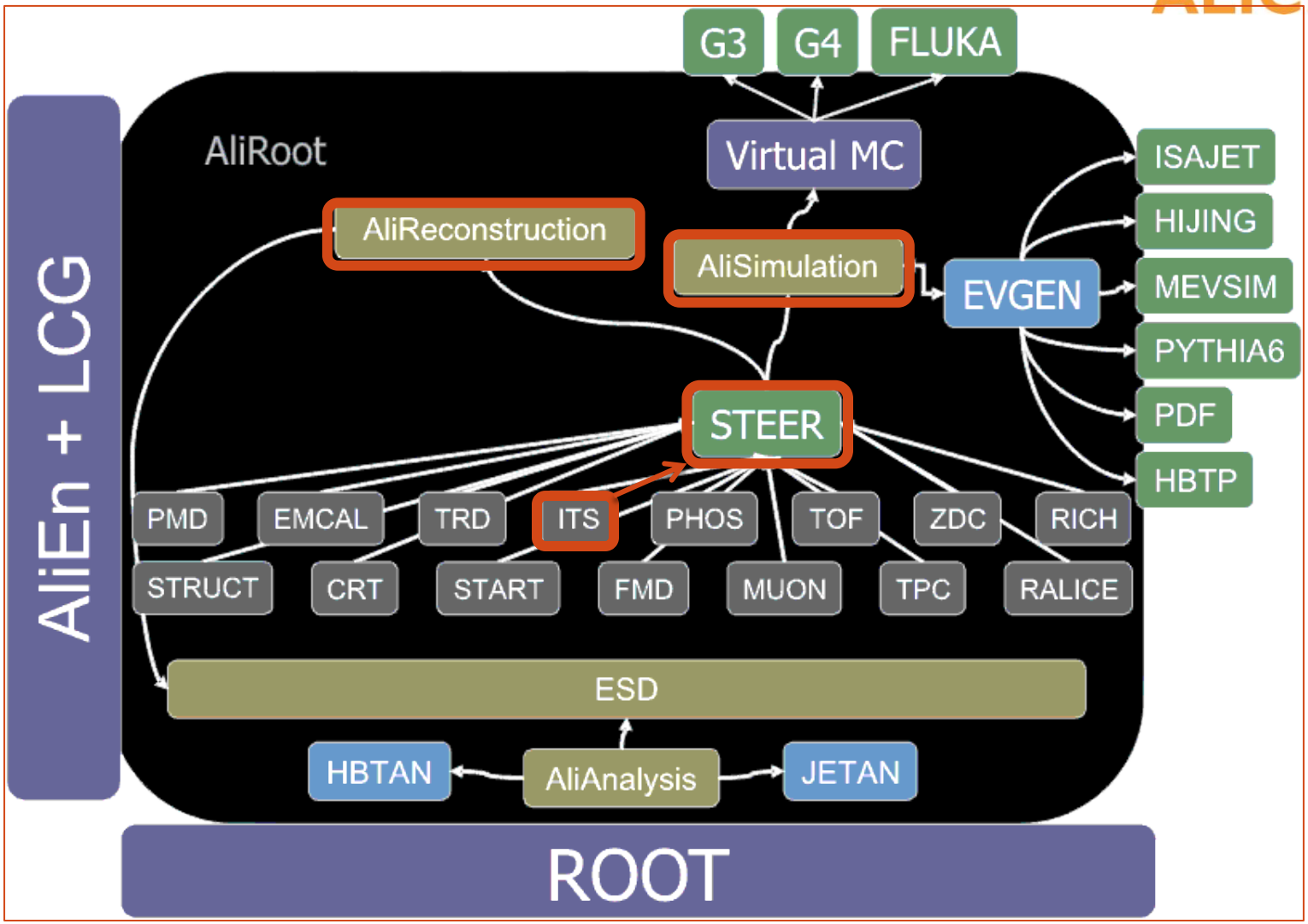
Structure and status of the ITS offline software

Massimo Masera

Torino

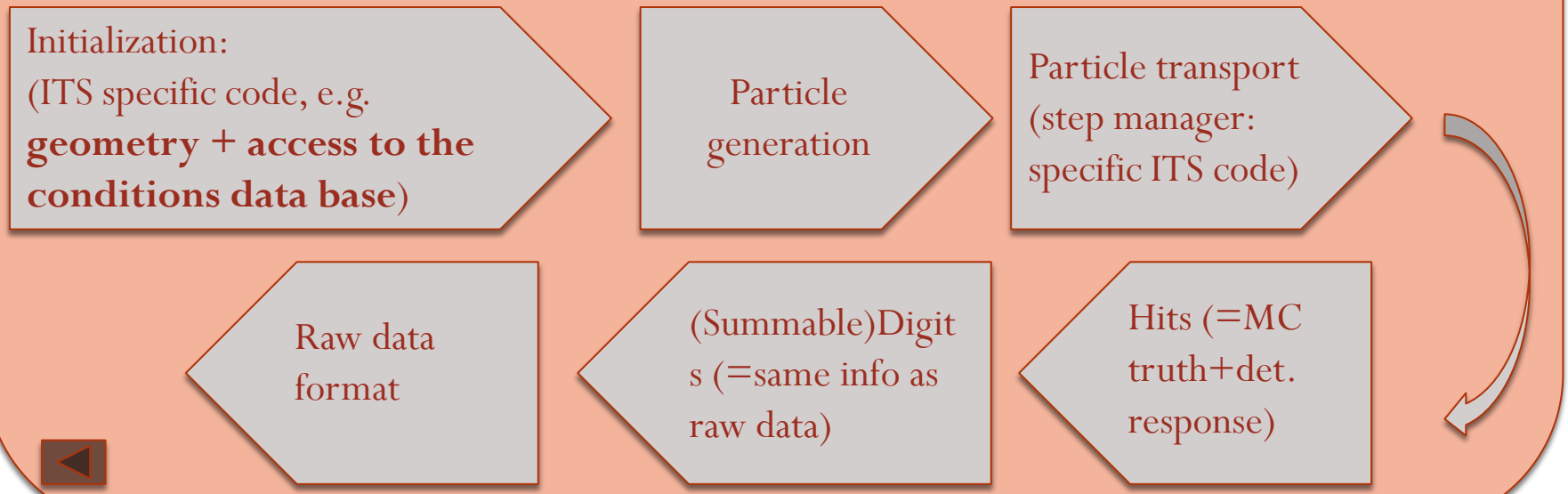
General AliRoot structure



AliSimulation

AliSimulation is the class which controls the simulation flow

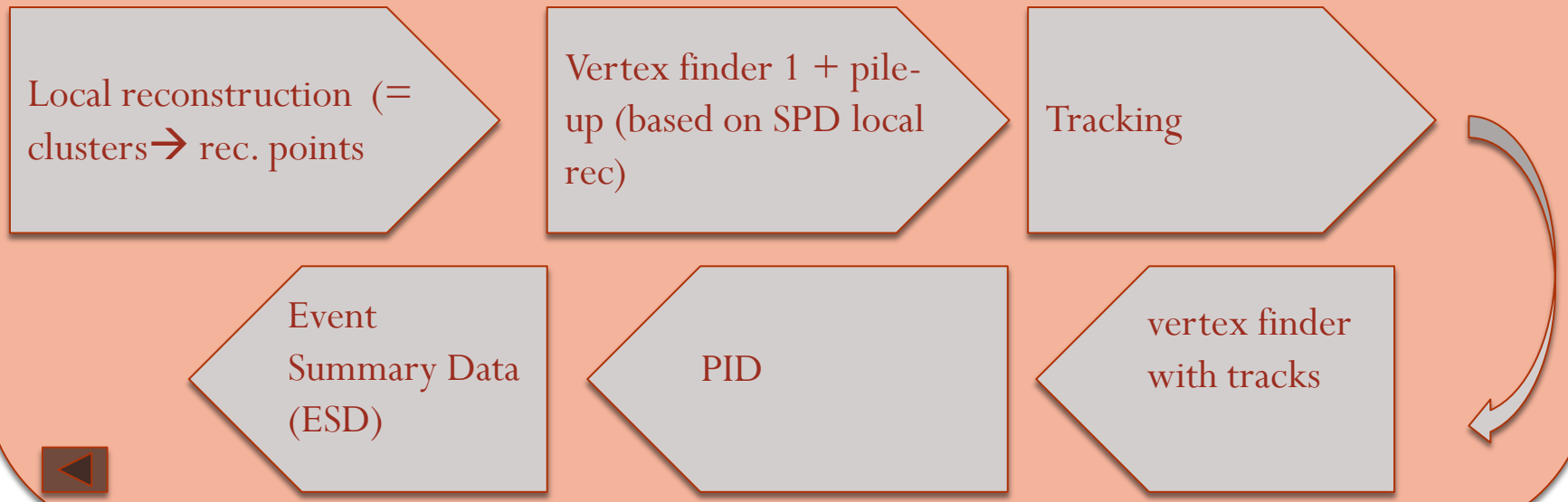
- it provides an interface to to MC generators
- it provides an interface to Transport Code (Geant)
- calls, for each simulation step, high level methods defined in AliModule and AliDetector classses, which are actually implemented in the detector daughter classes (AliITS)
- it has NO dependence on detector-specific code



AliReconstruction

AliReconstruction is the class which controls the reconstruction flow

- calls for each reconstruction step high level methods defined in STEER classes (AliRenctructor, AliVertexer, …), which are actually at least partly implemented in the detector daughter classes (AliITSReconstructor, AliITSVertexer, …)
- there is NO dependence on detector-specific code.
- Access to the Offline Conditions DB is done at STEER and Detector levels



ITS code

- The recommended way to access ITS code is through high level interfaces implemented in STEER
- There are NO dependencies on classes that are specific of other detectors
  - ✓ e.g. tracks prolonged from the TPC do not include TPC objects → information moving from a detector to another is coded in STEER in a detector independent way
  - ✓ this is an asset for any ITS upgrade simulation: new simulation software can be plugged in the present structure, provided it complies with interfaces defined in STEER
- Within ITS: 3 libraries: *ITSbase*, *ITSsim*, *ITSrec*.
  - ✓ rec and sim do not have inter-dependency
  - ✓ Software for the upgrade could be implemented in separate libraries with possible dependence on the existing code to maximize code re-use.
- Analysis code is outside the ITS module (Analysis framework + PWG specific classes)

Subdetectors

- ITS is seen by AliRoot as a single detector
- But, as we well know, ITS is made by three quite different subdetectors
  - ✓ The offline software tries to provide a common schema, through high level classes, for both the simulation and the local reconstruction.
  - ✓ The basic class organization for simulation and reconstruction is almost 10 years old (R. Barbera and B. Nilsen)
    - Subdetector specific code is implemented in daughter classes (e.g AliITSv11GeometrySXD, AliITSsimulationSXD, AliITSClusterFinderV2SXD)
    - The same structure holds for QA classes
    - The access to the Conditions DB is done separately for the 3 subdetectors, but the information is stored in classes which inherit from AliITSCalibration and AliITSResponse
- The software used to calibrate the detector at the run time is also a part of the offline framework, even though it is run indipendently via the Shuttle mechanism

Code management

- The ITS software is stored in the standard AliRoot subversion (svn) repository:
  - ✓ New code goes to the so called "svn trunk"
  - ✓ Trusted code migrates to the current release branch through a standard validation procedure ("savannah bug", core offline team discussion)
- The code in the trunk must compile without warnings (!), must obey the AliRoot Coding Convention and must pass the standard sim/rec test suites
  - ✓ These are very basic and loose conditions (no joke)
  - ✓ The "svn trunk" is intended for development and exchange among developers
- We strongly deprecate the development of private ITS/AliRoot versions: their integration at a later stage is usually a painful and error prone operation
- Several developers:
  - ✓ no formal rules: before committing it, the new code is typically seen and approved by the interested developers
  - ✓ to avoid clashes, commits to the trunk are done by (or agreed with) one person (m.m.)

Code development

- ITS code is used in production and its evolution is mainly driven by our understanding of the data (i.e. what is needed is basically there, but there is space for improvements)
- Monthly ITS offline and alignment meetings + occasional phone/EVO meetings on specific issues
- 220 classes in the ITS, 50 of them were modified in the last 6 months. We still have obsolete code in the repository that should be removed .
- In the last 6 months there were contributions from 16 people, even though the main developers are 4-5
- There are formal responsibilities only for the alignment (A. Dainese), the geometry (M. Sitta) and for subdetector-specific software (calibration, cluster finders, simulation): D. Elia, SPD; F. Prino, SDD; E. Fragiacomo, SSD.
- For the rest, development is done according to clear areas of interest (e.g. vertexing, pile-up, tracking, trackleting, plane efficiency studies, trigger implementation) without particular formalisations.