CMS OT Hybrids
Graphical User Interface for production testing

I. Mateos

Irene.mateos.dominguez@cern.ch

Content

1. Introduction
2. Hybrids test system architecture
3. Software structure
4. Testing at the hybrid level.
   - FPGA firmware.
   - Low level software Ph2_ACF.
   - Test card control libraries.
5. Communication between Ph2_ACF And GUI.
6. Testing at the system level (GUI).
7. Summary.

https://gitlab.cern.ch/imateosd/gui-for-hybrids-testing-on-crate-systems
*Not for distribution, still under development.

- Over 50 000 hybrids will be manufactured for the Phase 2 Upgrade.
- They need to be tested in order to be used in the detector.

2S module	Quantity required [pcs]
2S front-end hybrid **1.8 mm; 4 mm** left and right variant	2x 8750 + 2x 550
2S service hybrid **1.8 mm; 4 mm**	8750 + 550
Total 2S type hybrids to be tested	**27900**

PS module	Quantity required [pcs]
PS front-end hybrid **1.6 mm; 2.6 mm; 4mm** left and right variant	2x 990 + 2x 1750 + 2x 4000
PS readout hybrid **1.6 mm; 2.6 mm; 4mm** variant	990 + 1750 + 4000
PS power hybrid **common**	6740
Total PS type hybrids to be tested	**26960**

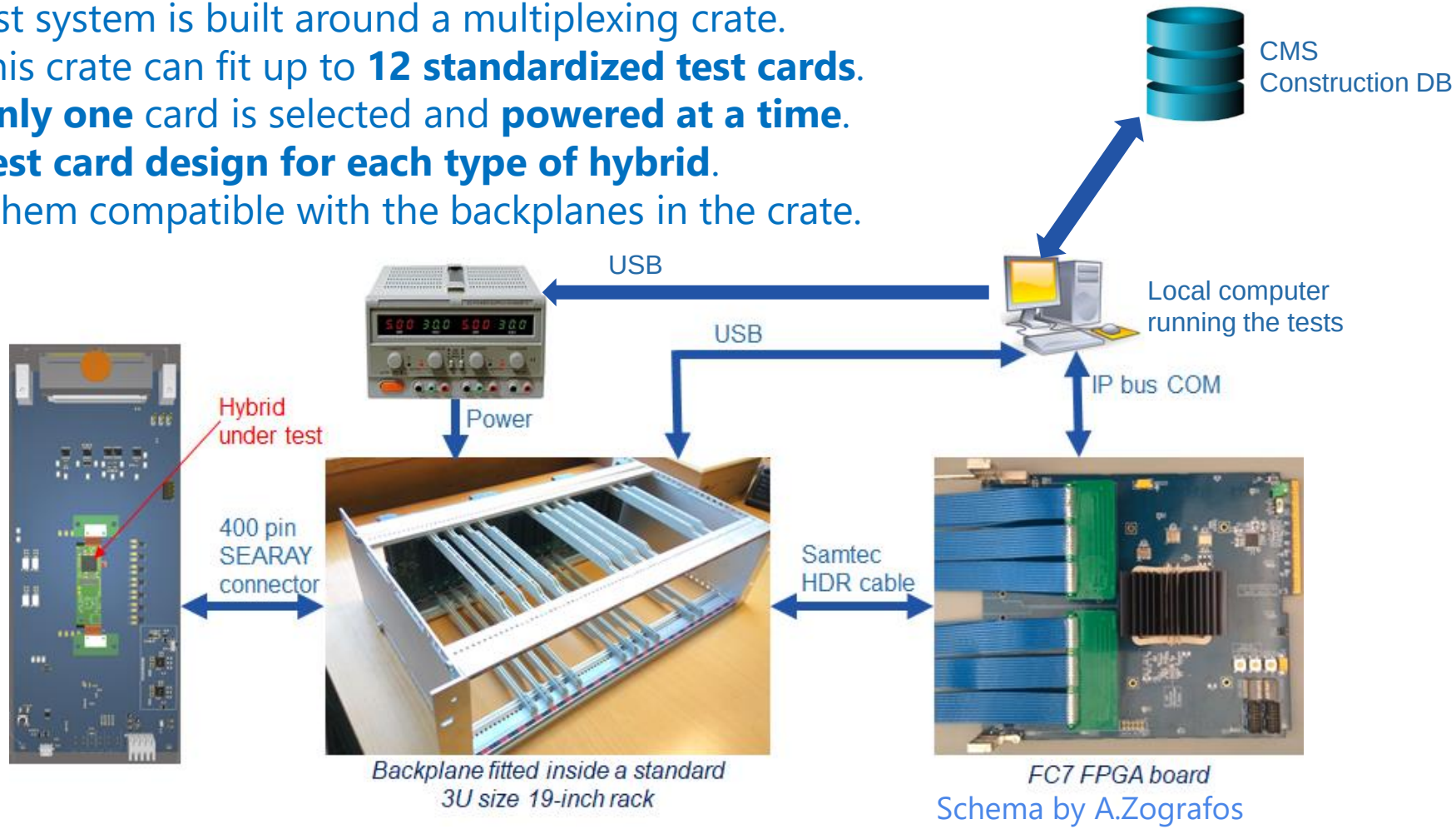
- To solve this problem, a test system has been developed to allow for the testing of the hybrids during production. This system is based on a **multiplexing crate**

The hybrids will be tested by the manufacturer. This creates the need for a complete testing system that can be used by a non-expert.

Goals for the software:

- Simplification of test process.
  - Single software for all types of hybrids.
  - Easy-to-use graphical user interface.
  - Tool to walk the user through the test run.
- Transparency of the test system to the user.
- Upload of test results and test measurements to database
  - Allows for monitoring of test progress from CERN.
  - Early detection of problems and quick reaction time.

- The test system is built around a multiplexing crate.
  - This crate can fit up to **12 standardized test cards**.
  - **Only one** card is selected and **powered at a time**.
- **One test card design for each type of hybrid**.
- All of them compatible with the backplanes in the crate.



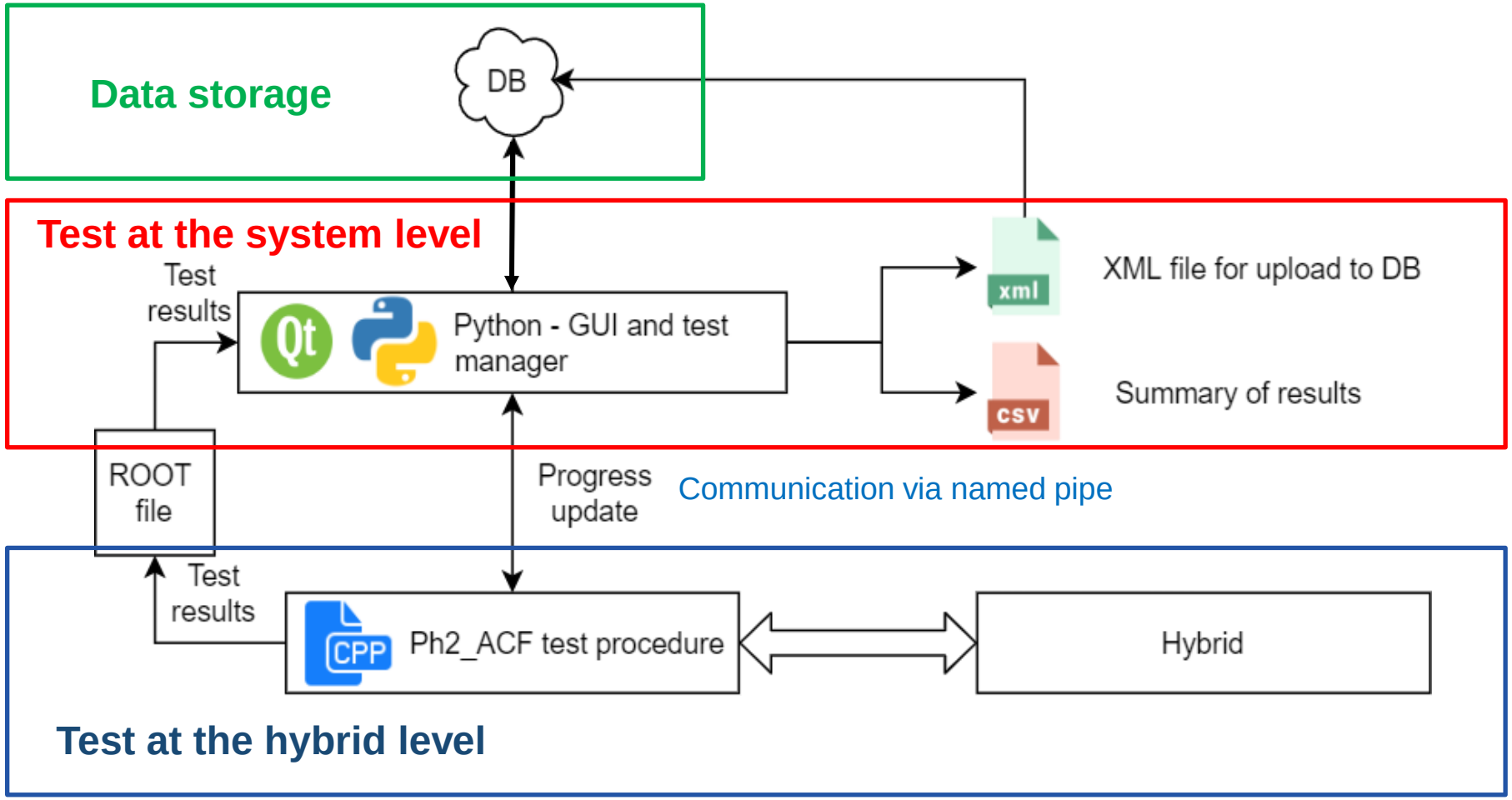
Schema by A.Zografos

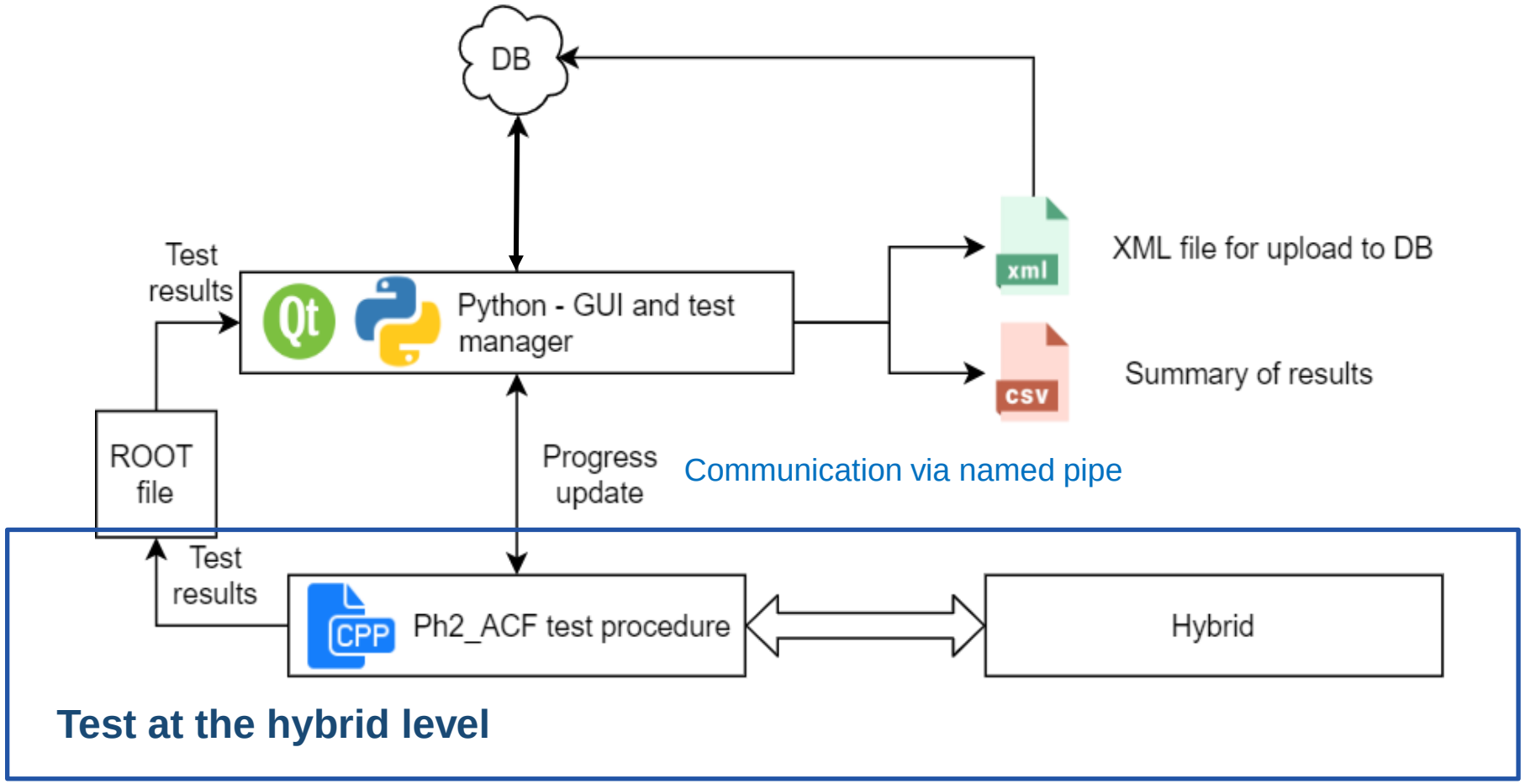
- Each crate allows for the **serialized testing** of 12 hybrids.
- Default setup will have three crates: allows for parallel testing processes (one process of serialized testing on each crate).

The software infrastructure can be divided into two main parts:

1. Testing at the hybrid level
   - FC7 FPGA firmware with specific blocks to interact with the hybrid under test.
   - Low level software.
     - Ph2_ACF test procedure to test individual hybrids via the FC7.
     - USB libraries to control the test card-specific features.

2. Testing at the system level (GUI)
   - Management of the testing of the hybrids at a larger scale.
   - Configuration of setup: power supplies, firmware and test procedures.
   - Processing and monitoring test results.
   - Upload of data to database.


Data storage
DB
Test at the system level
Test results
Python - GUI and test manager
xml
XML file for upload to DB
csv
Summary of results
ROOT file
Progress update
Communication via named pipe
Test results
CPP
Ph2_ACF test procedure
Hybrid
Test at the hybrid level

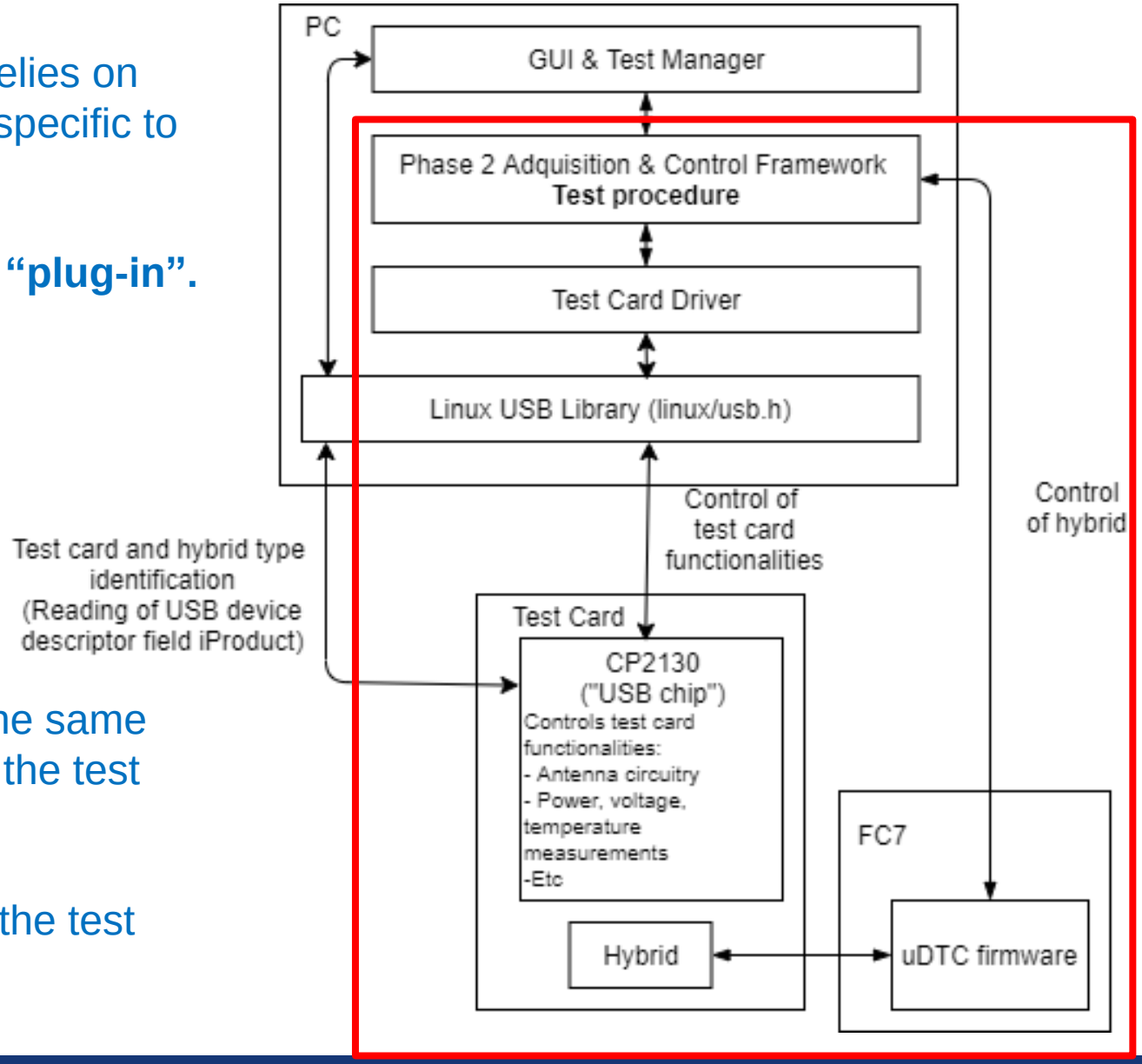

DB
Test results
Python - GUI and test manager
XML file for upload to DB
Summary of results
ROOT file
Progress update
Communication via named pipe
Test results
Ph2_ACF test procedure
Hybrid
Test at the hybrid level

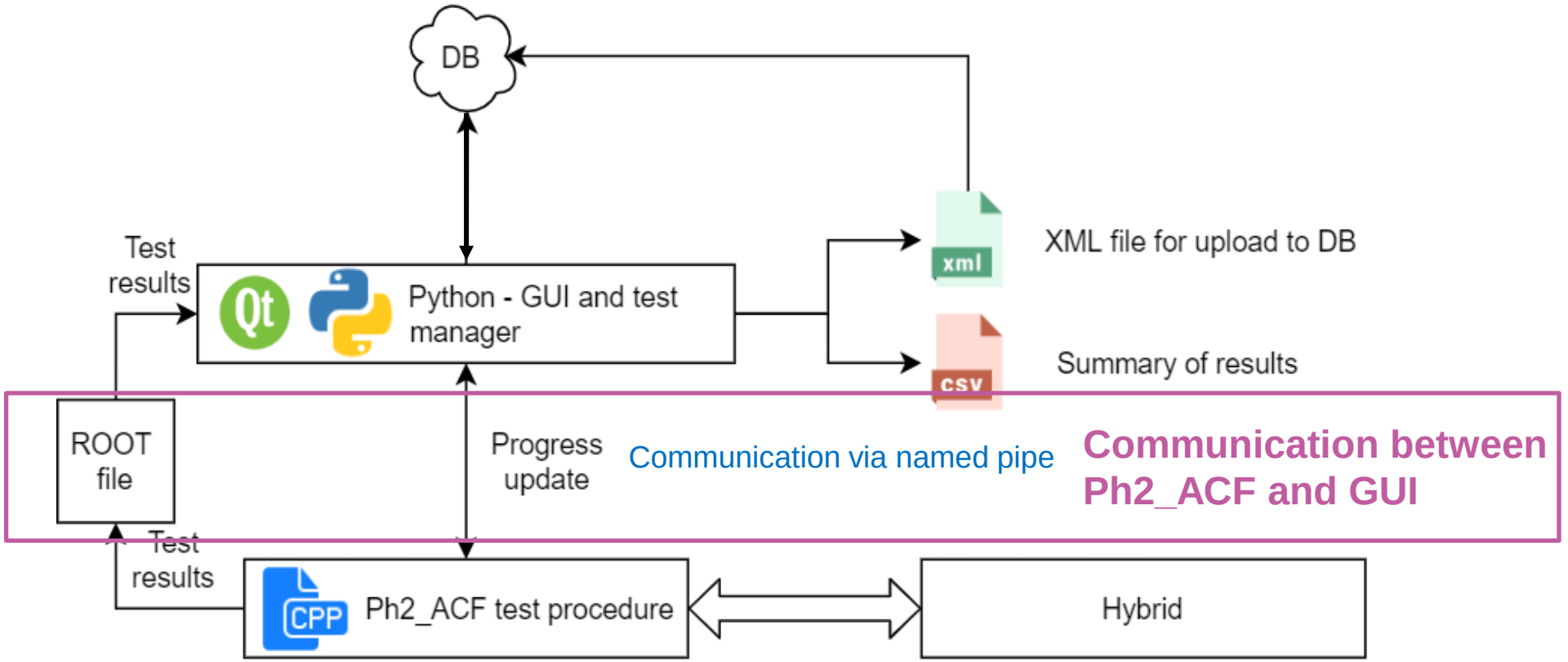
The testing at the hybrid level relies on three main pieces of software, specific to the hybrid type:

- Test procedure – **Ph2_ACF**.
- Test card driver – **Ph2_ACF "plug-in".**
- FC7 firmware.

All the test card designs share the same USB to SPI chip, which controls the test card features.

The chip is also used to identify the test card.

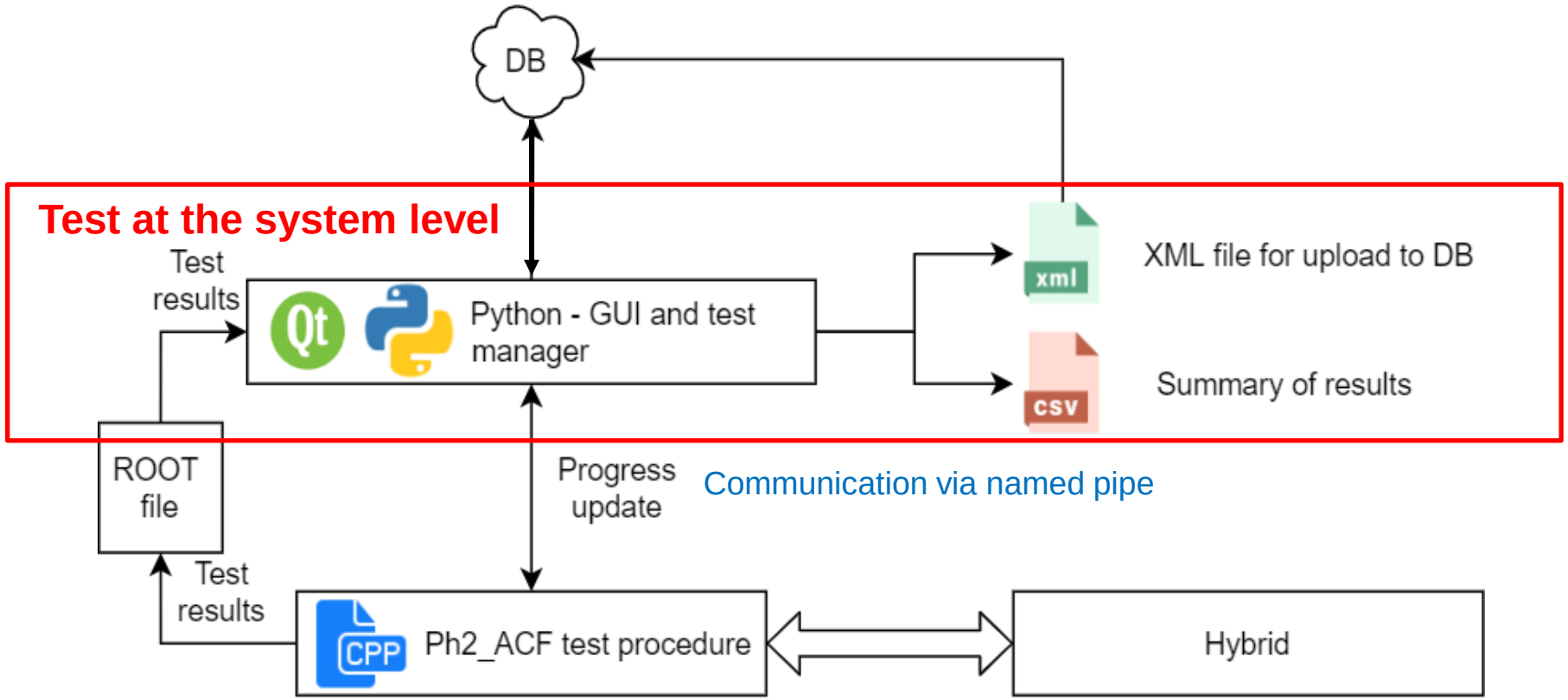



DB
Test results
Python - GUI and test manager
XML file for upload to DB
Summary of results
ROOT file
Progress update
Communication via named pipe
Communication between Ph2_ACF and GUI
Test results
Ph2_ACF test procedure
Hybrid

- Two parts:
  - During the run (progress report – **named pipe**):
    - The Ph2_ACF procedure writes to a named pipe that is read by the Python GUI.
    - Throught this named pipe, status messages and progress percentage are sent.

  An example can be found here: https://gitlab.cern.ch/alpapado/rundialog

  - After the run (results storage/communication – **ROOT file**):
    - All the results from the Ph2_ACF procedure are stored in the ROOT file.
    - The results to be used by the GUI are stored in a TTree that contains the most important measurements.
    - These results are processed by the GUI to define the hybrid status.


DB
Test at the system level
Test results
Python - GUI and test manager
xml
XML file for upload to DB
csv
Summary of results
ROOT file
Progress update
Communication via named pipe
Test results
CPP
Ph2_ACF test procedure
Hybrid

- Graphical tool, developed in Python.

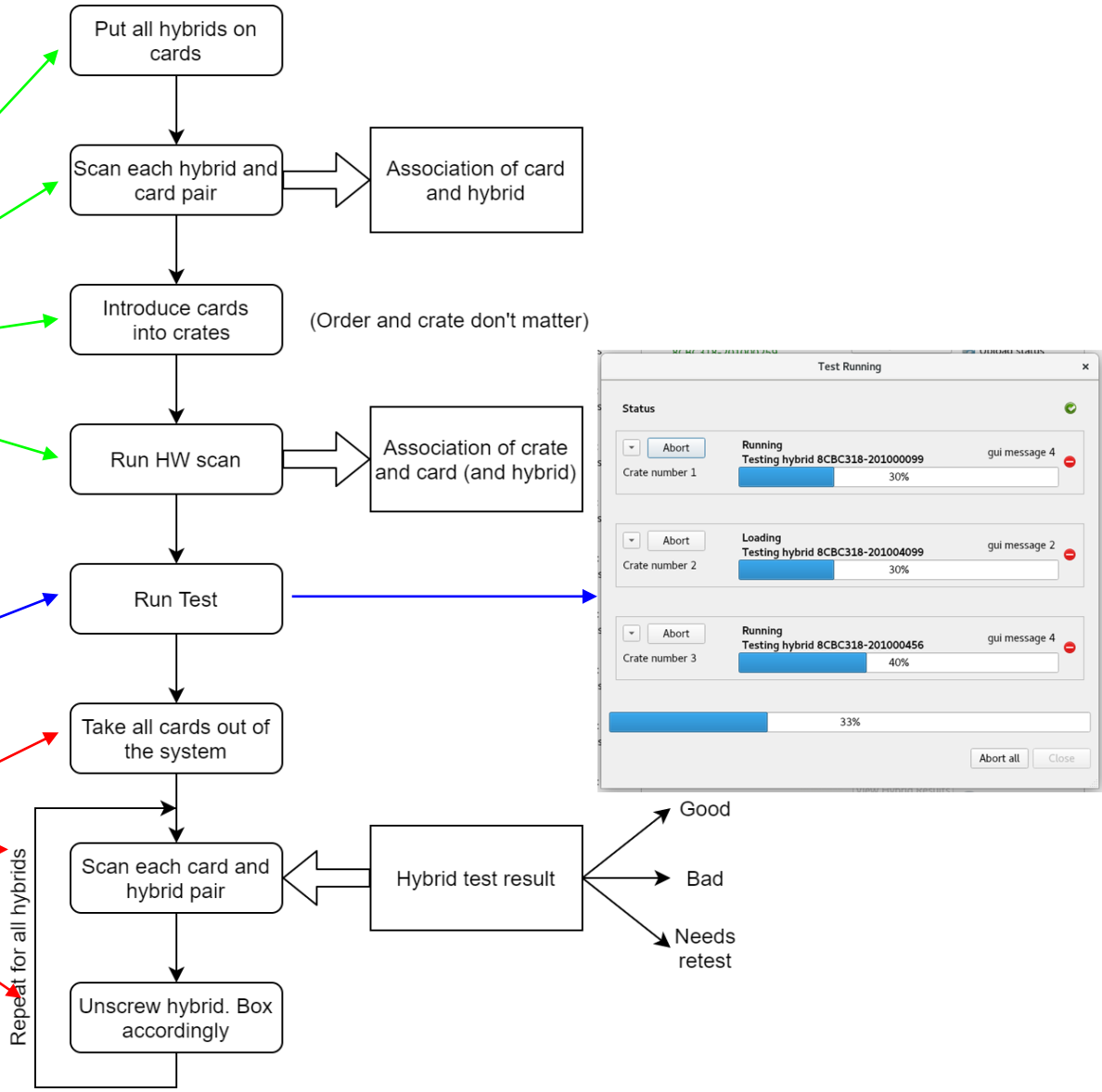
- Manages testing at the level of the system.
  - Manages connected hybrids and their positions in the crates.
  - Manages serialized testing in one crate/parallel testing across the crates.
  - Selects necessary firmware and test procedure.
  - Processes output of Ph2_ACF procedure and decide hybrid status.
  - Presents hybrid status to user and generates user-readable files with measurements.
  - Monitors results for detection of systematic errors.

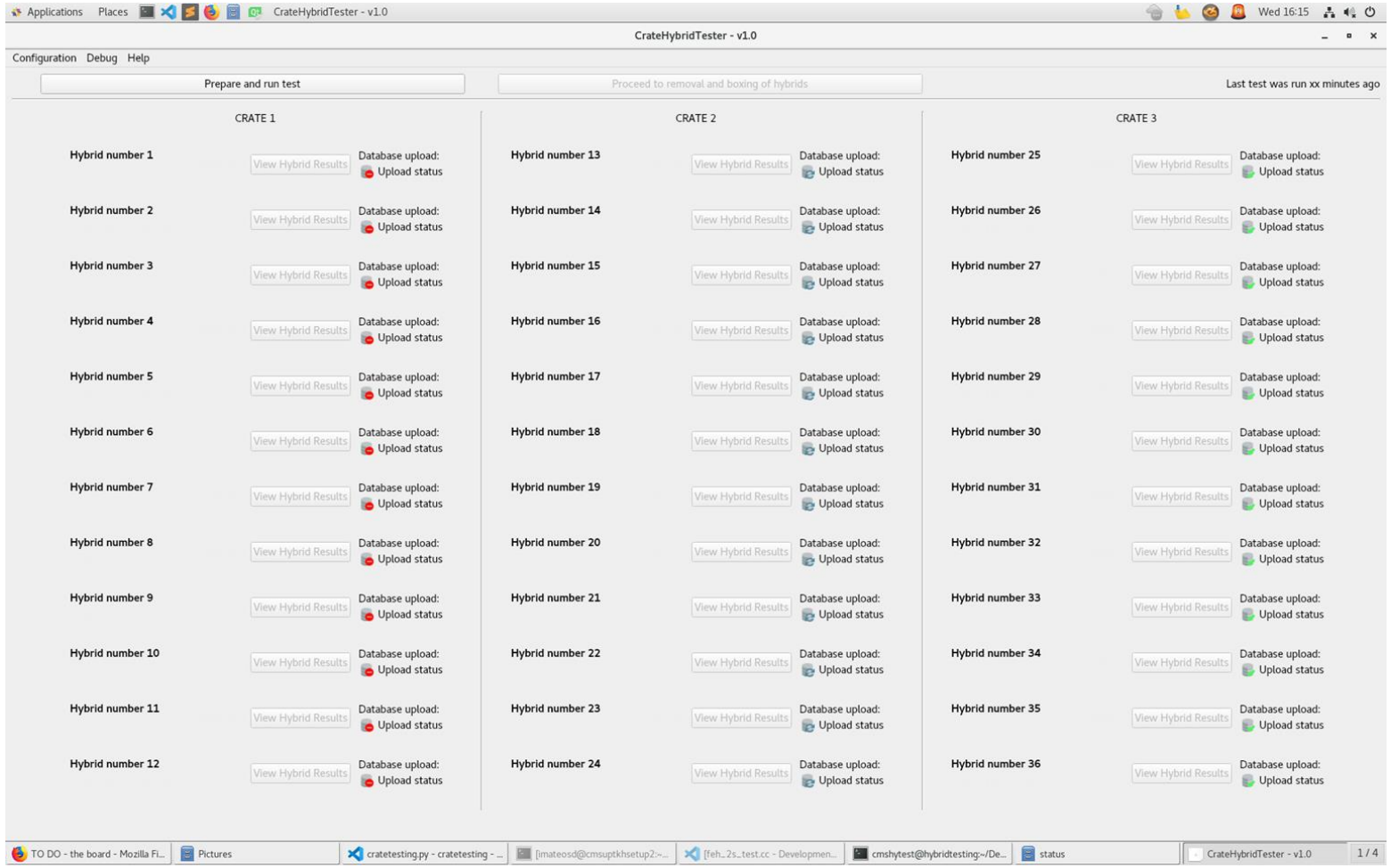
- Will store results in common database for general access.

TEST WORKFLOW

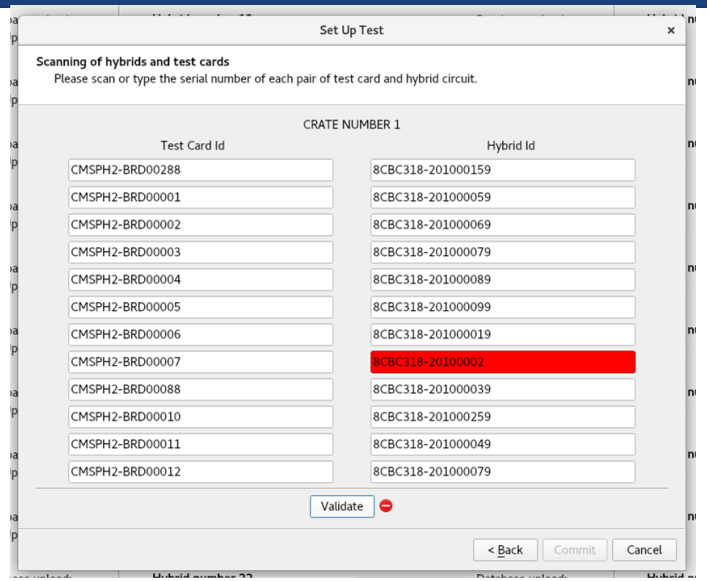
Parts of the GUI:

1. **Main window**: for results and launching of the test.
2. **Scanning wizard**: for the scanning of all the cards and hybrids. Walks the user through the process of inserting the hybrids on the system.
3. **Dialog of test run**: for monitoring the test.
4. **Dialog of hybrid removal**: for recovering the test results.

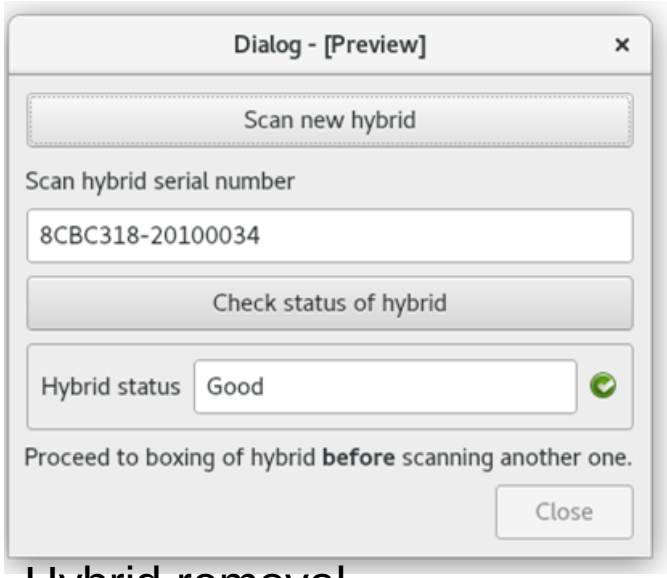




Main window

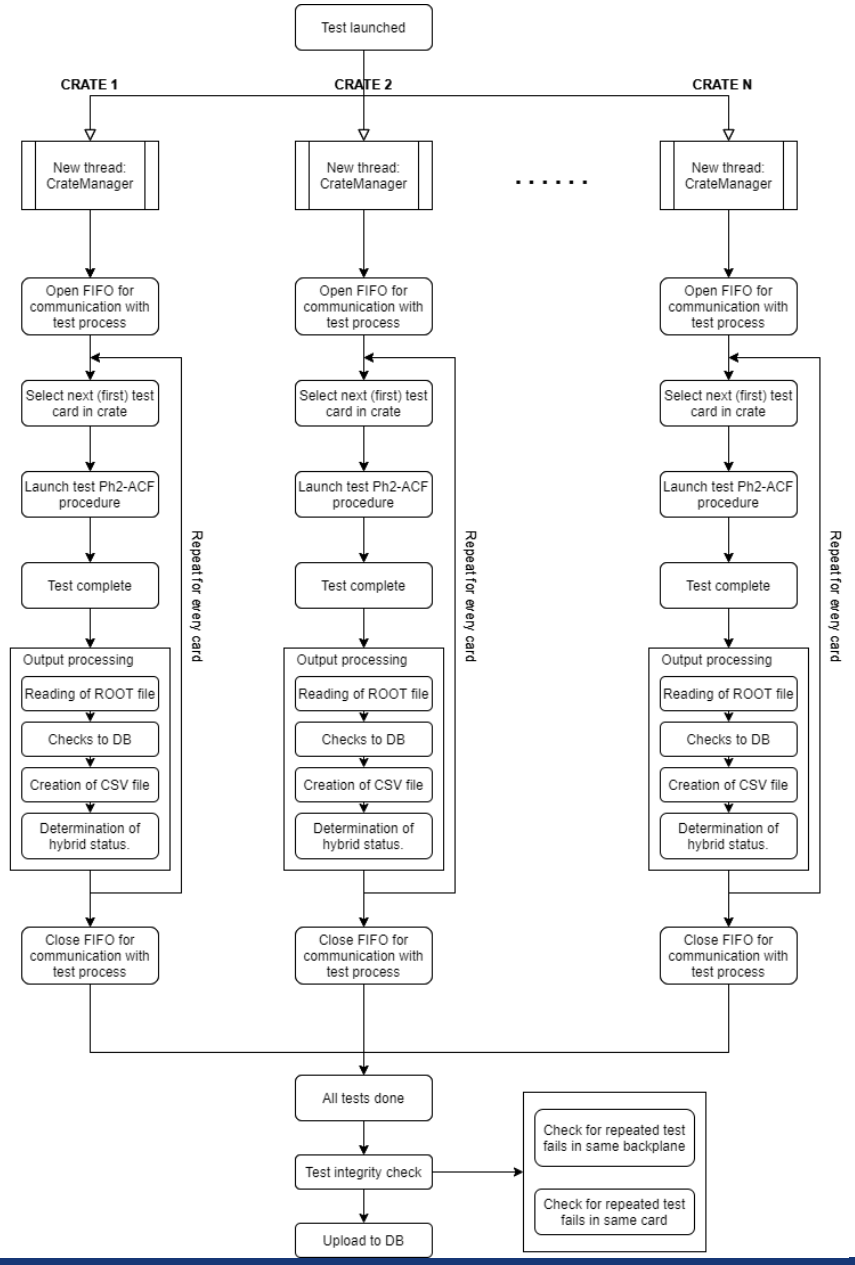


Scanning wizard



Hybrid removal

Structure of GUI for running tests on multiple crates.



- GUI based in Python.
  - Goal: to make the test system, the Ph2_ACF and the firmware images transparent to the user for all different hybrids.
  - Walks the user through the process of testing the hybrid.
  - Communicates with Ph2_ACF
    - via named pipe during run.
    - via ROOT file for the results.
  - Processes the output and decides the hybrid status based on the results.
  - Will upload all the data to the CMS Construction Database.

Thank you and enjoy the school!