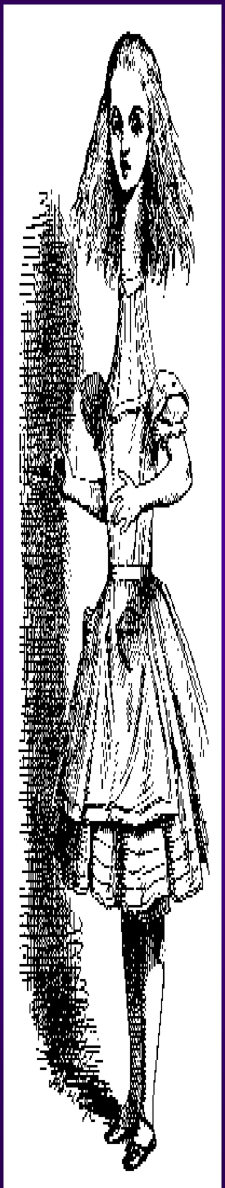
The ALICE Simulation Strategy

Andreas Morsch
For the ALICE Offline Group

Joint STAR/ALICE Offline Meeting
Brookhaven National Laboratory, Upton, NY
April 8-9, 2000

Outline

- General Strategy
- MC Interface to Transport Codes
- Generators
- Designed for Evolution
- Link to reconstruction and visualization
- Fast Simulation



General Strategy

- **Clear distinction between immediate and long-term requirements.**
- **Assure coherence of the whole simulation process:**
  - **Event generation**
  - **Particle Tracking**
  - **Signal Generation**
  - **Digitisation**
  - **Fast simulation**
- **Maximum (re)use of existing code and knowledge (people):**
  - **Geant3 based simulation code**
  - **Users come with FORTRAN+PAW+CERNLIB background**

Immediate Requirements

- Simulations needed for
  - Technical Design Reports
  - Detector design optimization
  - Proof of principle for new physics analysis ideas
  - Physics Performance Report
- Profit from OO design as early as possible
- Design for change with maximum code reuse, i.e. integration of new
  - Detector Components
  - Generators
  - Detector response strategies
- and
  - Fast simulation

Long Term Goals

- **Comparison between Geant3 and Geant4 using the same geometry and data structure is mandatory (QA)**
- **Smooth transition to Geant4 with maximum reuse of Geant3 based simulation (user-) code**
- **Possible integration of other tracking codes (fast simulators, FLUKA, ...)**

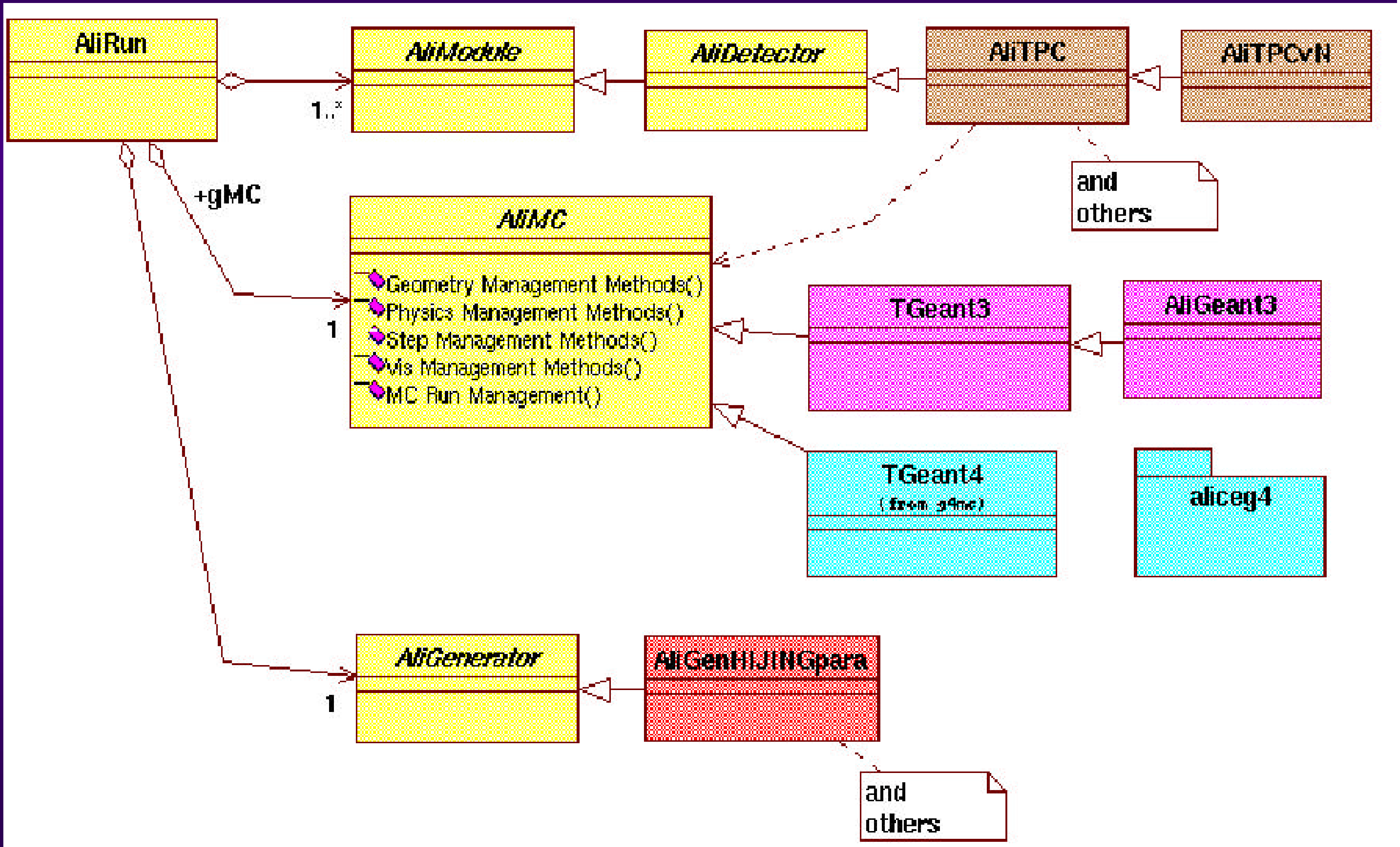
➔Use MC interface class to hide implementation specific features

➔ Define G3 and G4 geometries from the same code.

Simulation in the AliRoot Framework

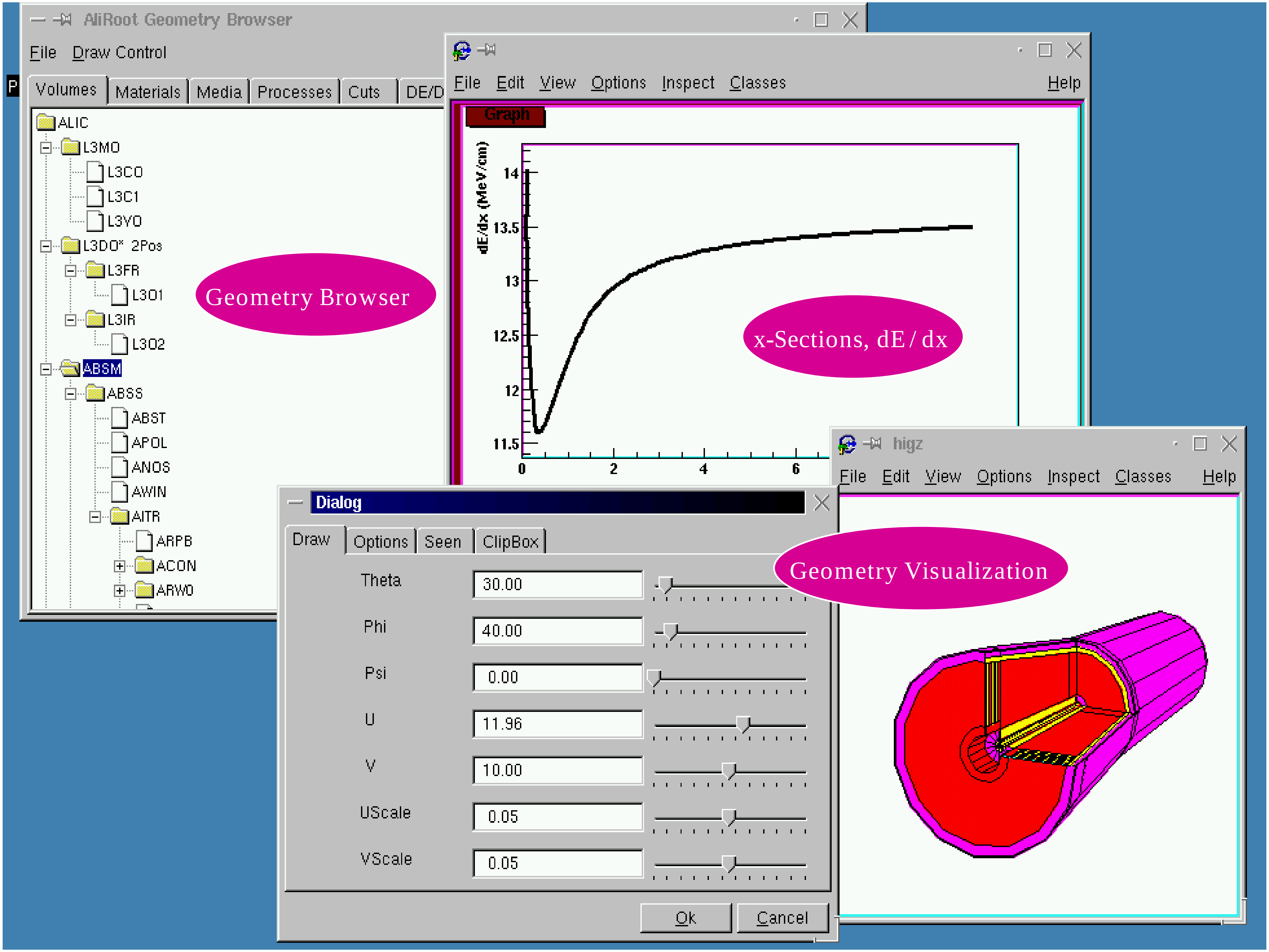
- **AliRun as central run manager**
- **Interface classes to provide modularity, flexibility and coherence of the simulation process**
  - **Monte Carlo interface: AliMC and AliVMC**
  - **Generator Interface: AliGenerator**
  - **Detector classes mapping the detector logical structure and granularity: AliModule, AliDetector**
- **In the future: Run manager base class to incorporate full simulation and different levels of fast simulation**
- **Visualization, browsers and GUI essential for debugging and ease of use**

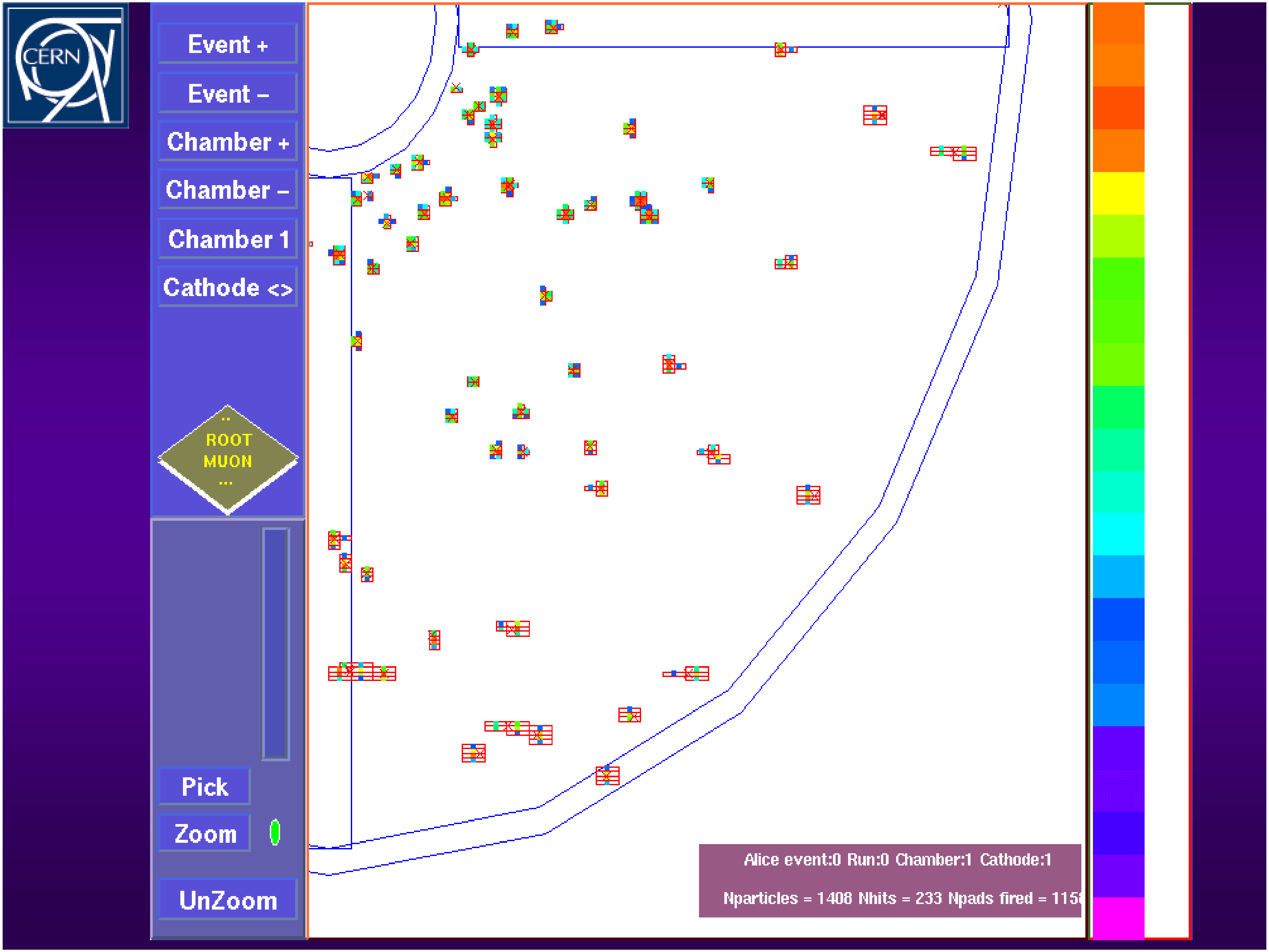
The AliRun Manager



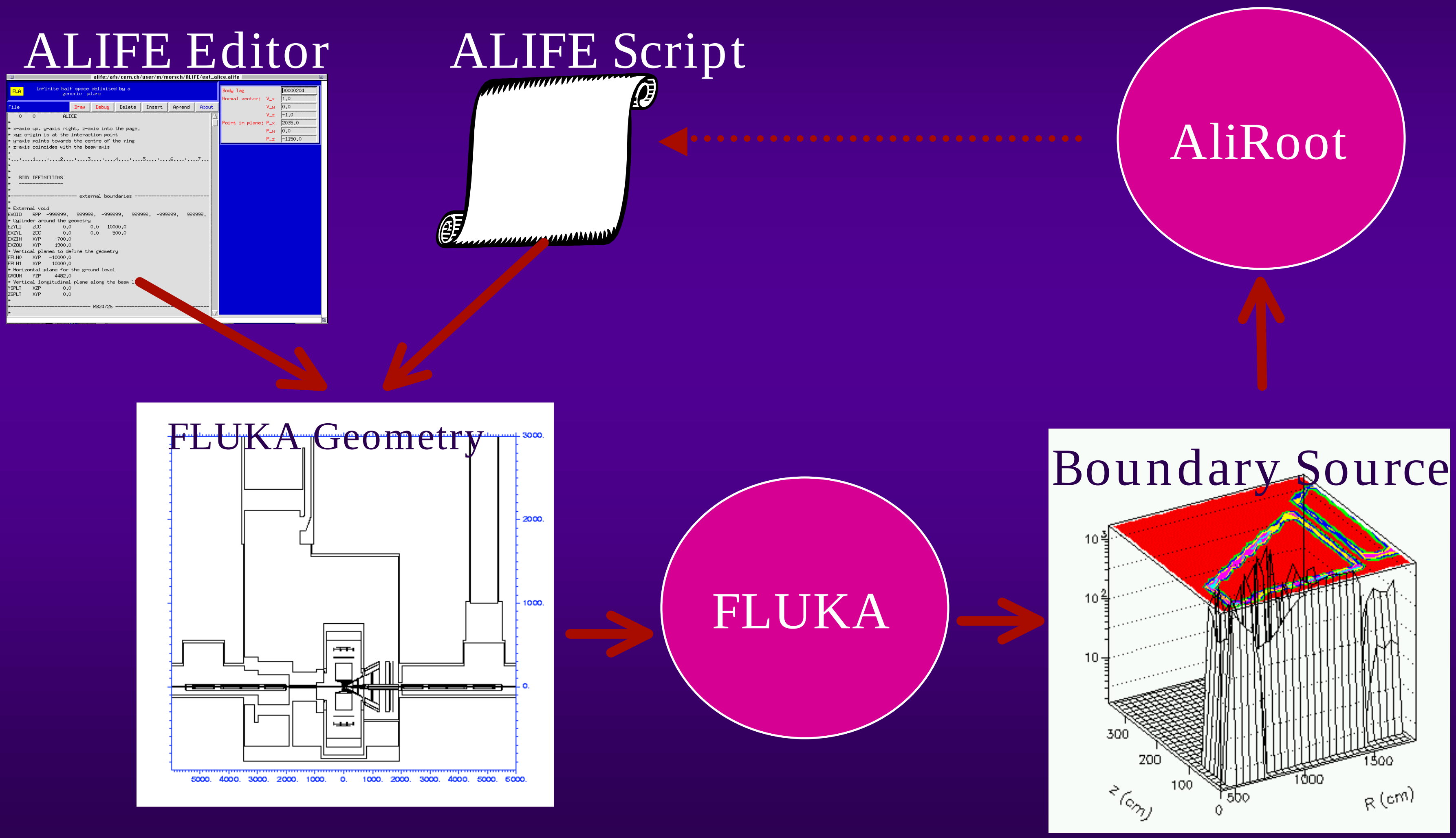
Virtual Monte Carlo Interface

- Both G3 and G4 simulations are based on the virtual Monte Carlo interface.
  - Both G3/G4 applications are defined from the same source
  - Same input, geometry, hits, control
- Implementation of AliMC for G3 = TGeant3 class
- Implementation of AliMC for G4 = TGeant4 class
  - Each domain is covered by its manager class: geometry, physics, stepping, visualisation
  - Each manager uses corresponding categories of G4
- “T”-classes contain only ALICE independent code


AliRoot Geometry Browser
File Draw Control
Volumes Materials Media Processes Cuts DE/D
ALIC
L3MO
L3CO
L3C1
L3VO
L3DO* 2Pos
L3FR
L3O1
L3IR
L3O2
ABSM
ABSS
ABST
APOL
ANOS
AWIN
AITR
ARPB
ACON
ARWO
Geometry Browser
File Edit View Options Inspect Classes Help
Graph
dE/dx (MeV/cm)
14
13.5
13
12.5
12
11.5
0
2
4
6
x-Sections, dE / dx
Dialog
Draw Options Seen ClipBox
Theta 30.00
Phi 40.00
Psi 0.00
U 11.96
V 10.00
UScale 0.05
VScale 0.05
Ok Cancel
higz
File Edit View Options Inspect Classes Help
Geometry Visualization

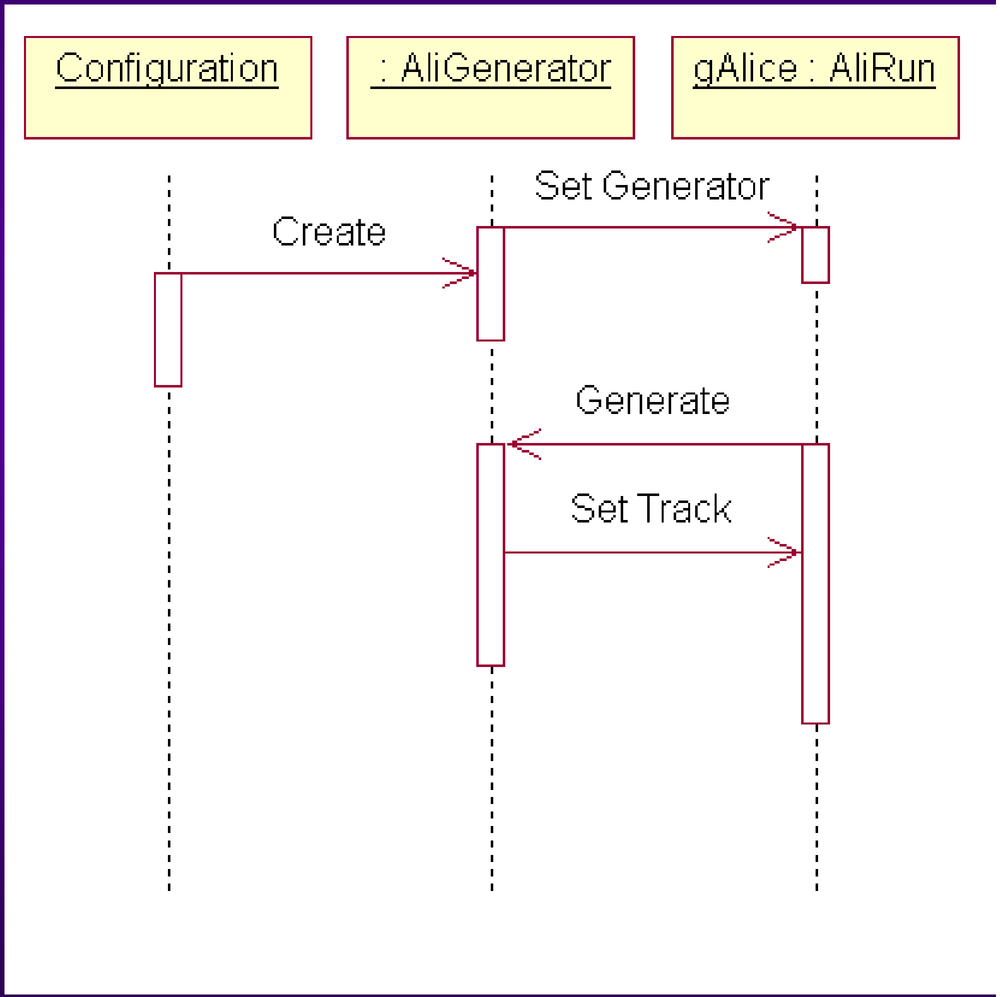

CERN
Event +
Event –
Chamber +
Chamber –
Chamber 1
Cathode <>
ROOT
MUON
Pick
Zoom
UnZoom
Alice event:0 Run:0 Chamber:1 Cathode:1
Nparticles = 1408 Nhits = 233 Npads fired = 115

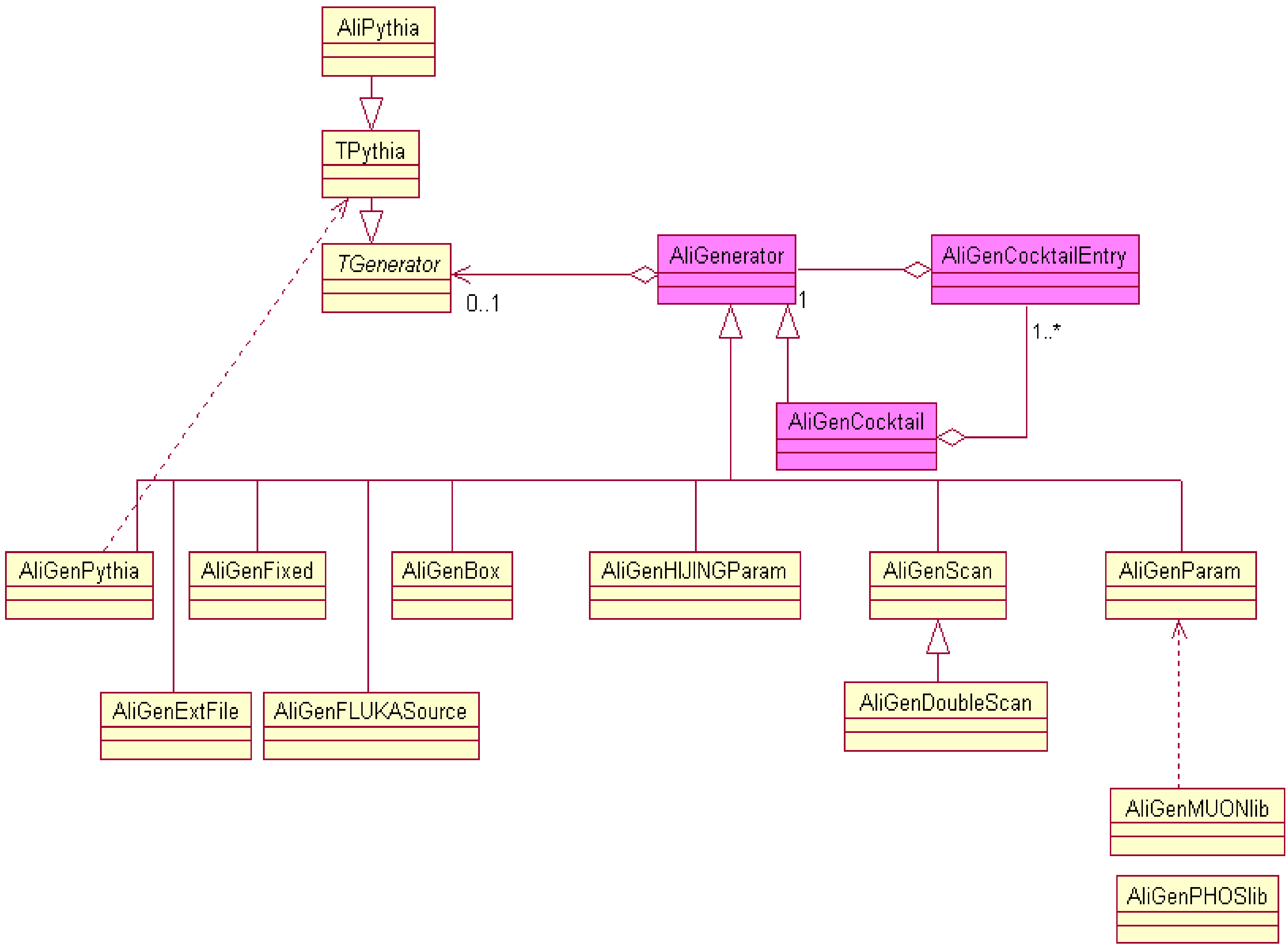
FLUKA in the ALICE Simulation Framework



Generator Interface: AliGenerator

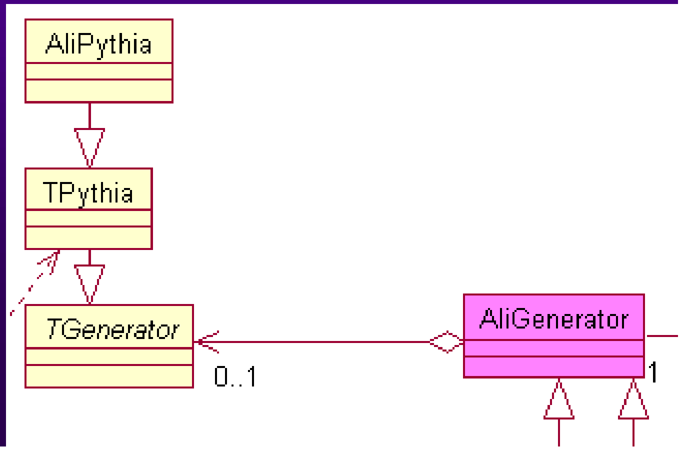
- **Provide user with**
  - **Easy and coherent way to study variety of physics signals**
  - **Testing tools**
  - **Background studies**
- **Possibility to study**
  - **Full events (event by event)**
  - **Single processes**
  - **Mixture of both ("Cocktail events")**




AliPythia
TPythia
TGenerator
AliGenerator
AliGenCocktailEntry
0..1
1
1..*
AliGenCocktail
AliGenPythia
AliGenFixed
AliGenBox
AliGenHIJINGParam
AliGenScan
AliGenParam
AliGenExtFile
AliGenFLUKASource
AliGenDoubleScan
AliGenMUONlib
AliGenPHOSlib

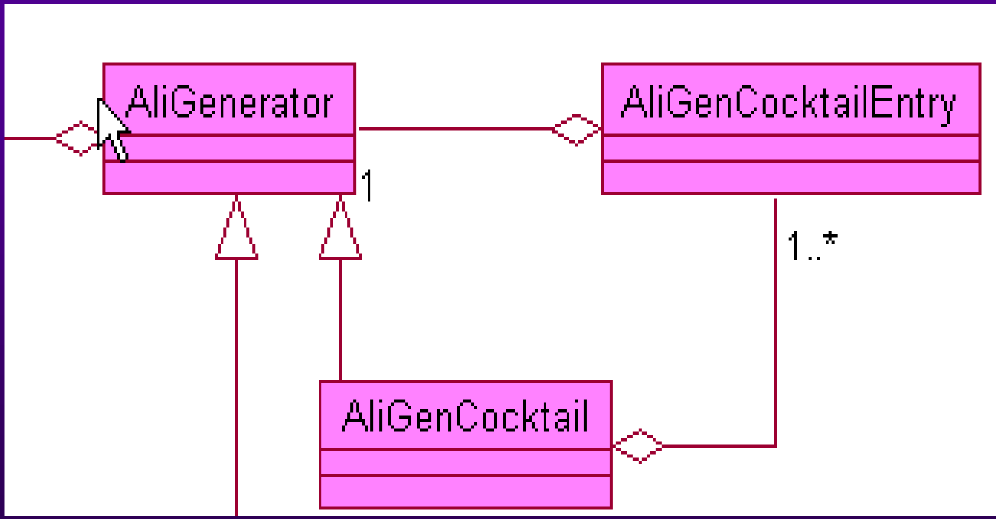
Interface to Pythia and Jetset

- TPythia derived from TGenerator
  - Access to Pythia and Jetset common blocks via class methods
  - implements TGenerator methods
- AliPythia derived from TPythia
  - High level interface to Jetset and Pythia
  - Tailored to our special needs:
    - generation of hard processes (charm, beauty, J/ψ..)
    - selection of structure function
      - Plan to implement nuclear structure functions
    - forced decay modes
    - particle decays … and more



One Step up: AliGenCocktail

- **Generation of Cocktail of different processes**
  - **Generation from parameterised transverse momentum and rapidity**
  - **Decays using JETSET**
  - **Rate and weighting control**



Design for Change and Code Reuse in the Simulation Framework

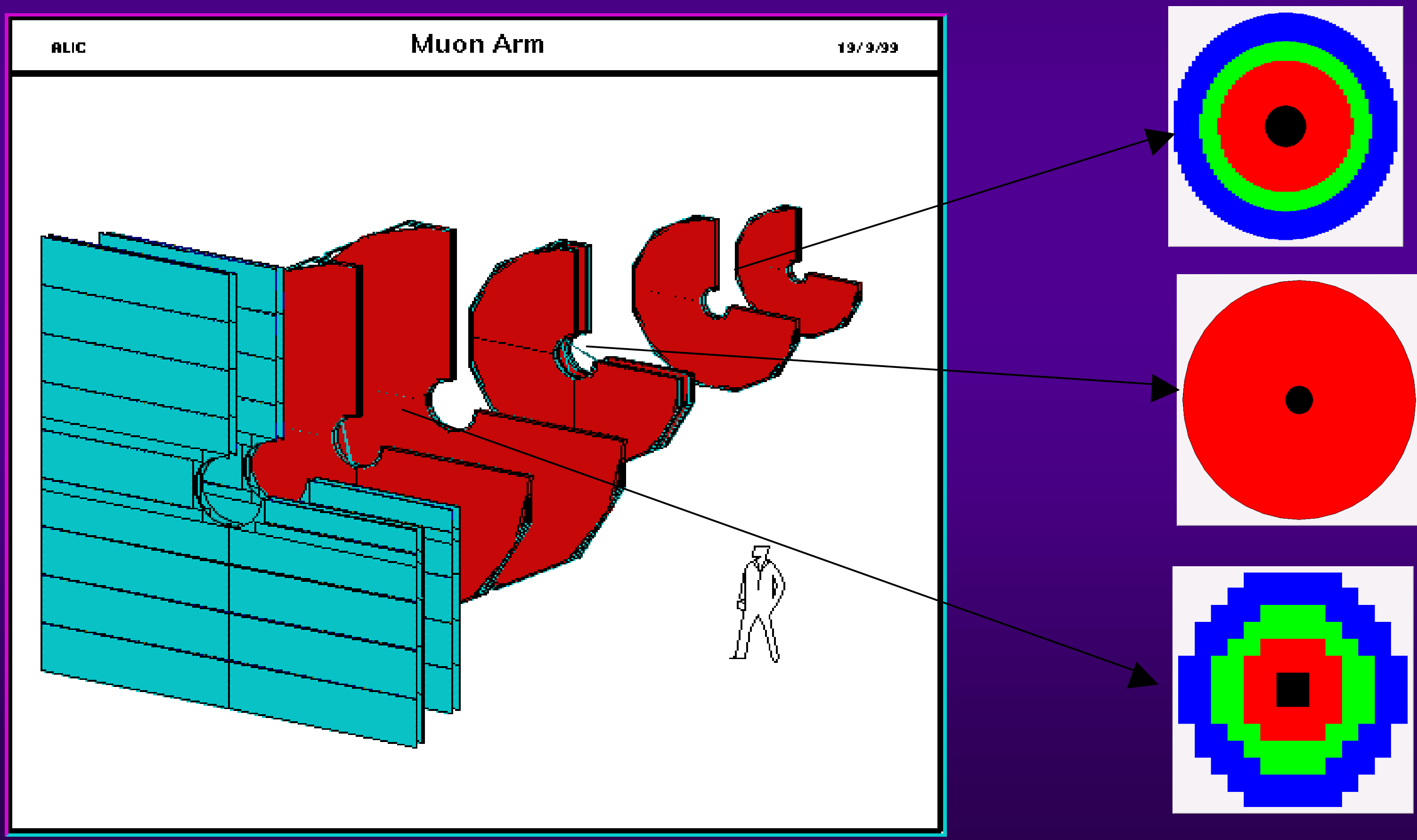
- Inheritance (formalized):
  - Different detector geometries and response simulation strategies are obtained through subclasses of AliDetector overriding CreateGeometry() and StepManager() methods.
- Additional member data in AliDetector subclasses
  - Allow for parametersation of geometry and response
  - Access methods allow reuse of these parameters in reconstruction and visualization.
- Composition and delegation
  - AliDetector subclass owns pointers to geometry and response objects
  - Very flexible run time configuration through abstract interfaces
  - Reuse of geometry and response behavior in reconstruction and visualisation.

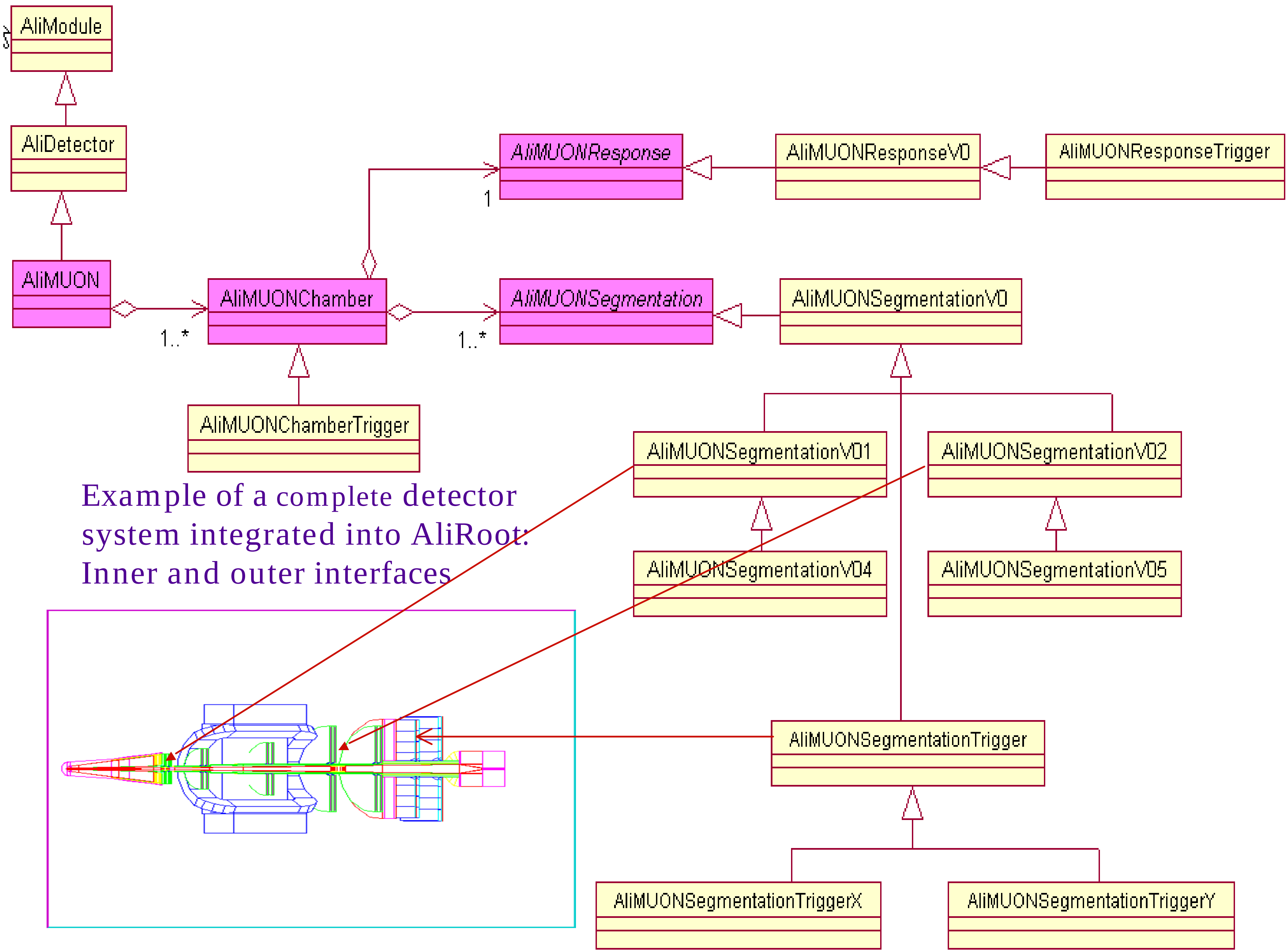
Common Base Classes for Simulation, Trigger and Reconstruction: Example Muon Arm

- Class design reflects detector physical layout, granularity and response (by composition)
- Segmentation and Response interface classes can be used for
  - Detector response simulation
  - Trigger Simulation
  - Hit Reconstruction
  - Visualization
- Current fully implemented in Muon-Arm and HMPID (+NA6i)

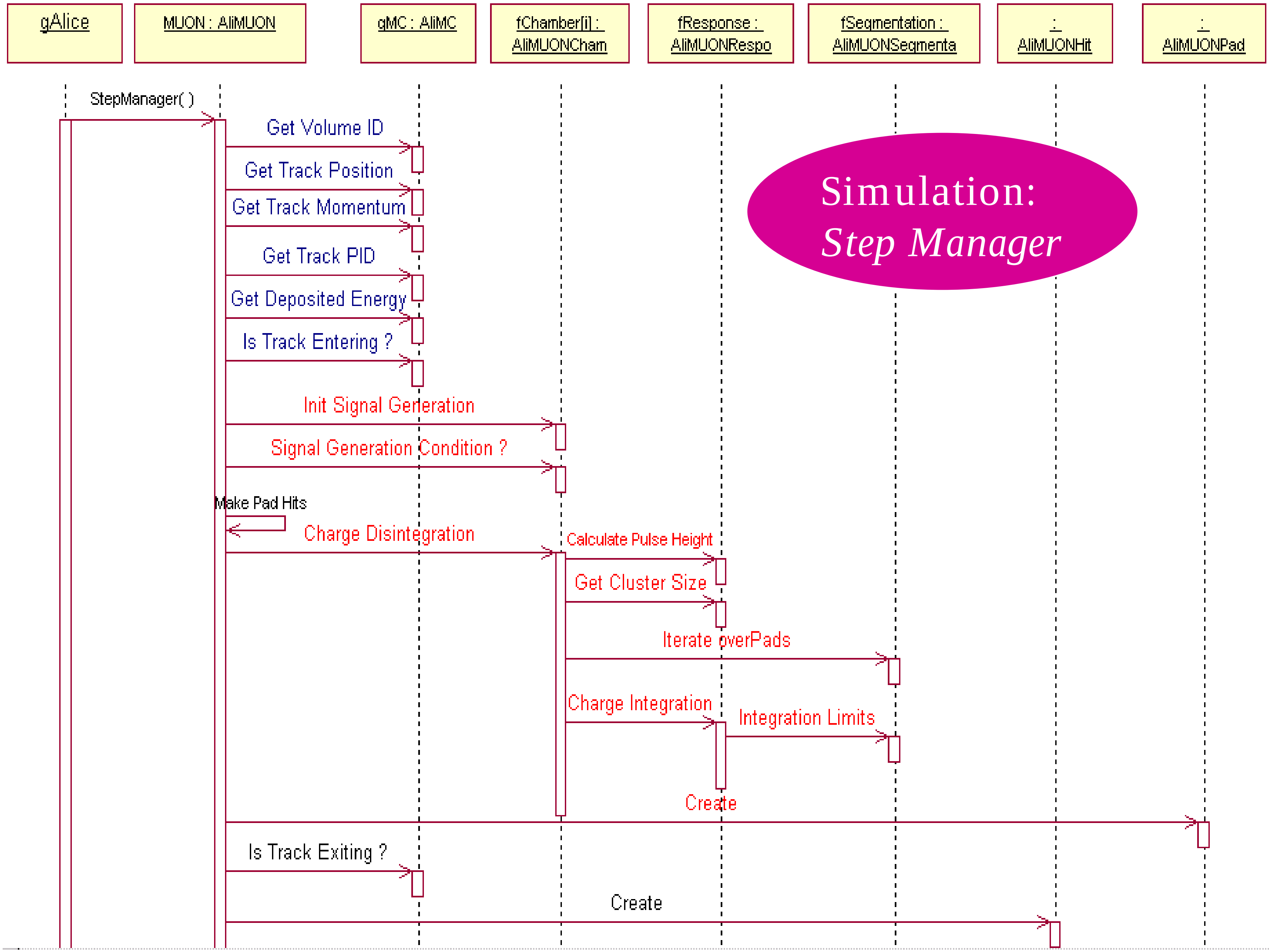
ALICE Muon Arm

Segmentation base class was developed out of the need to simulate CPCs and CSC with segmentation schemas changing from chamber to chamber, radially … and with time.

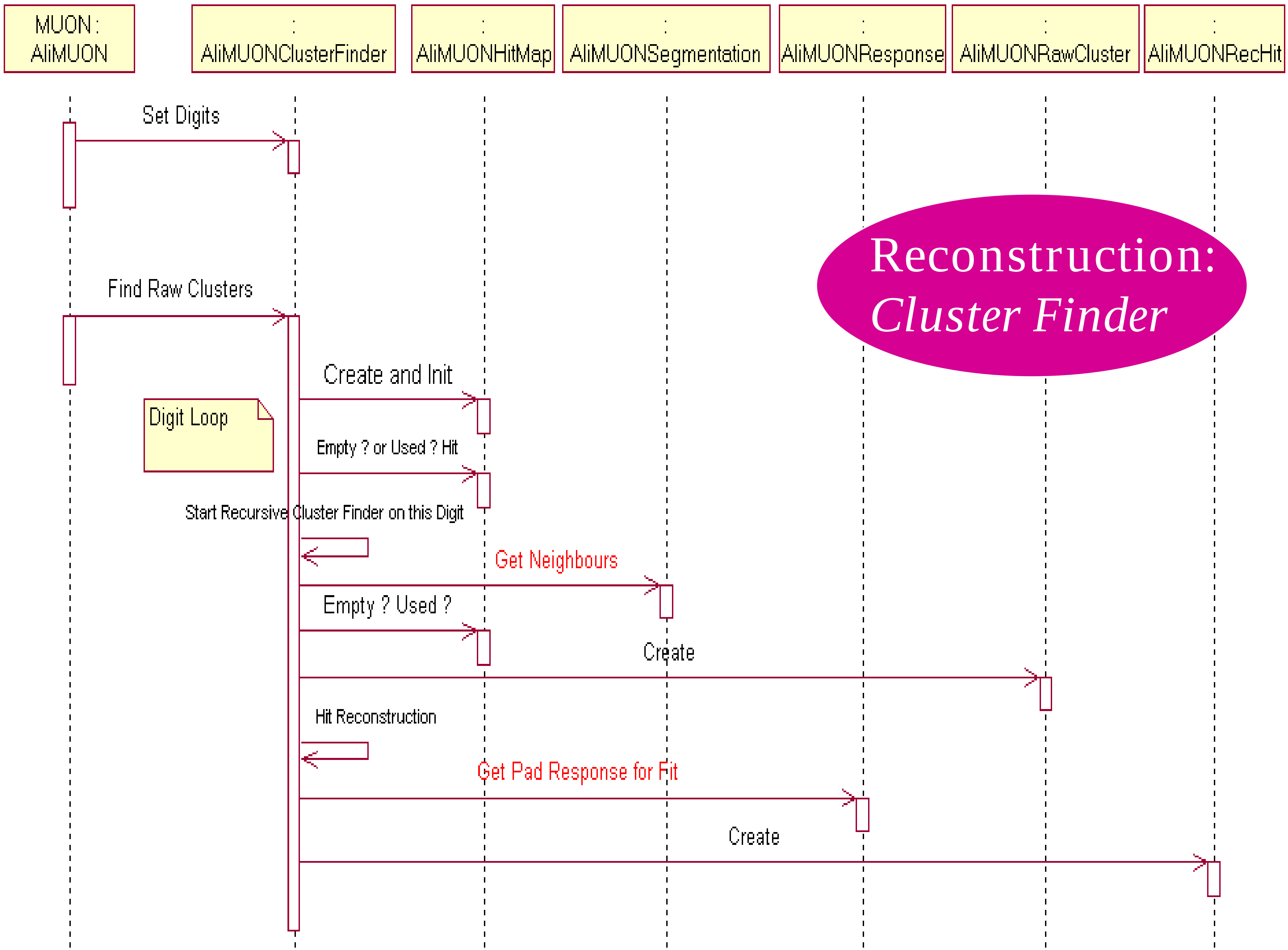




Example of a complete detector system integrated into AliRoot: Inner and outer interfaces


gAlice
MUON : AliMUON
gMC : AliMC
fChamber[i] : AliMUONCham
fResponse : AliMUONRespo
fSegmentation : AliMUONSegmenta
: AliMUONHit
: AliMUONPad
StepManager( )
Get Volume ID
Get Track Position
Get Track Momentum
Get Track PID
Get Deposited Energy
Is Track Entering ?
Init Signal Generation
Signal Generation Condition ?
Make Pad Hits
Charge Disintegration
Calculate Pulse Height
Get Cluster Size
Iterate overPads
Charge Integration
Integration Limits
Create
Is Track Exiting ?
Create
Simulation:
Step Manager

Reconstruction: *Cluster Finder*



Fast Simulation

- Not yet integrated into the AliRoot Framework
- Exists in ALICE as isolated modules (Root macros, kumacs, fortran-code)
- Was important in the (pre-) design phase of ALICE
- Will be again important for Physics Performance Report
  - Detector performance in terms of acceptances, efficiency and resolution is known
  - High statistic analysis without full MC sample.

Possible Design

- Design must allow for high flexibility if it should be of any use
- Main Components:
  - Fast detector class (through recursive composition)
  - Fast detector response (as abstract interface) for acceptance, efficiency, resolution
  - Black/Whiteboard for functions paramterizing the detector behavior
  - Abstract factory to create consistent set of components for each fast simulation task

Fast Simulation Draft Design