FUTURE CIRCULAR COLLIDER

2nd FCC Italy & France Workshop

NEW DEVELOPMENTS IN FULL SIMULATION SOFTWARE FOR FCC

Author: Enrico Lupi
Supervisors: Alvaro Tolosa Delgado, Brieuc Francois
Special Thanks: Juan Miguel Carceller

INTRODUCTION
THE GOAL OF THE PROJECT

The main pupose of this project is developing new tools for the validation of the **FCC software**.

What is validation?

Fundamental step in the software development lifecycle

Ensures the final product meets the **user's need** and fulfills its **intended purposes**

And for physics?

Making sure that the physics **results** obtained from the simulations are **compatible with our expectations**

How to achieve it?

GitLab CI/CD pipeline to run daily automated tests

Results comparing the output of new versions of the stack with reference stable ones can be checked on **website**

INTRODUCTION
THE GOAL OF THE PROJECT

The main pupose of this project is developing new tools for the validation of the **FCC software**.

What is validation?

Fundamental step in the software development lifecycle

Ensures the final product meets the **user's need** and fulfills its **intended purposes**

And for physics?

Making sure that the physics **results** obtained from the simulations are **compatible with our expectations**

How to achieve it?

GitLab CI/CD pipeline to run daily automated tests

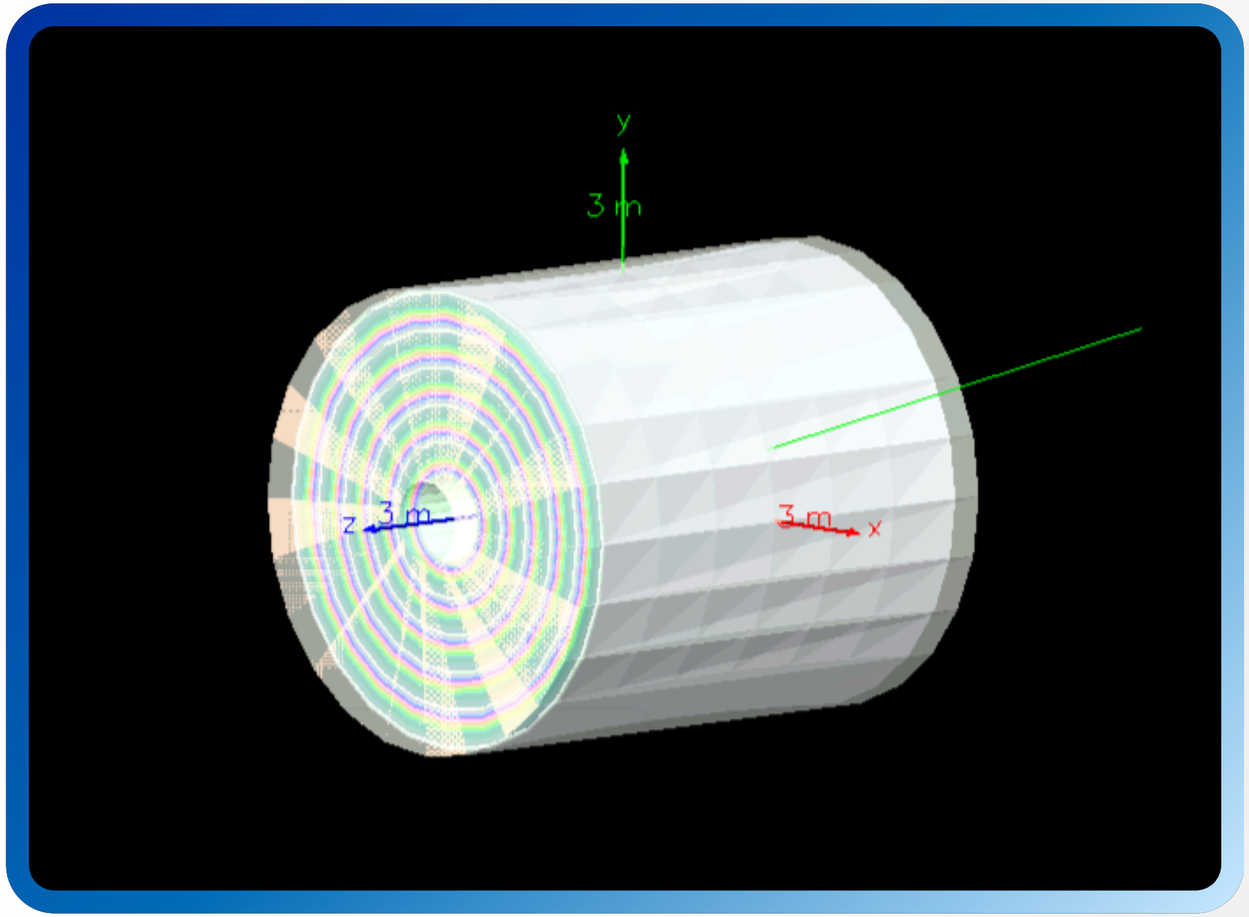
Results comparing the output of new versions of the stack with reference stable ones can be checked on **website**

WARNING!

This pipeline only validates the physics!
The software itself is tested by cron jobs in other repositories

DO WE REALLY NEED THIS? **WELL...**

- Found bug in *Geant4* navigation that was distorting the tracking and interaction of particles.
- Compilation was still successful and no run time error appeared
- Only possible to spot by **looking at physics** quantities
  - Problem was spotted “by chance” when shooting γ-rays at θ = 90° , when it became obvious
- Thanks to the physics validation system, the reaction time will be **~ 1 day!**



Drift Chamber detector containing the shape (not visible) responsible for the bug

HOW DOES THE PIPELINE WORK?
VARIABLES

The pipeline acts as a bash script executed in the pipeline runner.

The behaviour of the script is controlled by specific **variables** defined at the start of the file. Here are the most important...

Different pipeline schedules can be instantiated using different variable values.

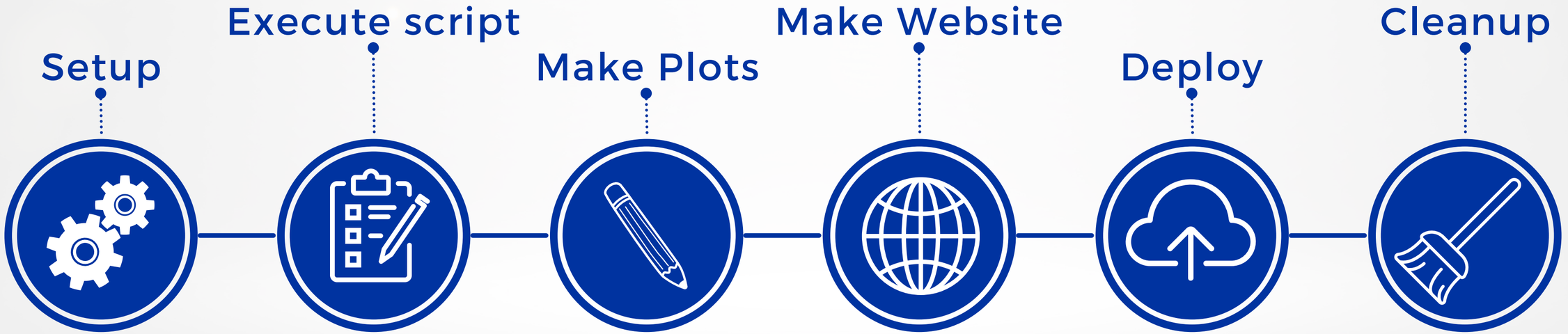
1. **VALIDATION_JOB_TYPE:**
   Which type of validation job to run. In order to use the new version of the pipeline, select *run_script*
2. **VERSIONS:**
   List of detector versions that need to tested, separated by a comma (e.g. "ALLEGRO_o1_v03, IDEA_o1_v03, CLD_o3_v01")
3. **MAKE_REFERENCE_SAMPLE:**
   Whether to store the output of the simulation and reconstruction phase as a reference for future use or new results to be checked
4. **TAG:**
   Which tag to use for the key4hep release, identified by its date
5. **COMPARISON_TEST:**
   Which test to use to compare histograms:
   Exact match, Chi squared or Kolmogorov-Smirnov

HOW DOES THE PIPELINE WORK?
STAGES

The pipeline is divided into stages, logically distinct steps that are run in a **specified order**.

The execution of the stages can depend on the global pipeline variables set at the start or on the success of the previous stages, providing a way to handle different situations.

Here are the stages when **VALIDATION_JOB_TYPE** is set to **run_script**:



HOW DOES THE PIPELINE WORK?
STAGES

SETUP:

- Clean the working directory
- Download the *key4hep-reco-validation* repository

EXECUTE SCRIPT:

- Execute scripts containing:
  - Simulation + reconstruction step
  - Analysis step to fill histograms in output ROOT file

- The script is **not** part of the pipeline itself.
  In particular, *simreco* script should be written by detector experts and live in *FCCConfig* environment

- Optionally move output ROOT files to specific reference folders if MAKE_REFERENCE_SAMPLE variable is set to "yes"

HOW DOES THE PIPELINE WORK?
STAGES

MAKE PLOTS:

- Compare histograms in output ROOT file to reference ones
- Plotsthe two distributions with different background color depending on the result of the comparison: matching ⚪, not matching 🔴, missing reference 🟡
- Send warning email in case one or more tests fail

MAKE WEBSITE:

- Create the html files for the static website.

DEPLOY:

- Deploy the website online

CLEANUP:

- Remove useles files from working directory

SCALIBILITY
ADDING NEW TESTS

The pipeline has been designed with **flexibility** in mind, so that adding new detector concepts or subsystems to be tested would be as easy and straightforward as possible.

New Detector Option

Simple 3 steps process:

- Create appropriate **bash script** containing simreco and analysis stage
- Properly place it in the **correct folder** of the repository following the naming convention
- Add the detector version to the **VERSIONS** pipeline variable

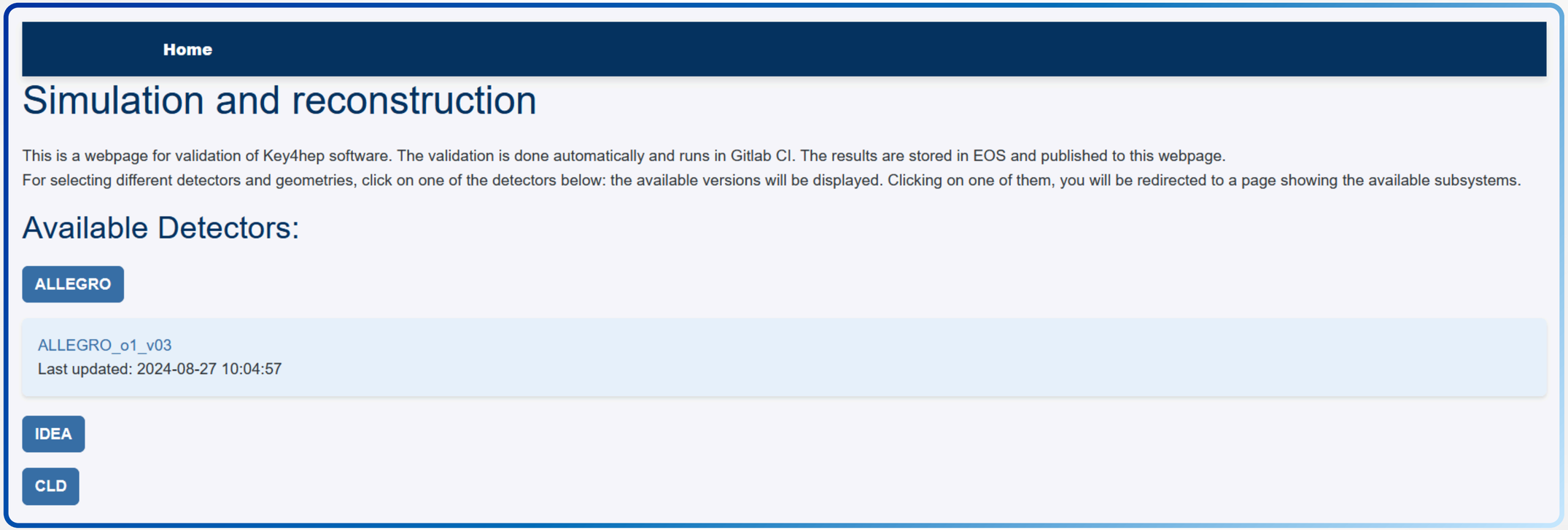
New Subsystem

Even simpler, only the analysis file for the specific detector needs to be changed:

- Just add the correct **histogram declarations and fill them** in the analysis script

RESULTS
THE VALIDATION WEBSITE

The website provides an accessible and easy-to-navigate way to check the results of the pipeline.



Home page, containing the list of available detectors

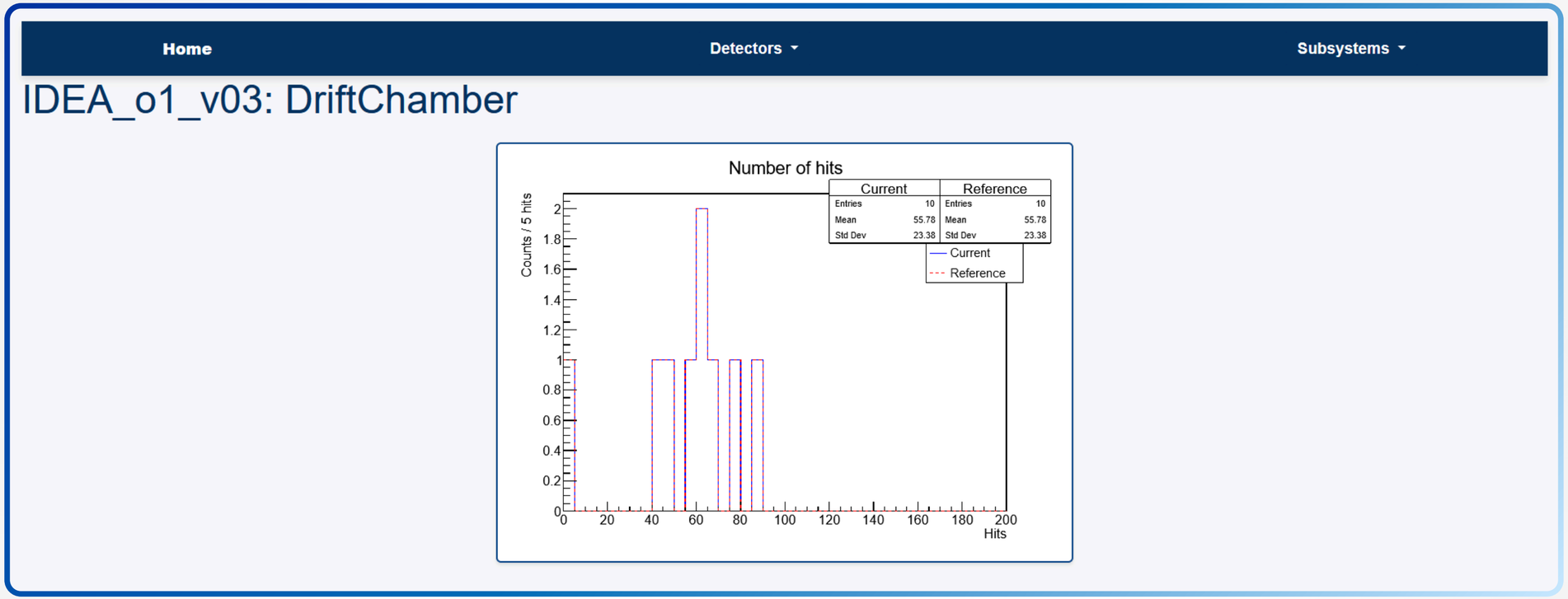
RESULTS
THE VALIDATION WEBSITE

The website provides an accessible and easy-to-navigate way to check the results of the pipeline.

Detector version page, containing metadata information and list of available subsystems

RESULTS
THE VALIDATION WEBSITE

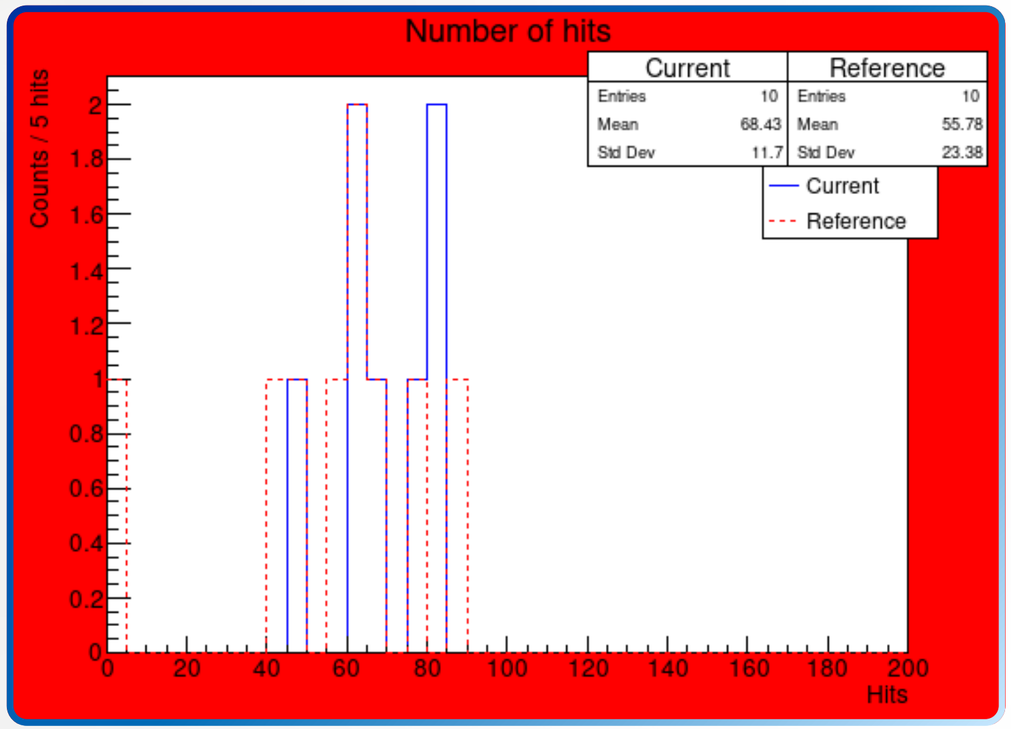
The website provides an accessible and easy-to-navigate way to check the results of the pipeline.

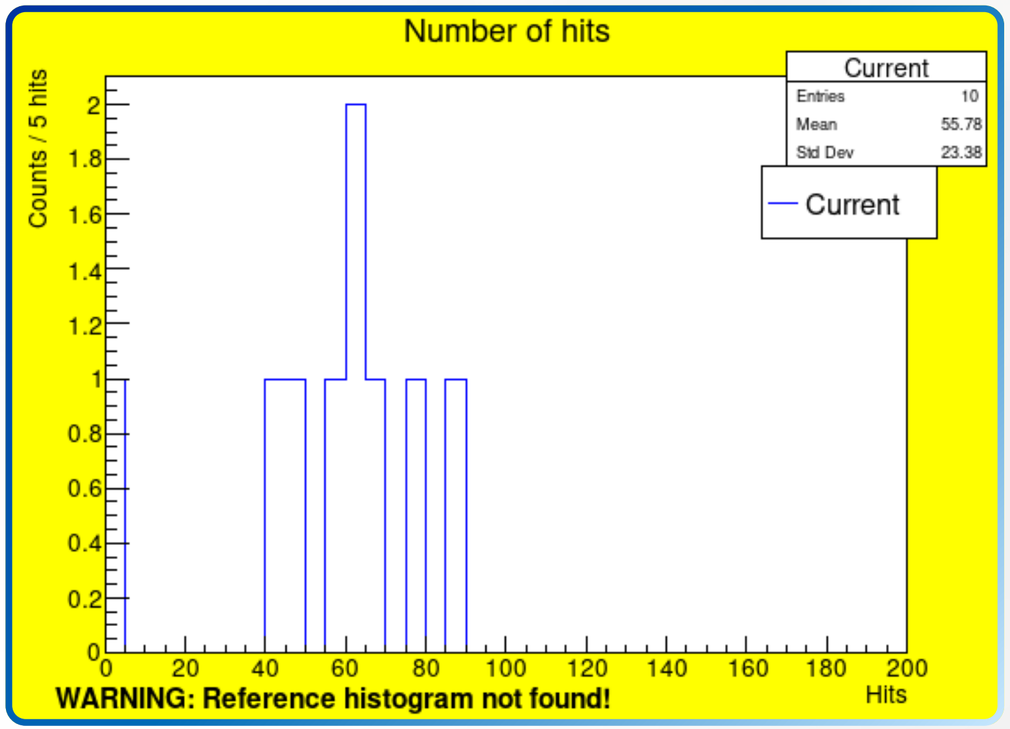


Subsystems page, containing the histogram plots.
In this example, the new and reference histograms match.

RESULTS
THE VALIDATION WEBSITE

The website provides an accessible and easy-to-navigate way to check the results of the pipeline.





Examples of plot appearance when histograms do not match (left)
or when the reference is missing (right).

RESULTS
THE VALIDATION WEBSITE

Here are the plots already available...

Detector Version	Subsystem	Histograms

ALLEGRO_o1_v03	• Electromagnetic Calorimeter - Barrel	• CaloCluster Energy
		• CaloTopoCluster Energy
		• ECalBarrelModuleThetaMergedPosition:
		  ◦ total Energy per evt
		  ◦ X, Y and Z position

IDEA_o1_v03	• Drift Chamber • Vertex Detector • Vertex Inner Barrel • Vertex Outer Barrel	Number of Hits
CLD_o3_v01	• Standalone ARC Detector	• Photon counts per event • Photon counts vs. θ • Photon counts vs. θ of incoming particle

...and many more to come!

OUTLOOK
CURRENT LIMITATIONS

NUMBER_OF_EVENTS pipeline variable exists, but sim digi reco scripts actually are fixed to **only 10 events**

- Could lead to **large fluctuations** that trigger notifications even though the physics is still valid
- Could make the system **blind to small** but relevant **changes** in the physics

Might be worth it to modify the scripts and let the number of events be an input

Pipeline relies on the software working as intended, and only checks the physics

Some cecks are implemented to handle **software failure**, but they are not working perfectly..

Pipeline **variables are global**: same event number, statistical test, significance level... for all detectors and subsystems

OUTLOOK
FUTURE DEVELOPMENTS

ERROR HANDLING SYSTEM:

- Implement fully functional check to handle software failure
- Avoid failure for one detector to affect the validation of others

MORE VALIDATION!

- The pipeline is ready, but there is still a lot of work to do...
- Add more tests and populate the validation website

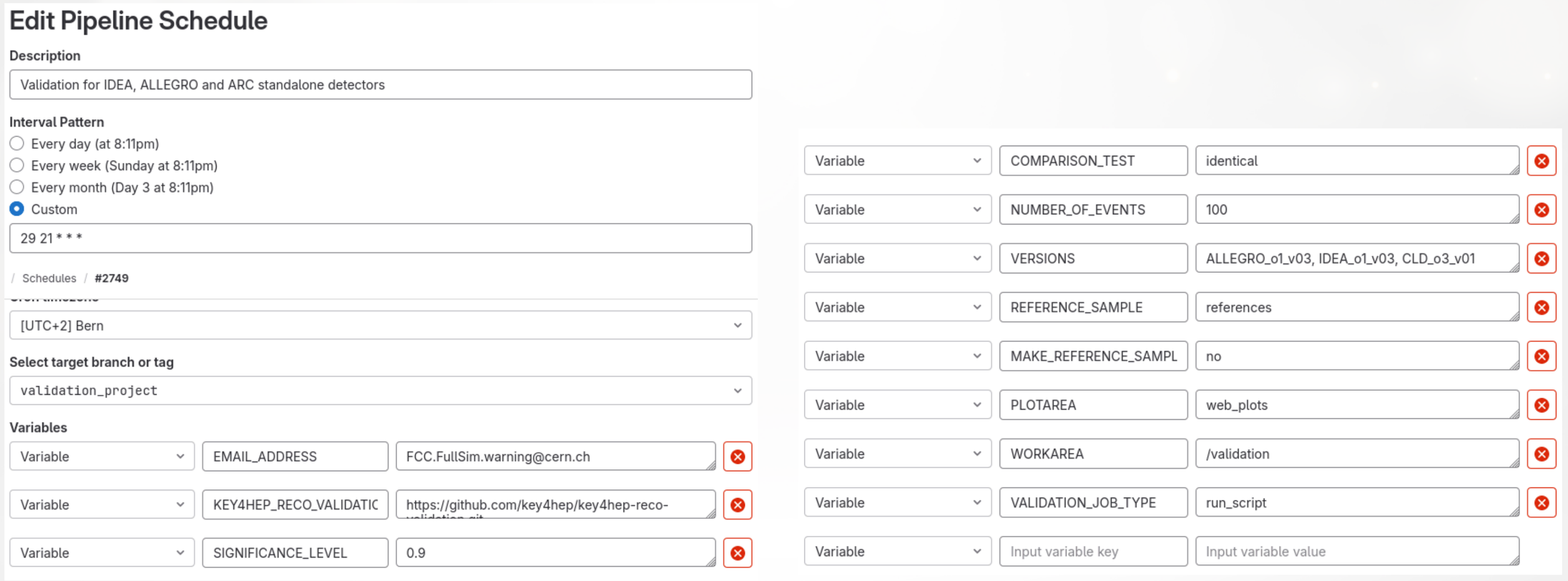
THANK YOU!

Contacts:
Enrico Lupi: enrico.lupi@cern.ch, enricolupi00@gmail.com

Links:
Valiation website: https://key4hep-validation.web.cern.ch/index.html
Key4hep-validation-reco repository: https://github.com/key4hep/key4hep-reco-validation

BACKUP SLIDES

HOW DOES THE PIPELINE WORK?
THE PIPELINE SCHEDULE EDITOR



SCALIBILITY
ADDING A NEW DETECTOR

The pipeline has been designed with flexibility in mind, so that adding new detector concepts to be tested would be as easy and straightforward as possible.

1. **CREATE BASH SCRIPT**
2. **ADD NEW SCRIPT TO KEY4HEP-RECO-VALIDATION**
3. **ADD NEW DETECTOR TO *VERSIONS* VARIABLE**

1 Create bash script to run in the **EXECUTE SCRIPT** stage
Goal: produce a properly structured ROOT file containing histograms

Two steps:

- **Simulation and reconstruction**
  usually done with script in FCCConfig, e.g. :
  ```
  source $FCCCONFIG/share/FCC-config/FullSim/ALLEGRO/
  ALLEGRO_o1_v03/ctest_sim_digi_reco.sh
  ```
- **Analysis**
  usually is done with seperate python or ROOT script

Histograms should be saved as TH1 into specific TDirectories corresponding to the subsystems under study for the plot stage to work correctly.

SCALIBILITY
ADDING A NEW DETECTOR

The pipeline has been designed with flexibility in mind, so that adding new detector concepts to be tested would be as easy and straightforward as possible.

1. **CREATE BASH SCRIPT**
2. **ADD NEW SCRIPT TO KEY4HEP-RECO-VALIDATION**
3. **ADD NEW DETECTOR TO *VERSIONS* VARIABLE**

2 The bash script needs to be saved in the correct subdirectory of **key4hep-reco-validation**

The repository's structure mirrors the one for **k4geo:**

- Go to scripts/FCCee/ and check if there already is a directory for the **geometry** or create one if needed

```
key4hep-reco-validation/
└── scripts/
    └── FCCee/
        ├── ALLEGRO/
        │   └── ALLEGRO_o1_v03/
        │       └── ALLEGRO_o1_v03_script.sh
        ├── IDEA/
        │   └── IDEA_o1_v03/
        │       └── IDEA_o1_v03_script.sh
```

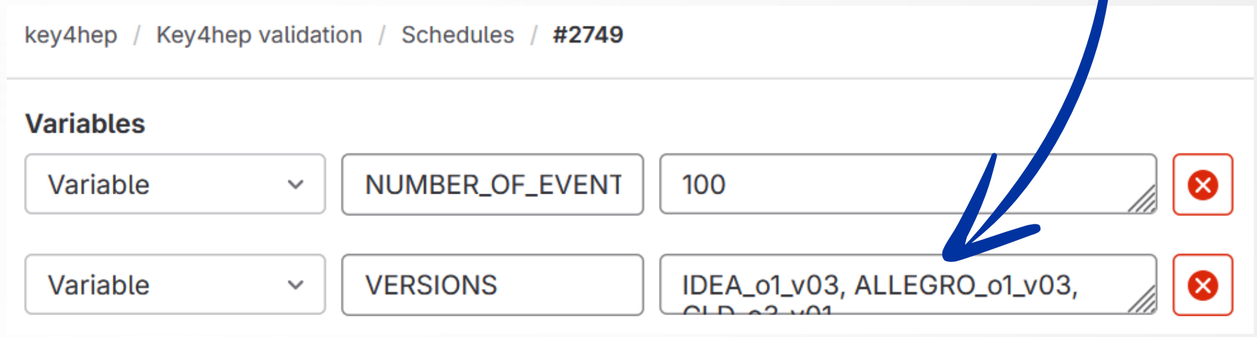
- Create a subdirectory for the specific version

SCALIBILITY
ADDING A NEW DETECTOR

The pipeline has been designed with flexibility in mind, so that adding new detector concepts to be tested would be as easy and straightforward as possible.

1. **CREATE BASH SCRIPT**
2. **ADD NEW SCRIPT TO KEY4HEP-RECO-VALIDATION**
3. **ADD NEW DETECTOR TO *VERSIONS* VARIABLE**

3 Add the name of the version to be tested in the **VERSIONS** list variable



This step can only be done by the owner or maintainer of the project, so please get in contact with them

SCALIBILITY
ADDING NEW VARIABLES

Adding a new variable or even subsytem to analyze is even easier, as you only need to change one file.

1 Open the analysis script (e.g. IDEA_make_TH1.py)

2 Add the correct TDirectory and histograms initialization at the beginning

```
dir_DCH = outputFile.mkdir("DriftChamber")

hist_dch_hits = ROOT.TH1F("h_DriftChamber_hits",
                          "Number of hits;Hits;Counts / 5 hits",
                          40, 0, 200)

dir_list.append(dir_DCH)
histo_list.append([hist_dch_hits])
```

3 Add the analysis inside the event loop

```
# Loop over dataset
for event in podio_reader.get("events"):

  ##############################################################################
  ##                                                                          ##
  ##                BEGIN: Drift Chamber Event Loop                           ##
  ##                                                                          ##
  ##############################################################################

  n_hits = 0
  for dch_hit in event.get("DCHCollection"):
    n_hits += 1
  hist_dch_hits.Fill(n_hits)
```

And that's it! Only add the new sections in the correct file without modifying anything else.