May 9th, 2023

Detector & MDI meeting

Release validation workflow

Initial proposal

N. Bartosik [a, b]

for the **Muon Collider Physics and Detector Group**

[a] INFN Torino *(Italy)* [b] CERN *(Switzerland)*

Release validation: the need

Our software for full detector simulation has evolved significantly over the past 3 years

Now starting a new chapter in release setup following the Key4hep model
using Spack package manager configured in mucoll-spack repository

We need to set up a validation procedure for every release change

↳ **raised a long time ago and becoming particularly relevant now**
with bigger changes coming to our software stack

This fits well with another deliverable: detector-performance report

↳ **requires a set of standard configurations for the full-simulation chain**
input → simulation → digitization → reconstruction → analysis

The two tasks are linked through the mucoll-benchmarks repository:

- **standard configurations for each step of the full-simulation process**
used as a reference for detector-performance plots, tutorials, etc.
- **dedicated scripts running these configurations with reduced inputs**

MuonCutil repository

Components of a release: what to validate?

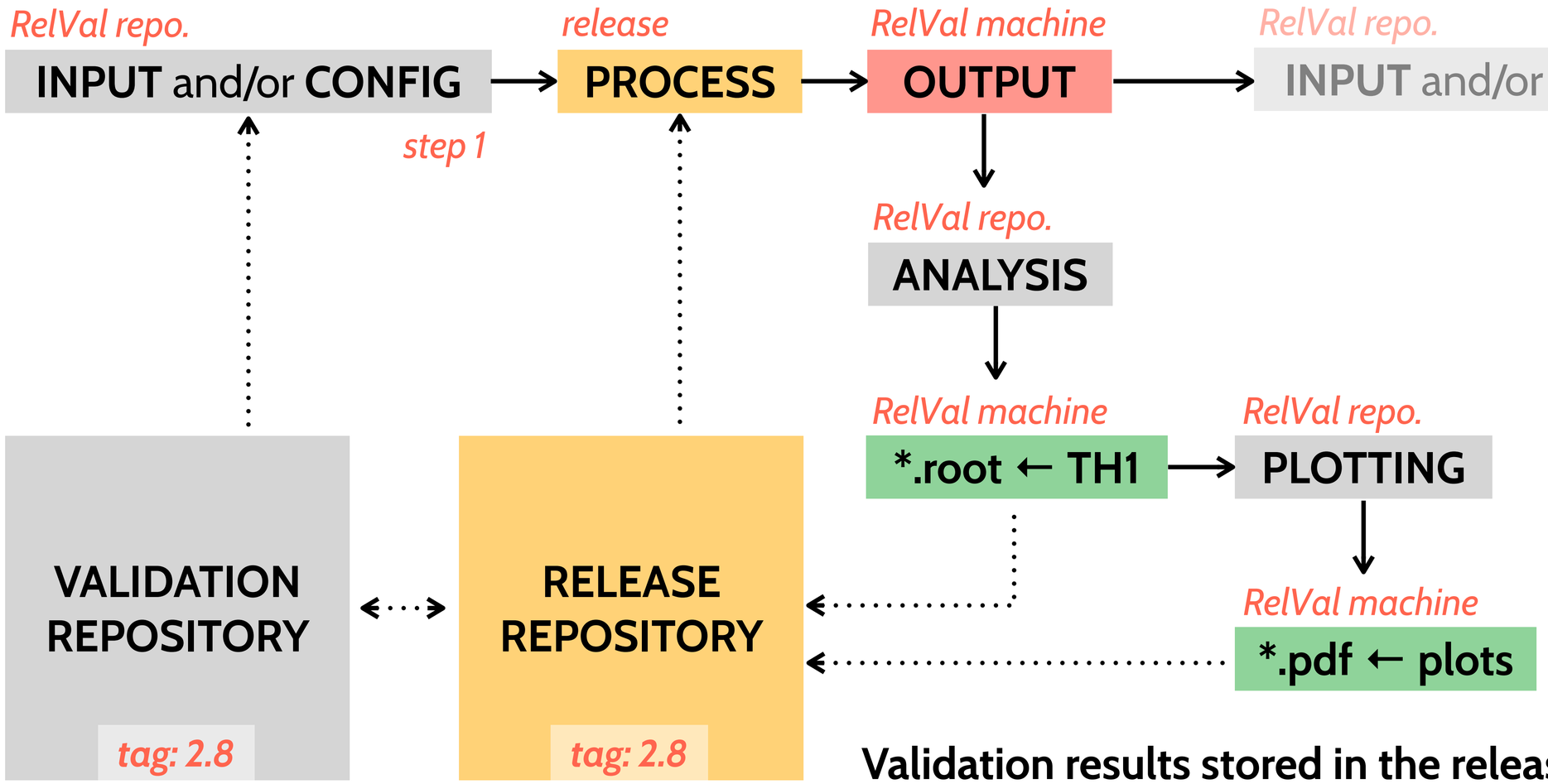
A complete release of Muon Collider simulation software combines several components:

- **environment** operating system + container *(Docker, Apptainer)*, CVMFS installation
  possible issues: *linking of system libraries, changes in system tools, etc. (→crash)*
- **base packages** ROOT, Geant4, DD4hep, PandoraPFA, ACTS, etc.
  possible issues: *change of interfaces (→crash), change of algorithms (→varied results)*
- **specific packages** Overlay, Digitizers, Filters, BIB subtraction, etc.
  possible issues: *change of algorithms, units, constants, defaults (→varied results)*
- **configurations** Overlay, Digitizers, Filters, BIB subtraction, etc. **+ baseline geometry**
  possible issues: *change of configuration parameters, paths (→varied results)*

Crash will be detected by simply running the workflow
Varied results can be detected only by comparing plots from 2 releases

We want the validation workflow to cover all the critical parts of each component
using a minimal number of steps and inputs to make it fast

Suggested workflow would have the following structure:



Validation results stored in the release repository

↳ ROOT files with TH1 allow to compare any 2 releases using overlay histograms in a single plot

Validation steps: initial proposal

The following validation workflows are proposed:

*validation plots produced **here***

1. **Particle gun samples:** generate → simulate → digitize → **reconstruct** ×3
   muons (tracker + yoke), photons (ECAL), pions (HCAL) [+ electrons to be added later]
   ↳ **plot reconstructed object properties:** angle, p_T, energy, reco. efficiency
   validating geometry layout + basic object reconstruction algorithms

2. **Physics signal sample:** generate → simulate → digitize → **reconstruct**
   μμ → H → bb process in Whizard
   ↳ **plot reconstructed object properties: b-jet** (angle, p_T, reco. efficiency), **bb** (angle, invariant mass)
   validating jet + secondary vertex reconstruction

3. **particles gun + BIB sample:** **convert** → **simulate** → **digitize** → reconstruct?
   ~10 muons with overlay of ~1% of BIB from MARS15 [+ few photons to be added later?]
   ↳ **plot basic properties of MCParticles, SimHits, RecHits:** position, time, energy + tracks/PFO objects?
   validating BIB-related code: conversion from MARS15/FLUKA, overlay, filtering/subtraction

More cases can be added as more features are added as baseline in the release
BIB subtraction in ECAL/HCAL, electron reconstruction, etc.