R3BRoot Framework

D. Kresan
GSI, Darmstadt

First R3BRoot Development Workshop
July 28 — 30, 2015
GSI, Darmstadt

Outline

- Introduction to concept
- Relation to FairRoot
- Combined solution for R3B analysis
- Framework components
  - Analysis tasks
  - Data objects
  - Parameter containers
- Overview of the code structure
- Overview of Workshop

Introduction

- R3BRoot is software framework for simulations and data analysis of R3B experiments

- It is derived and based on the FairRoot framework — common functionality for FAIR experiments

- Modular structure, has no executables — dynamic shared libraries are loaded on demand, depending on the steering-macro

Introduction

Simulation

- Simulation is fully ROOT based
- Virtual Monte Carlo (VMC): easy switch between Geant3 / Geant4 (keeping physics input, detector geometry, field map, stepping implementation)
- Additional reconstruction stage in simulation: digitization — generate detector hits, based on Monte Carlo information

Introduction

Data analysis

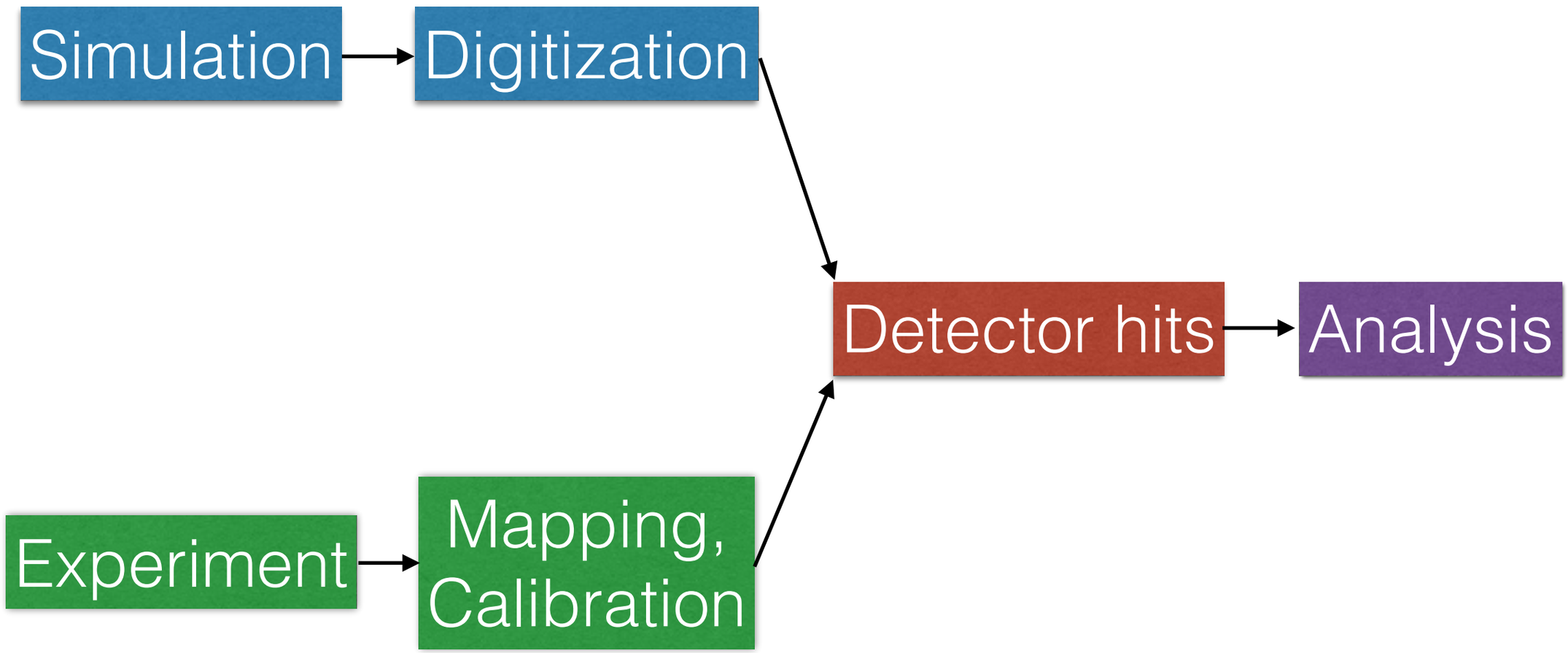
- Multiple data sources support: LMD file, Remote Event Server — will be substituted by R3B-specific UCESB source

- Analysis is organized in user-defined tasks (algorithms) — modular and flexible

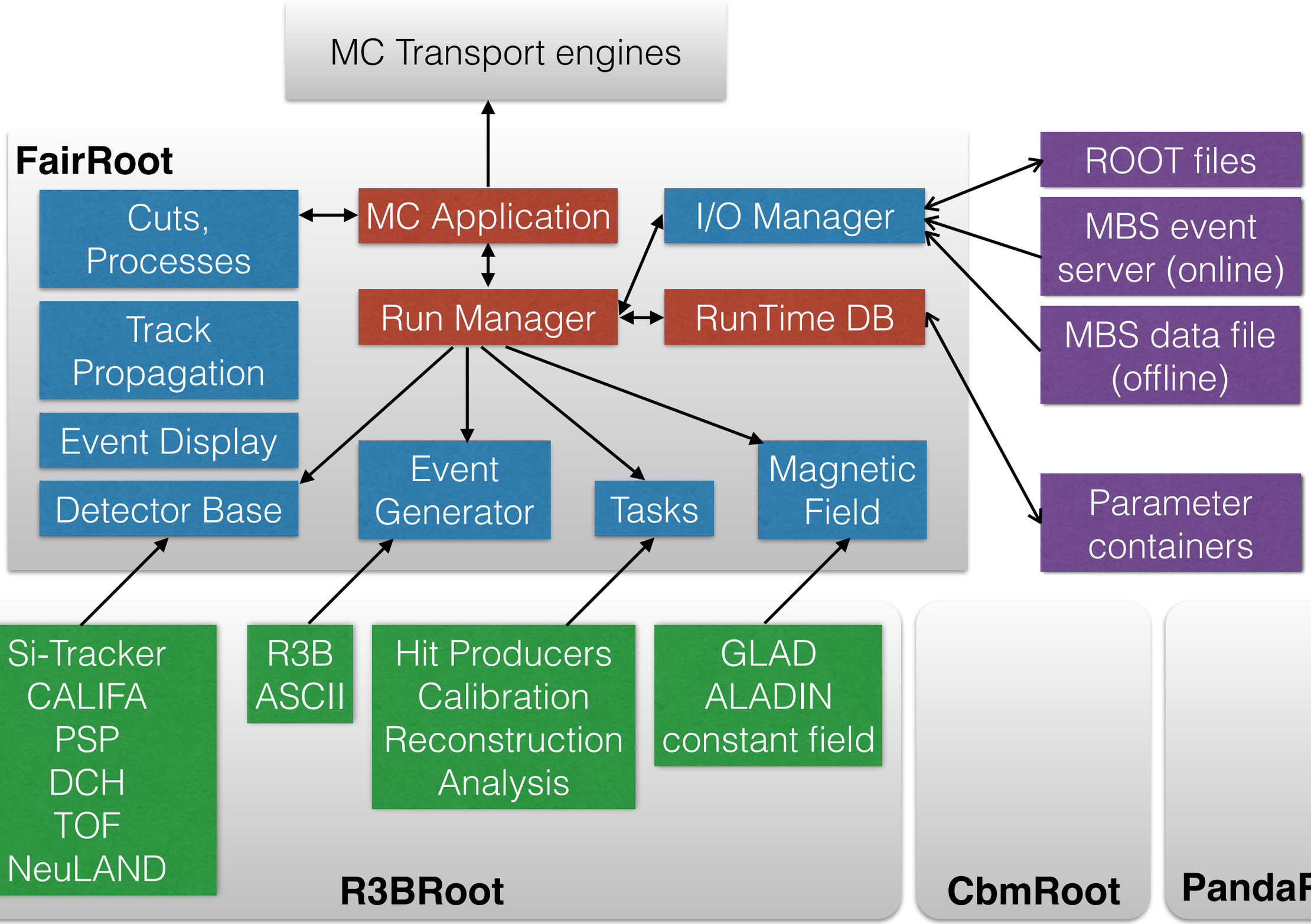
- The same algorithms implementation is used for simulation and analysis

Simulation and Data Analysis

Common analysis algorithms for simulation and experimental data



Relation to FairRoot (talk by F. Uhlig)



Combined solution for R3B

UCESB + R3BRoot + Tracker

UCESB

- MBS data unpacking. Channel mapping.

R3BRoot

- Calibration. Reconstruction. Analysis.

Tracker

- External tracker implementation.

Basic framework components

Analysis tasks

- Based on ROOT TTask mechanism
- Inherit from base class FairTask
- Modular scheme with hierarchy support (defined in steering macro)
- Framework takes over Input / Output implementation and parameter initialization
    - A task has access to all data produced by previous tasks during the same run (or from file)

Data objects

- Input / Output is ROOT based — TTree
- Arrays of data are stored in TClonesArray
- Data objects derive from TObject ROOT class
- Additional base classes: FairMCPoint, FairHit, R3BRawItem, etc…

Parameter handling

- Calibration parameters are to be stored in user-defined containers
- Handling is done using FairRuntimeDB singleton object
- Currently 2 implementations are supported:
    - Human readable ASCII file (suitable for small amount of parameters)
    - ROOT file — for storing C++ objects

Code structure

Code structure overview (1)

- Detector folders

  cal (CALIFA); xball (Crystal Ball); land, neuland (neutron detectors); dch (Drift Chamber); dtof, mtof, tof (TOF walls); los (start counter); gfi, mfi, psp, startracker, tracker (tracking detectors); passive (ALADIN, GLAD, Target)

- Contain implementation of detector-related algorithms (stepping, digitization, calibration), parameter containers (both for simulation and data analysis)

Code structure overview (2)

- field — field maps implementation and data files
- gconfig — configuration scripts for simulation engines
- geometry — ROOT files with detectors geometry, media file
- input — input files for simulation

Code structure overview (3)

- macros — steering macros
- plists — Geant4 physics lists
- r3bbase — Base classes for R3B
- r3bdata — (with subfolders per detector) data objects (MCPoints, Hits, etc…)

Code structure overview (4)

- r3bdb — parameter containers for calibration (currently CALIFA, NeuLAND, LOS)

- r3bgen — R3B-specific event generators for simulation

Upcoming…

- Git repository
    - R3BRoot will soon move to GitHub
    - Take over FairRoot workflow (talk by A. Rybalchenko)
- UCESB interface
    - B. Löher is implementing R3BUcesbSource — interface to UCESB software package
    - R3BRoot will read already unpacked and mapped detector items (raw hits)

Overview of the workshop program

Tuesday, July 28

- Setting up — preparing the environment, software installation

- Running analysis — structure of a steering macro, run TOF unpacker

- Adding data classes — create data class for storing TOF raw data, plug it into unpacker

Wednesday, July 29 (1)

- Adding analysis task — create task for calculation of TOF time calibration parameters, which runs in the same macro with unpacker

- Calibration and parameter handling — implement skeleton for storing of parameters using Runtime Database

Wednesday, July 29 (2)

- Applying calibration parameters — create new task and data class for time-calibrated TOF data. Use calibration parameters to calculate time [ns]

- Web-based event display — fill and publish histograms remotely during data unpacking

Thursday, July 30

- Combined analysis — create new analysis task accessing TOF and NeuLAND data

- Creating / accessing detector geometry — structure of a macro to create geometry file. Accessing geometry in the analysis using Runtime Database

- Event display — components of 3D event display. Adding new elements. Exercise with GUI

Next talk by R. Karabowicz on General Infrastructure

Wish you a successful workshop

Questions?