CARF

an Analysis&Reconstruction Framework for CMS

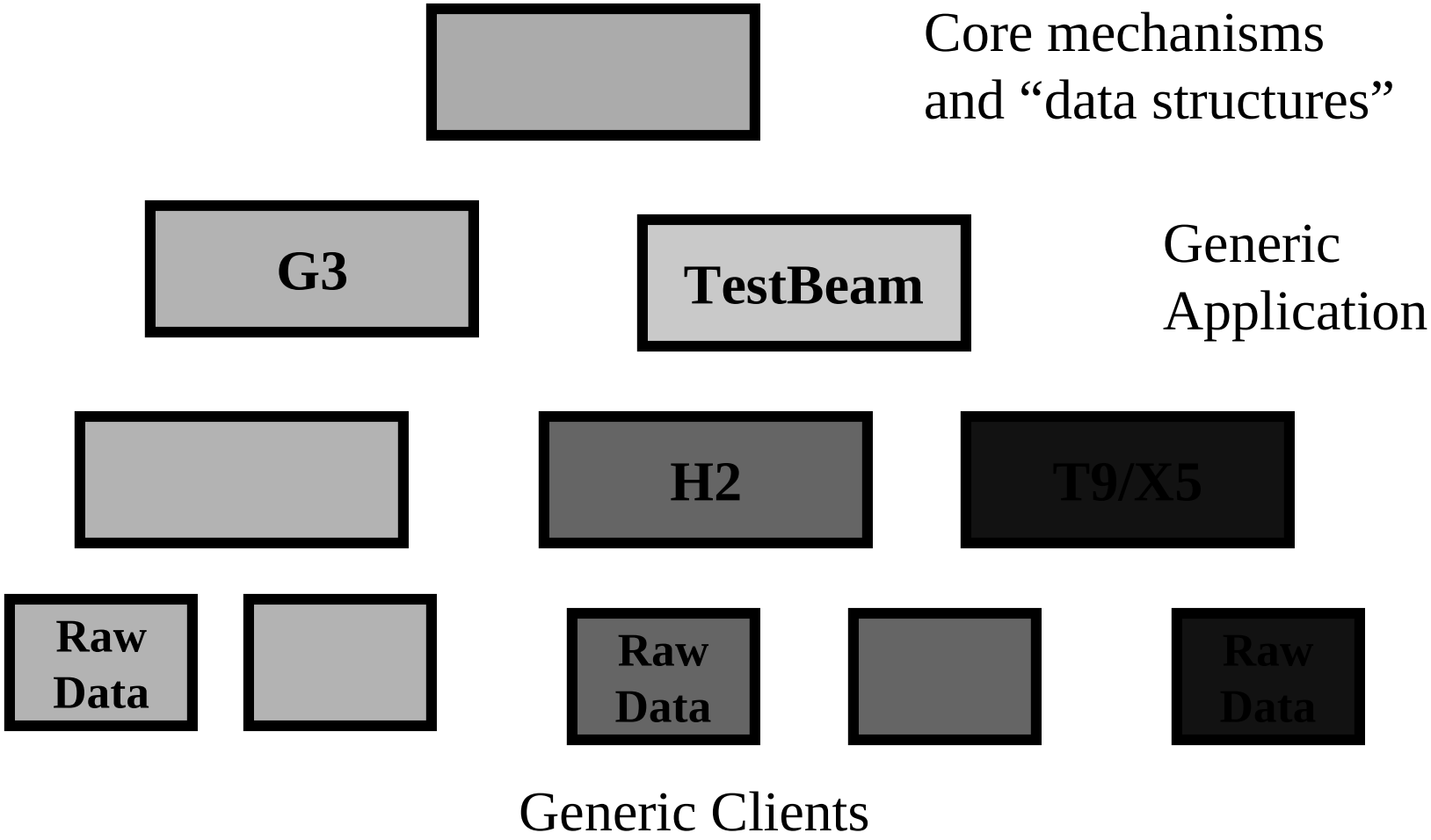
Vincenzo Innocente

CERN/EP/CMC

CARF Development Philosophy

- Tailored to CMS analysis and reconstruction
- Serves present ORCA applications
  - priorities driven development
  - backward compatibility, legacy code and data!
- Bottom-up (concrete→abstract) development in synergy with other sub-systems
- It is also a prototype of 2005 software
  - offers more than one solution to a single problem

CARF layered Structure



Framework Basic Dynamics

- No central ordering of actions, no explicit control of data flow: only implicit dependencies
- External dependencies managed through an Event Driven Notification to "subscribers"
- Internal dependencies through an Action on Demand mechanism

Framework Main Services

- Define the events to be dispatched and links them to the their actual source
- Allow the selection among available resources (user plug-in's)
- Manage the "not yet removed" sequential components

Framework Ancillary Services

- User Interface
- Error Report (Exception management)
- Logging facilities
- Timing facility (statistics gathering)
- Utility library
  - Notably Objy utilities, wrappers and generic persistent capable classes

Framework middle layer

- A CARF Application is characterized by the events it dispatches
- Implementation of generic clients to specific services (events)
  - simplified API
  - uniform detailed design
  - uniform use of ancillary services
- Requires synergy with detectors' sub-systems

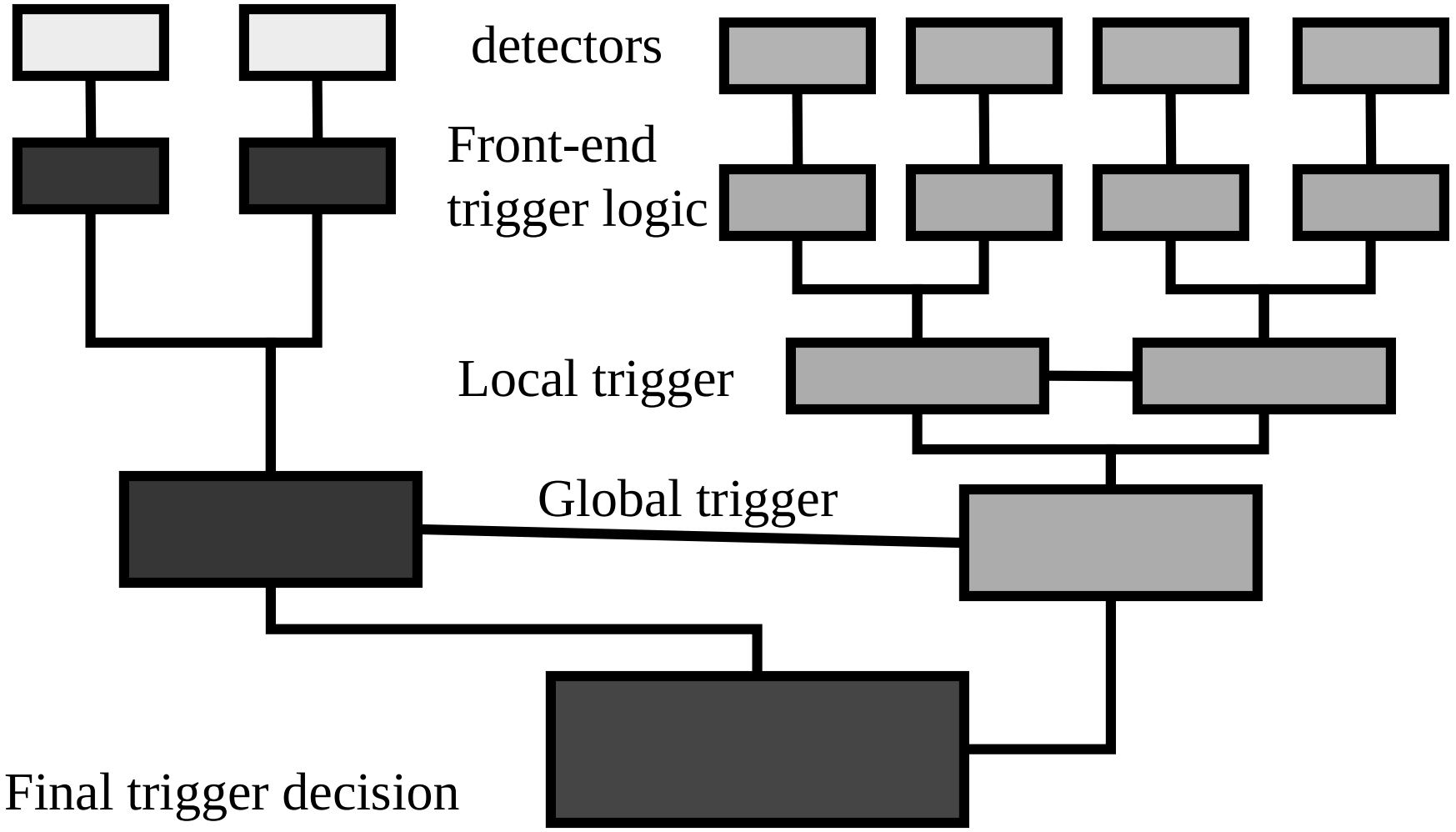
Use Cases

L1 Trigger Simulation

Track Reconstruction

“Physics” reconstruction

L1 Trigger Simulation

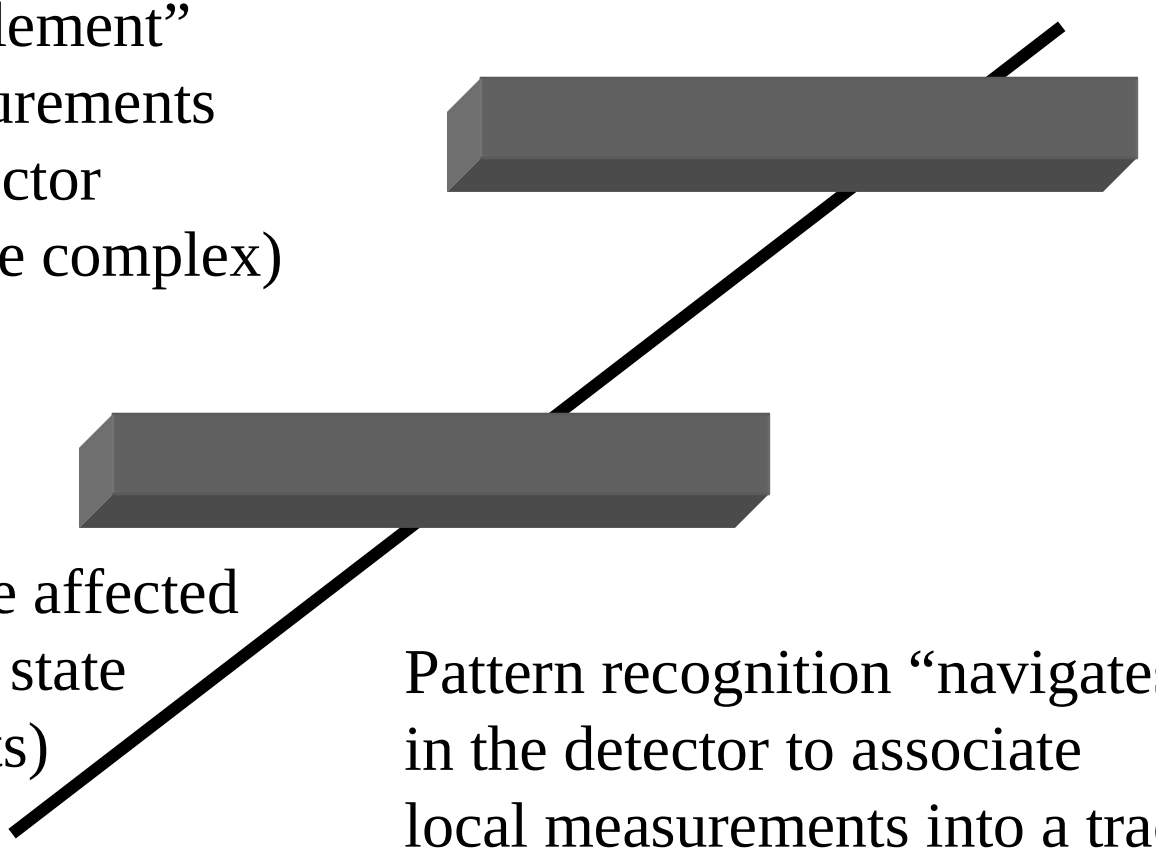


L1 Trigger Simulation

- Accurate simulation of real electronics
- In the real experiment only “final decision data” are propagated forward
- Also required
  - Monitoring of single trigger units
  - Comparison of L1 trigger w.r.t. full reconstruction
  - ability to simulate just a part of the system
  - “save” computing intensive intermediate results

Track Reconstruction

For each “detector element”
there are local measurements
of trajectory state-vector
(just position or more complex)



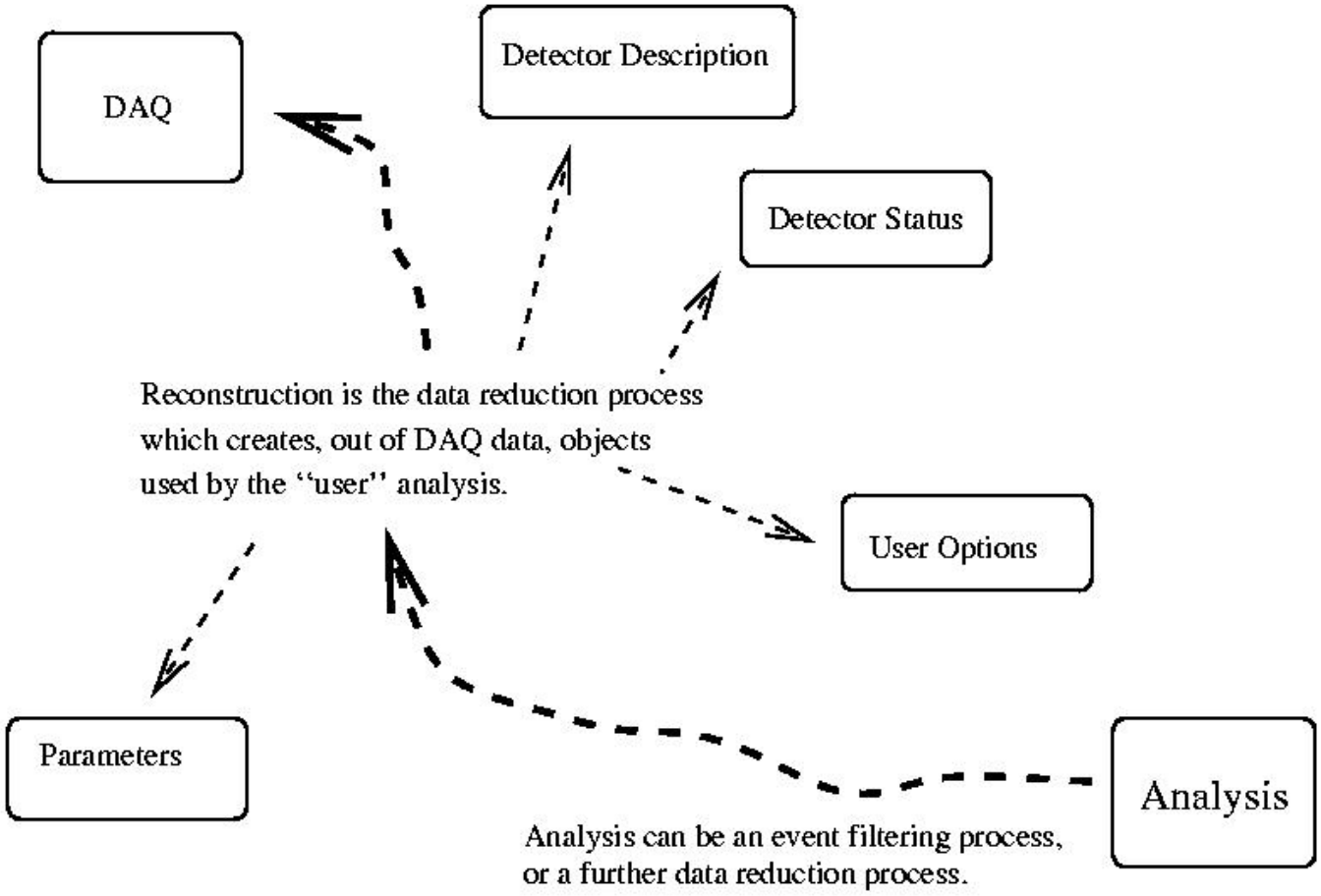
Local measurements are affected
by the detector element state
(calibrations, alignments)

Pattern recognition “navigates”
in the detector to associate
local measurements into a track

“Physics” reconstruction

- 4-vector-like objects are built out of trajectories and localized energy deposits
- A wide range of PID, jet, vertex etc algorithms can be applied to produce others 4-vector-like objects
- Access to the “original” detector data maybe required

Reconstruction Sources



Reconstruction Scenario

- Reproduce "Detector Status" at the moment of the interaction:
  - front-end electronics signals (digis)
  - calibrations
  - alignments
- Perform local reconstruction as a continuation of the front-end data reduction until objects "detachable" from the detectors are not obtained
- Use these objects to perform physics reconstruction and analysis

Components

- Reconstruction Algorithms
- Event Objects
- Other services (detector objects, parameters, etc)
- Legacy not-OO data (GEANT3)

CARF Fundamentals
(inside the black box)

Detector Components

Event Driven Notification

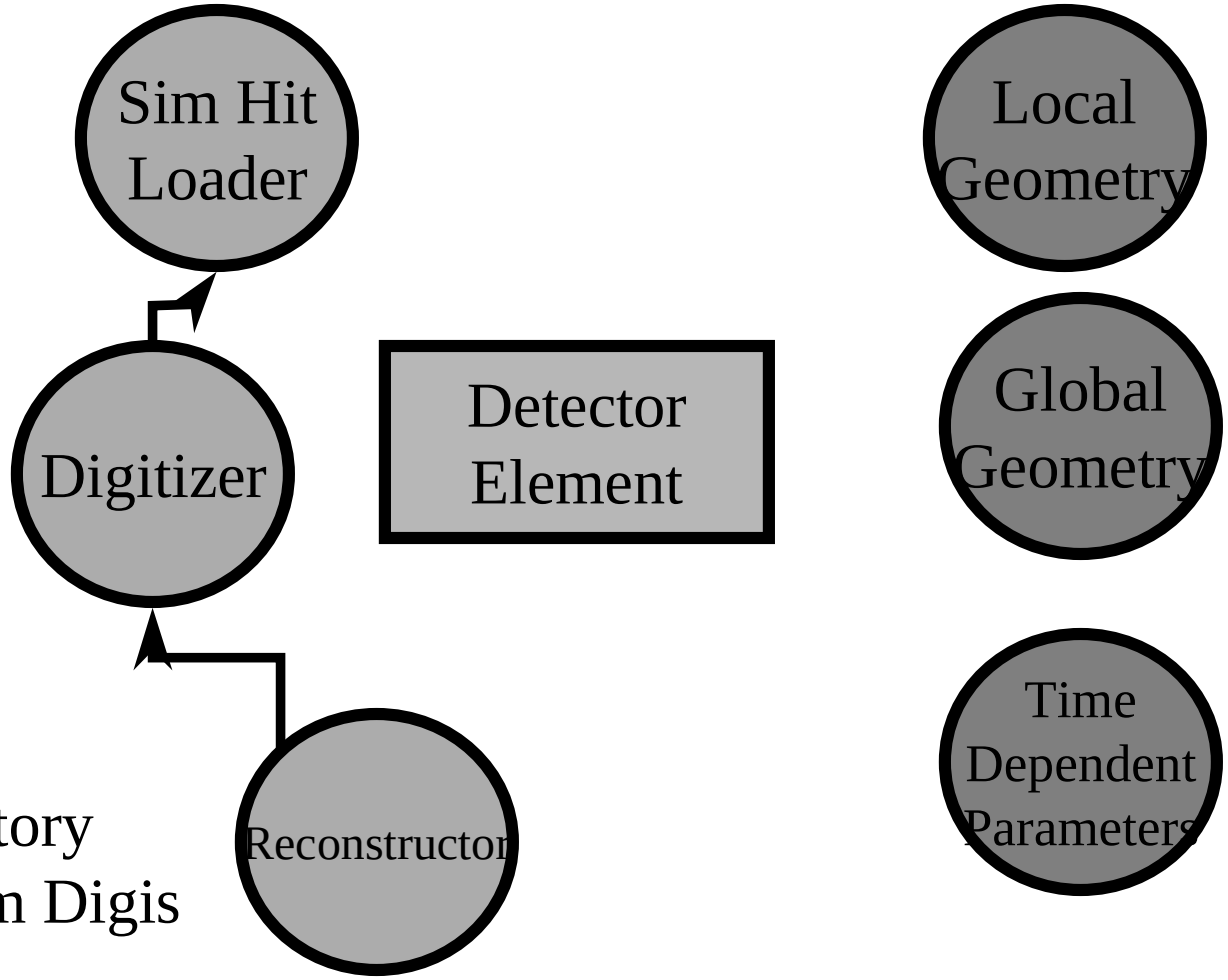
Action on Demand

Detector Components

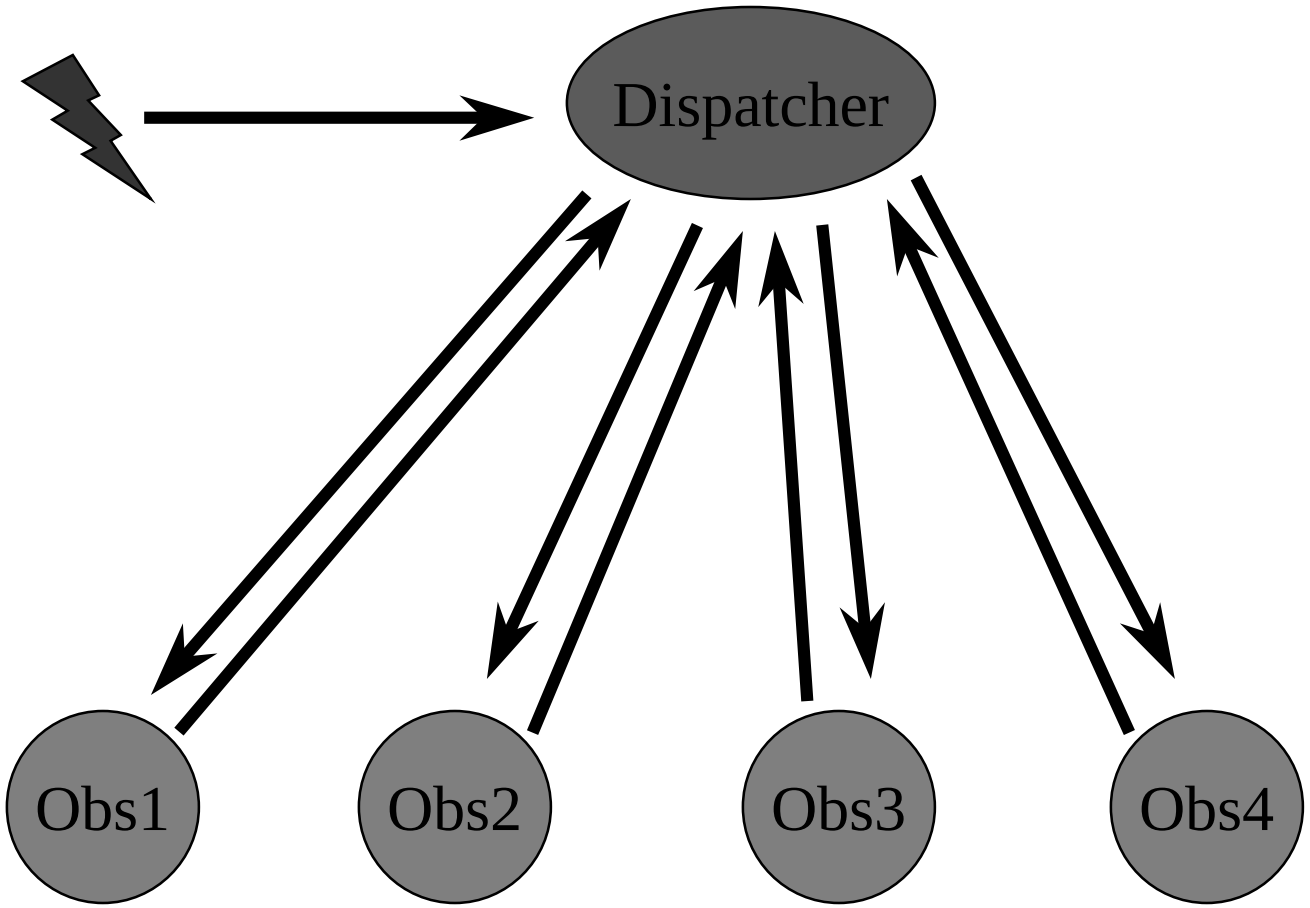
Load simulated hits from MC

Generates Digis from SimHits (or loads them from db)

Reconstruct measured trajectory state-vector from Digis

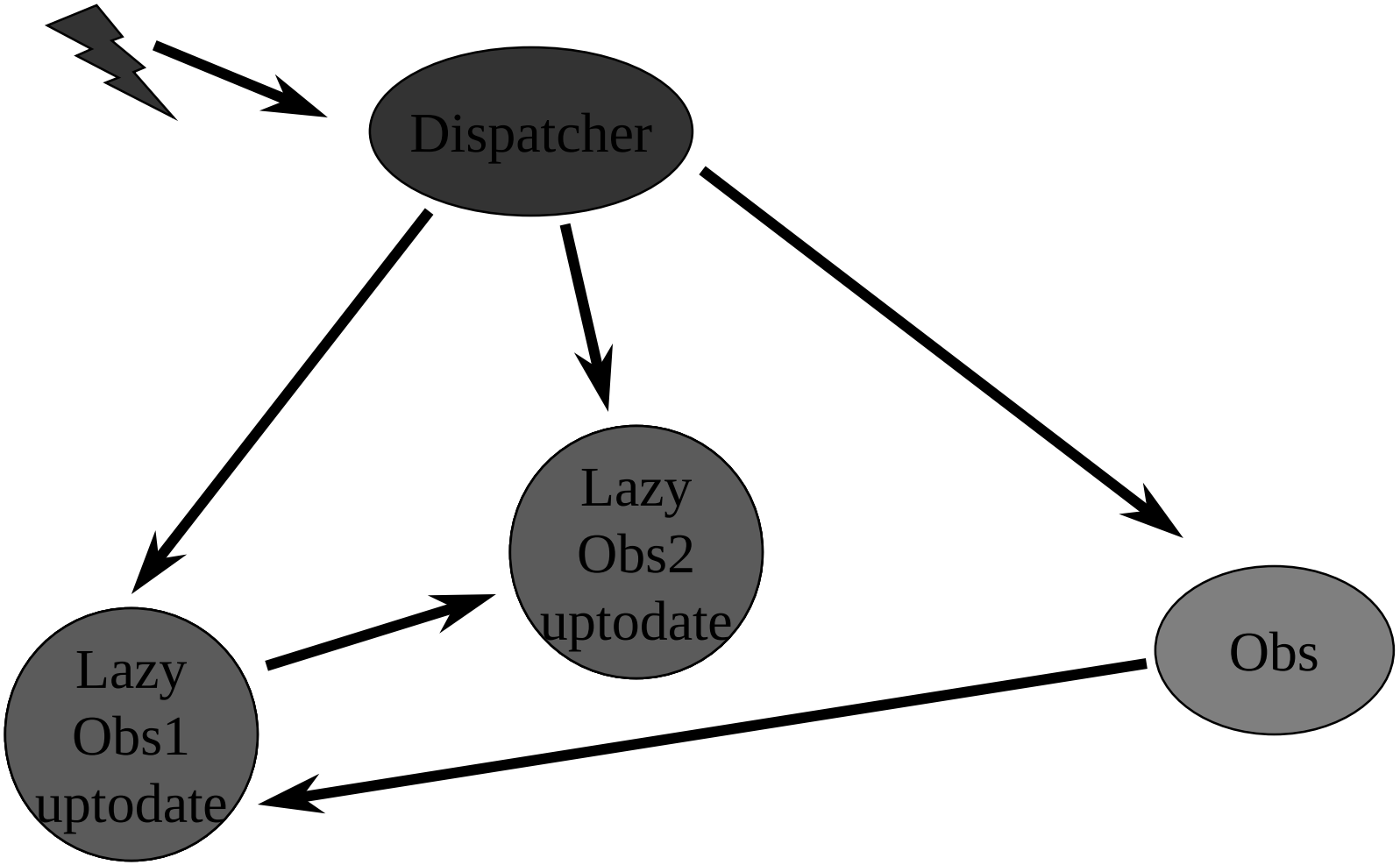


Event Driven Notification

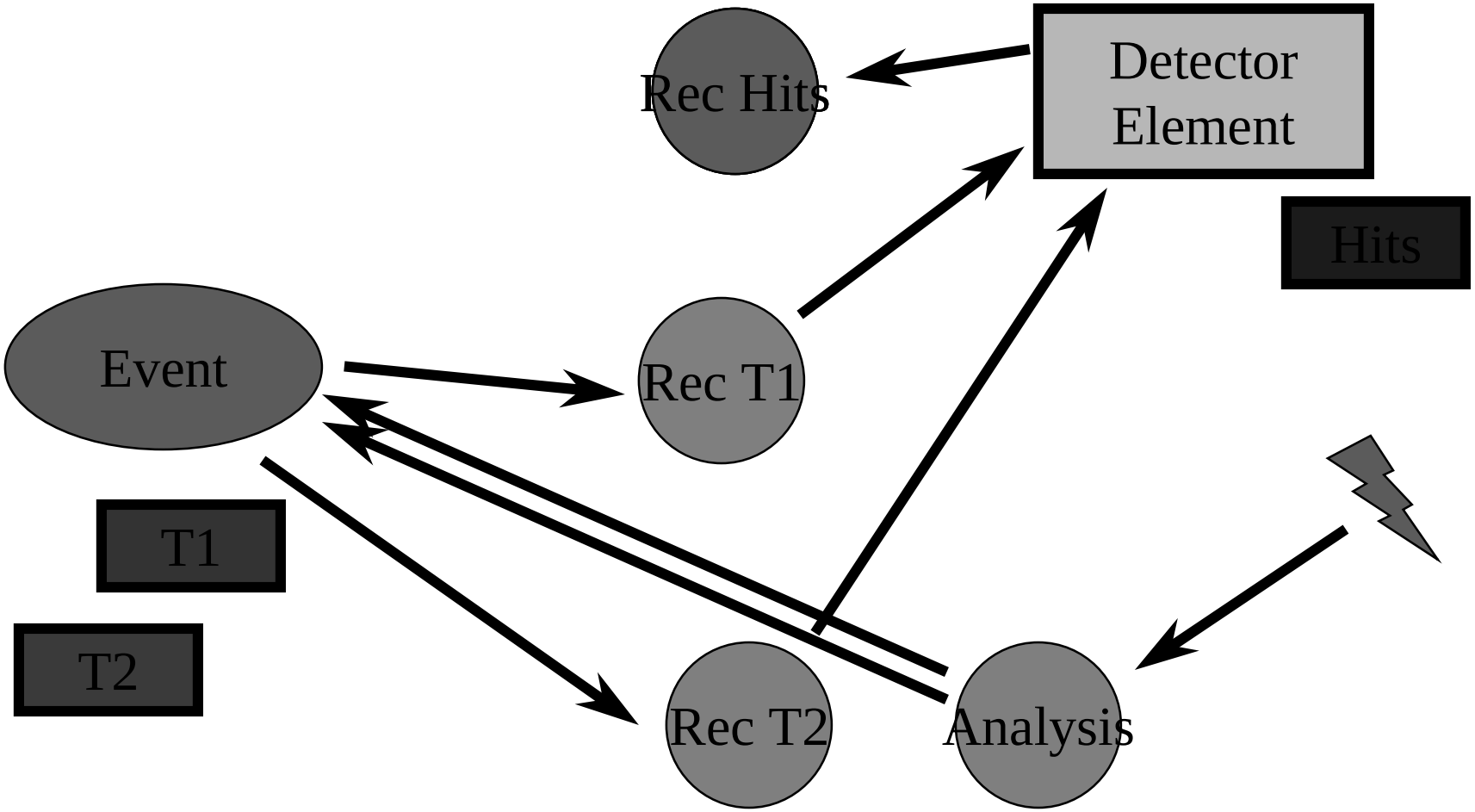


Observers

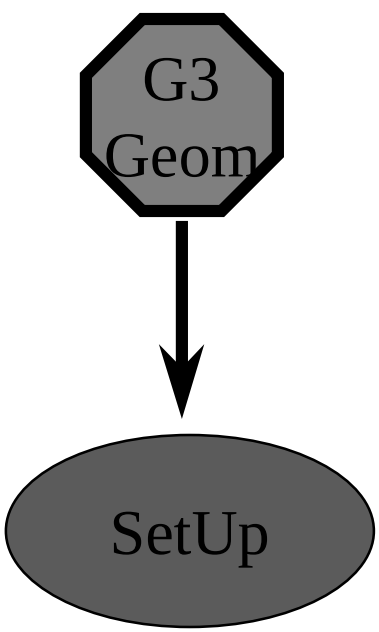
Active and Lazy Observers



Action “on Demand”



“Events” currently dispatched



Start Xing

Ready to build new
G3 simulated event

SimPileUp

New pile-up event
ready in Zebra memory

SimTrigger

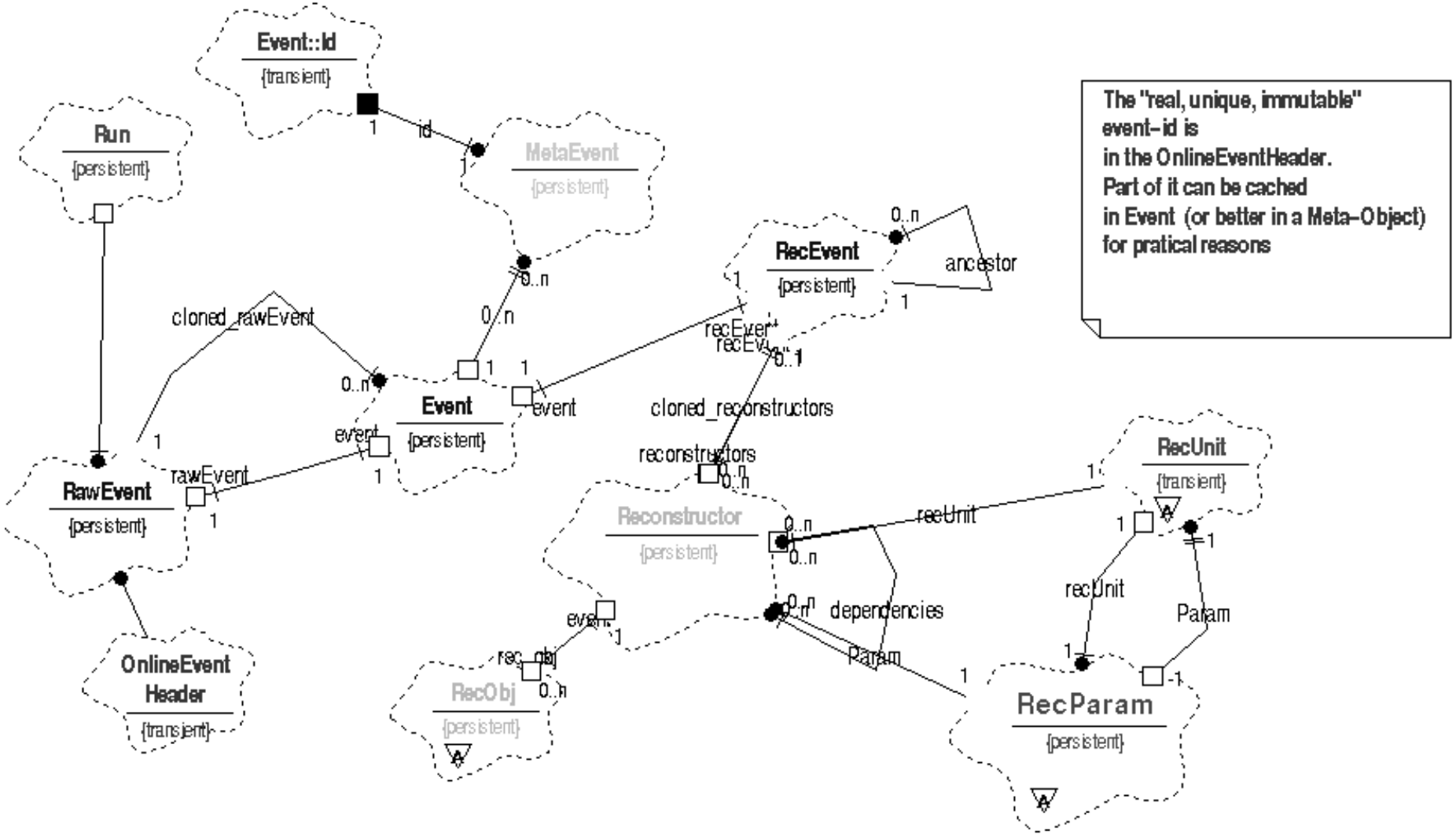
New trigger event
ready in Zebra memory

SetUp Observers are Objects
which depend on geometry

G3Event

New event “ready”
to be analyzed

“RecObj” Object Model



Object identification

- A Detector object collection is identified by:
  - object type (or super-type)
  - detector element it belongs to
- implicitly belongs to the "current crossing"
- Required the detector to be "in place" and "operational"

Object identification

- A RecObj collection is identified by:
  - object type (or super-type)
  - name of the Reconstructor (same as RecUnit)
  - event it belongs to
- Does not require the RecUnit to be in place or operational