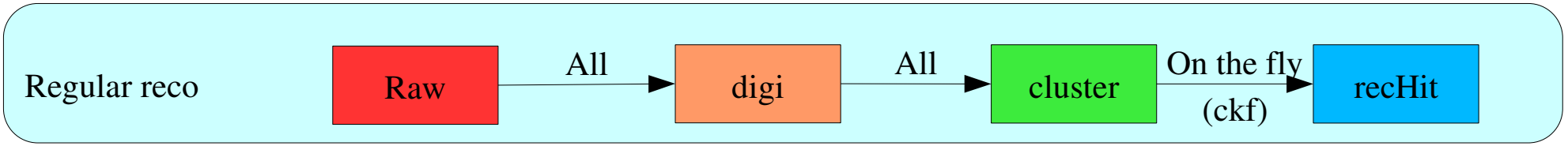
siStrip Unpacking On Demand

Restrict the number of unpacked regions of the siStrip tracker: HLT oriented tracking

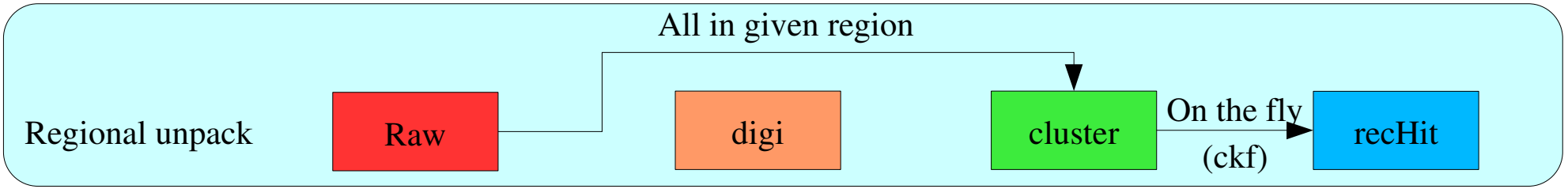
Jean-Roch Vlimant

siStrip Unpack On-Demand

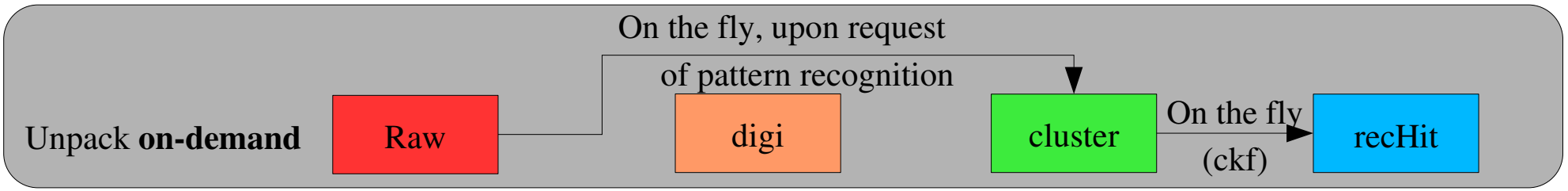
- Silicon strip unpacking is known to take a long time



- Regional unpacking has been implemented (B. Mangano and all) in 131_HLT and 16X cycles
  - Full tracker
  - Regions around ecal super cluster



- To go even further, we implemented the siStrip "unpacking on-demand"
  - Only unpack/clusters modules **requested during** pattern recognition



Two possible implementations

- The one presented on August, 22, 2007 (slides 4 to 10), slightly optimized
  - Does absolutely everything on the fly
  - Drawbacks and show-stoppers
    - The *DetMap* has to be duplicated in *DetODMap*
      (might be avoided with more in depth modification of *xs*)
    - *SiStripCluster*s have to be copied over in a vector hold by the *MeasurementTracker*,
      to get *begin* and *end* iterator
    - *TrackerRecHit2D*s have to be created with a **pointer** to the SiStripCluster
      pointer cannot be make persistent.

- A new one, presented here (slides 11 to ...)
  - First a module interface the unpacking
    - Let the MeasurementTracker defines a region to unpack for all modules of the tracker
    - Produces a RefGetter with those regions: put into the event
  - During pattern recognition
    - MeasurementTracker is updated: retrieve a handle to the RefGetter
    - MeasurementTracker::idToDet(id) is called: actually unpacking for the
      MeasrurementDet (if not already done)
  - Major drawback
    - A “region” has to be defined for **ALL the module** of the tracker: useless

First Implementation

- Pros
  - Minimal number of region defined
- Cons
  - Clusters are not in the event, and cannot be referenced

Concrete Implementation (1)

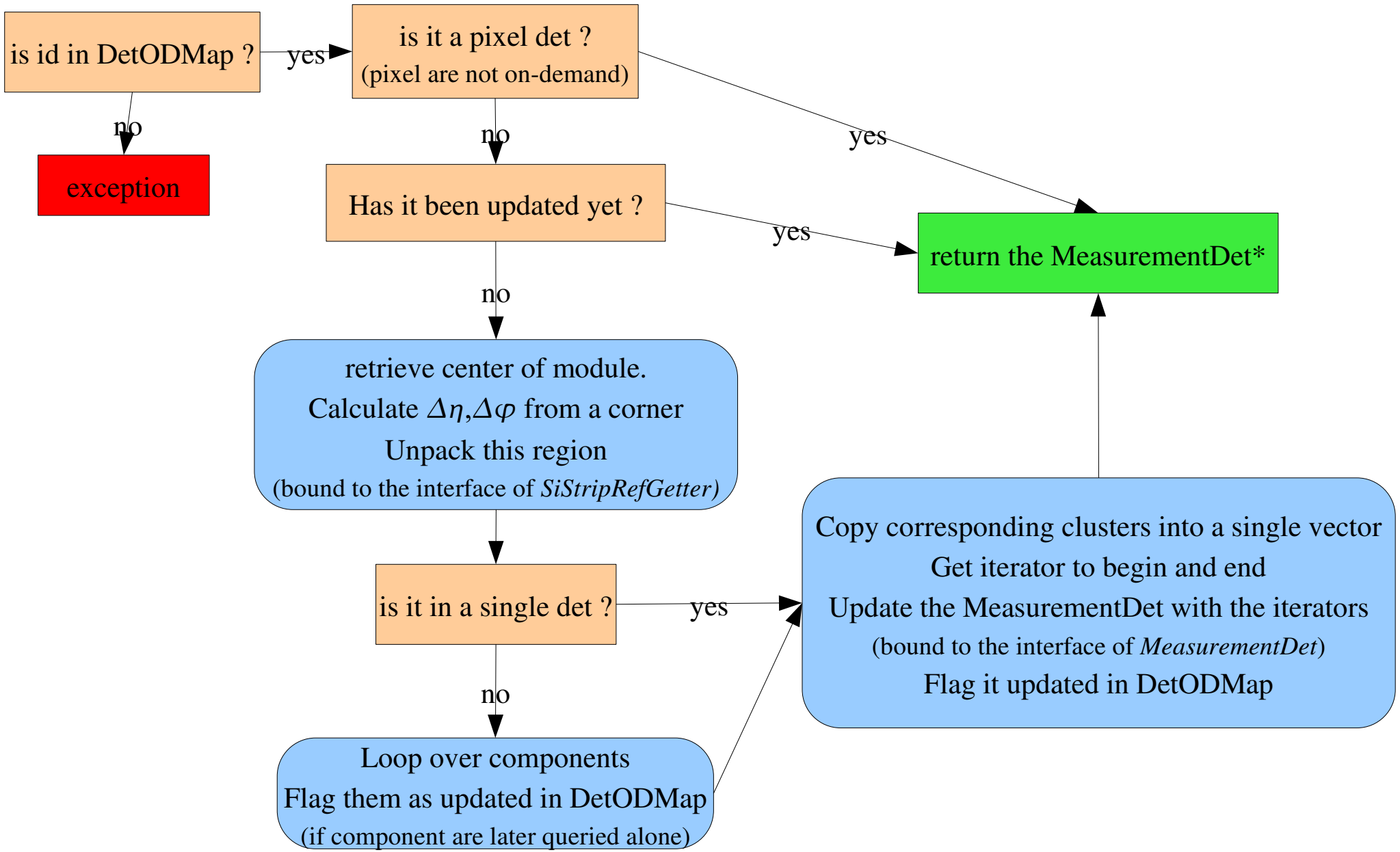
- The code I have written has the minimal effect on existing code
- Some more in depth modification may make it faster/smarter

- New class: *MeasurementTrackerOnDemand*
  - Inherits from *MeasurementTracker*: **minimal changes to existing code**
  - Overloaded function:
    - *update(const edm::Event&)*
      - Does *updatePixels(const edm::Event&)* for the pixel "a la" MeasurementTracker: nothing's new here
      - Gets the *SiStripLazyGetter<SiStripCluster>* from the event
      - Reserve 10000 regions in a *SiStripRefGetter<SiStripCluster>* for further use

Concrete Implementation (2)

- New class: *MeasurementTrackerOnDemand*
  - Inherits from *MeasurementTracker*: **minimal changes to existing code**
  - Overloaded function:
    - *idToDet(DetId & id)*
      - Look up for the id “a la” MeasurementTracker: to check the id is valid (DetMap)
        - If pixel: return the *MeasurementDet**
        - If strip det
          - Has it been updated yet ?
            - If yes: return the *MeasurementDet**
            - If not:
              - Unpack around the module
              - Update the *MeasurementDet**
                - Separate components if glued
                - Make an entry in the map (DetODMap) to avoid double unpacking

Overloading of *idToDet(DetId & id)*



Validation/test (1)

- Nicholas Cripps is looking into fully validating the track reconstruction
- Regular track reconstruction cannot be broken ... (hopefully)
- "Regional" clustering has been run and still works: not extensively tested

- Extensive tests have been for the unpack-on-demand within L3Muon reconstruction
  - http://indico.cern.ch/conferenceDisplay.py?confId=19406
  - comparison between regular cluster reconstruction and on-demand only.
  - No loss of any track (not that I can noticed over thousands of events)

Paths/modules	modes	
	regular	**on-demand**
SiStripRawToClustersFacility	N/A	5 ms
siPixelClusters	< 1 ms	< 1 ms
siPixelRecHits	< 1 ms	< 1 ms
siStripClusters	310 ms	N/A
Total:	310 ms	5 ms

Single muon gun: 5-200 GeV

	INside-OUT (5-7 hits)			
Paths/modules	ckfPXL		ckfRSpxl (new)	
Description	From pixel pairs		bare seed on pxl surf.	
	regular	**on-demand**	**regular**	**on-demand**
Overhead	310 ms	5 ms	310 ms	5 ms
L3Muons	100 ms	115 ms	37 ms	40 ms
Total:	410 ms	120 ms	347 ms	45 ms

- ~300 ms overhead dropped
- Algorithm timing increases by 10-25%

Validation/test (2)

- Now comparing (more fair) the timing of
  - "Regional" unpacking (**all** raw to cluster)
  - Unpacking on demand (**only requested** raw to cluster)

Defines all the region to the SiStripLazyGettter

EDAnalyzer that updates the MeasurementTracker

Paths/modules	modes	
	"regional"	**on-demand**
SiStripRawToClustersFacility	5 ms	
SiStripRoI	10 ms	N/A
siPixelClusters	2 ms	
siPixelRecHits	< 1ms	
measurementTrackerUpdator	180 ms	6 ms
L3 muon algorithm	X ms	X*1.1 ms

Increase of 10-25% of the algorithm timing when on-demand

- On-demand will be faster for L3 muon path up to X ~ 1 s

Additionnal tools

- MeasurementTrackerUpdator:
  - EDAnalyzer that calls *MeasurementTracker::update(const Event&)*
  - Avoid to count the update in timed module

- MeasurementTrackerClusters:
  - EDProducer that writes to the event the clusters unpacked/created by the MeasurementTracker
  - Keep a record of the unpacked/created clusters

Second Implementation

- Pros
  - Cluster are in the event and can be Referenced
- Cons
  - Way too many region defined: One per module. Large *RefGetter*

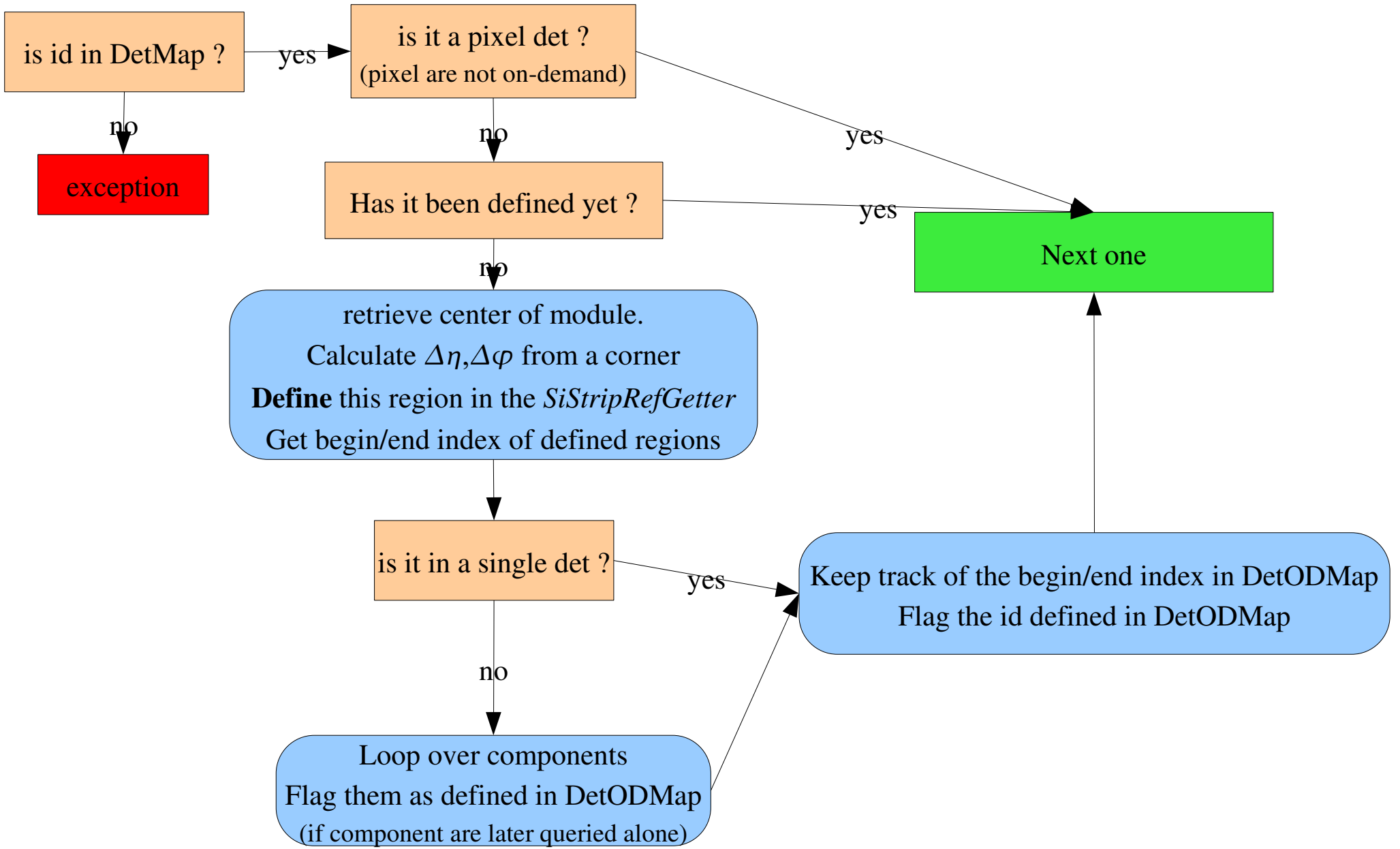
Concrete Implementation (1)

- The code I have written has the minimal effect on existing code

- New class: *MeasurementTrackerOD*
  - Inherits from *MeasurementTracker*: **minimal changes to existing code**
  - New function:
    - *define(const edm::Event&,*
      Handle<SiStripLazyGetter<SiStripCluster>>,
      auto_ptr<SiStripRefGetter<SiStripCluster>>)
      - Loop over available *MeasurementDet*s
        - Does nothing for pixels
        - Update the getter with regions around strip modules (splitting the glued)

- Module *MeasurementTrackerUpdator*
  - Produces a *SiStripRefGetter<SiStripCluster>*
  - Retrieve the *LazyGetter* from the event
  - Retrieve the *MeasurementTrackerOD* and calls *define(event,LazyGetter,RefGetter)*
  - Put the *RefGetter* to the event

Definition of *define(...)*



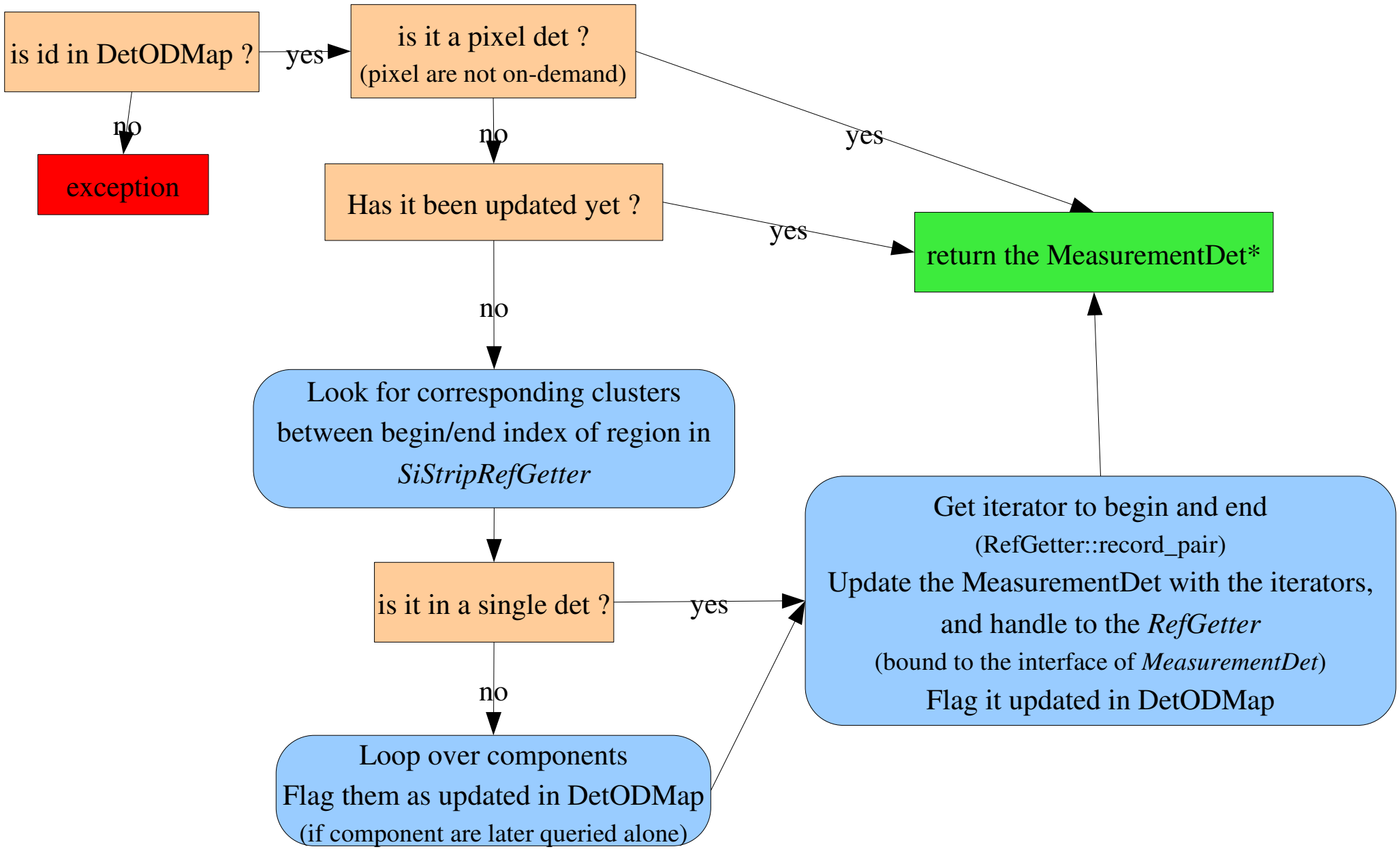
Concrete Implementation (2)

- New class: *MeasurementTrackerOD*
  - Inherits from *MeasurementTracker*:
  - Overloaded function:
    - *update(const edm::Event&)*
      - Does *updatePixels(const edm::Event&)* for the pixel “a la” MeasurementTracker: nothing's new here
      - Gets the *SiStripRefGetter<SiStripCluster>* from the event

Concrete Implementation (3)

- New class: *MeasurementTrackerOnDemand*
  - Inherits from *MeasurementTracker*:
  - Overloaded function:
    - *idToDet(DetId & id)*
      - Look up for the id “a la” MeasurementTracker: to check the id is valid
        - If pixel: return the *MeasurementDet**
        - If strip det
          - Has it been updated yet ?
            - If yes: return the *MeasurementDet**
            - If not:
              - Retrieve cluster reference from the *RefGetter*
                - Update the *MeasurementDet**
                  - Separate components if glued
                  - Make an entry in the map (DetODMap) to avoid double unpacking

Overloading of *idToDet(DetId & id)*



Second Implementation (bis)

- Pros
  - Less region defined
- Cons
  - Still more regions defined than really needed

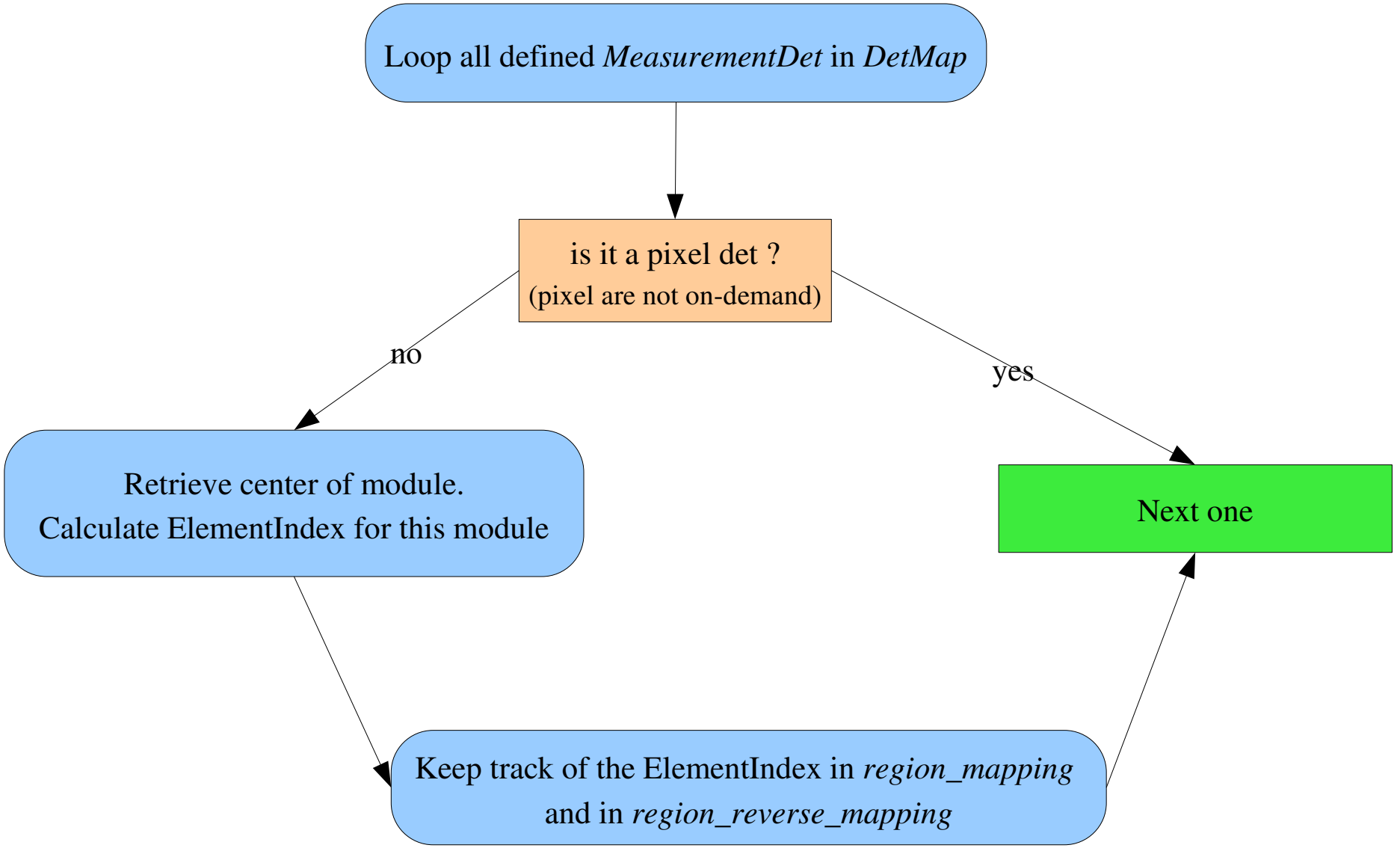
Concrete Implementation (1)

- The code I have written has the minimal effect on existing code

- New class: *MeasurementTrackerOD*
  - Inherits from *MeasurementTracker*: **minimal changes to existing code**
  - New function:
    - *constructor(...)*
      - Loop over available *MeasurementDets*
        - Does nothing for pixels
        - Find out the ElementIndex for this strip modules
    - *define(const edm::Event&,*
      Handle<SiStripLazyGetter<SiStripCluster>>,
      auto_ptr<SiStripRefGetter<SiStripCluster>>)
      - Update the getter with the list of elementIndex
      - Register region index in a map

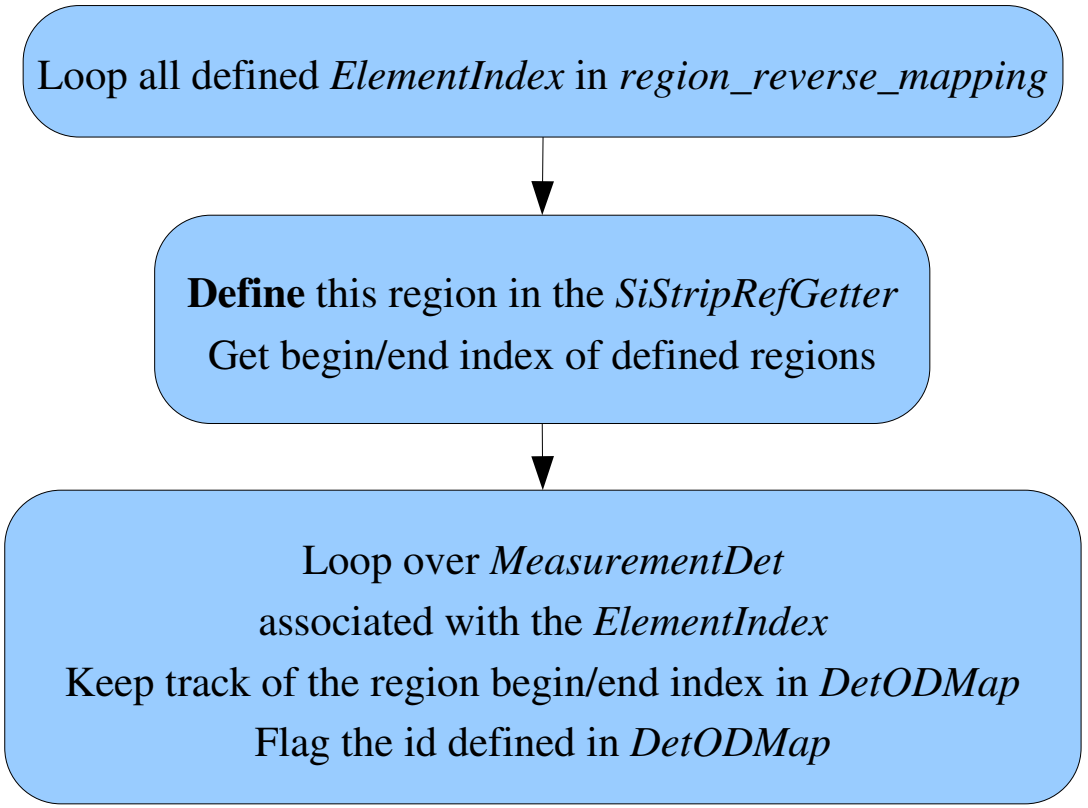
- Module *MeasurementTrackerUpdator*
  - Produces a *SiStripRefGetter<SiStripCluster>*
  - Retrieve the *LazyGetter* from the event
  - Retrieve the *MeasurementTrackerOD* and calls *define(event,LazyGetter,RefGetter)*
  - Put the *RefGetter* to the event

Overloading of *constructor(...)*



Definition of *define(...)*

- Called by an external module that put the *RefGetter* in the event



Concrete Implementation (2)

- New class: *MeasurementTrackerOD*
  - Inherits from *MeasurementTracker*:
  - Overloaded function:
    - *update(const edm::Event&)*
      - Does *updatePixels(const edm::Event&)* for the pixel “a la” MeasurementTracker: nothing's new here
      - Gets the *SiStripRefGetter<SiStripCluster>* from the event

Concrete Implementation (3)

- New class: *MeasurementTrackerOnDemand*
  - Inherits from *MeasurementTracker*:
  - Overloaded function:
    - *idToDet(DetId & id)*
      - Look up for the id “a la” MeasurementTracker: to check the id is valid
        - If pixel: return the *MeasurementDet**
        - If strip det
          - Has it been updated yet ?
            - If yes: return the *MeasurementDet**
            - If not:
              - Retrieve cluster reference from the *RefGetter*
                - Update the *MeasurementDet**
                  - Separate components if glued
                  - Make an entry in the map (DetODMap) to avoid double operation

Overloading of *idToDet(DetId & id)*