ID and MS tracks could be further separated from the combination part,
This requires an additional tool and a different combination interface.

```
TrackCollection* MTM::find(TrackCollection* idTracks, TrackCollection* msTracks)
```

```
TrackCollection* MTC::find(TrackCollection* matchedTracks)
Track* MTC::find(Track* idTrack, Track* msTrack)
```

Muon tagging with muon segments

Interface for Muon Segment Tagging with already available segments

```
TrackCollection* MST::find(TrackCollection* idTracks, SegmentCollection*
msSegments)
```

Note: unlike the following methods, this requires no additional data access.

Interface for Muon Tagging with Muon Data

```
TrackCollection* MTMD::find(TrackCollection* idTracks, PrepRawDataCollection*
msData)
```

or, if data access is performed inside the tool,

```
TrackCollection* MTMD::find(TrackCollection* idTracks)
```

Interface for Muon Tagging with Calorimeter Data

```
TrackCollection* MTCD::find(TrackCollection* idTracks)
```

Note: data access is performed inside the tool,

Points to be discussed

Summarizing:

- use of input TrackParticles should be dropped if not strictly needed
- output TrackParticle and Muon creation should be handled with common tools
- output muon tracks should contain all the info needed for refitting them
- define the minimal set of output objects
- saving additional objects to StoreGate should be optional (disabled for EF operation)
- agree on interfaces