DE LA RECHERCHE À L'INDUSTRIE

WeTest - Tests automation utility for EPICS

Novembre 5th 2019

Francis Gohier

Table of contents

Part I – User point of View

- Needs and History
- Principle
- Writing a scenario file
- Execution and GUI
- Test results Report
- DEMO

Part II – Behind the scene

- Implementation Architecture
- File reading and validation
- CLI and GUI communication
- Naming validation
- PDF Report generation
- Other changes planned
- WeTest development workflow
- Sharing WeTest

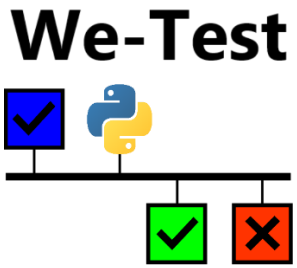
- Needs and History
- Principle
- Writing a scenario file
- Execution and GUI
- Test results report

Part I – User point of View

Needs and History

Main objective : Automate Acceptance Tests

- prepare tests beforehand
- test are executed automatically rather than manually
- therefore the same tests can be replayed over and over again
- automatically generate a proper test report



History

- In 2014 WeTest prototype written by J-F. Denis: scripts to execute tests described in CSV files and generate a PDF report

- In 2016, N. Senaud implements WeTest, using YAML files instead of CSV
- In 2017, Injector delivery at Catania (Italy) for ESS (European Spallation Source)

- Development and Maintenance handed to F. Gohier, in 2018
- In 2018 not used for a delivery because lacking features to test process

- Adding GUI (Graphical User Interface) in 2019, based on work by V. De Menezes
- Adding Macros support (variable substitution) in 2019
- Used in 2019 for ESS nBLM acceptance tests at Saclay (France)

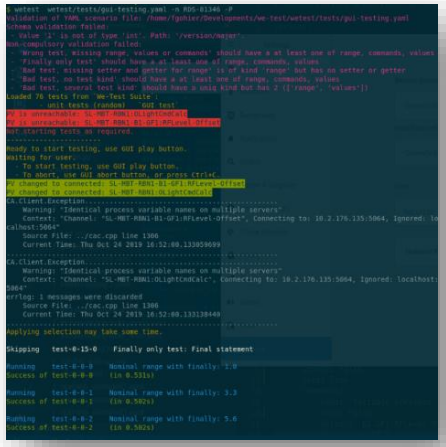
- Tests are being written for deliveries in 2020 for SARAF and IPHI projects

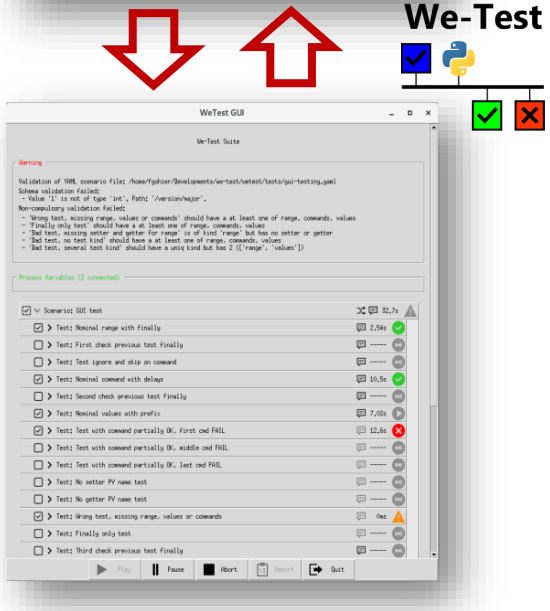
Principle: Overview

- Tests are described in a YAML file

```yaml
# This is a comment
version: [major: 1, minor: 1, bugfix: 0]
macros:
  - ENUM_VALUES: ["SWITCHING_OFF", "OFF", 1]
config:
    type: functional
    on_failure: continue
    name: Type test
    prefix: SL-MBT-RBN1
    delay: 2
tests:
  - name: "Nominal range with finally"
    delay: 0.5
    setter: :OLightCmdCalc
    getter: :OLightCmdCalc
    message: One line message !
    range:  { start: 1, stop: 10, step: 2.3}
    finally:
      setter: -B1-GF1:RFLevel-Offset
      value: 12.5
  - name: 'Nominal test enumerate'
    setter: :ADCS1-STAT-RB
    getter: :ADCS1-STAT-RB
    values: $(ENUM_VALUES)
    skip: True
    ignore: False
  - name: 'Nominal test waveform (char, int...)'
    setter: :UIncPwrMes-BPTR
    getter: :UIncPwrMes-BPTR
    commands:
      - name: "Char waveform as string"
        setter: :ADCS1-INFO
        getter: :ADCS1-INFO
        value: "abcd"
      - name: "Float waveform"
        setter: :UIncPwrMes-BPTR
        getter: :UIncPwrMes-BPTR
        value: [1, 2, 3.14, "inf", "-INF"]
      - name: "Int waveform"
        setter: :Test-Int-Waveform
        getter: :Test-Int-Waveform
        value: [1, 2, "A"]
      - name: "testing small delta and big margin"
        delta: 1
        margin: 5
        set_value: [95, 190]
        get_value: [100, 200]
```

- Executed in CLI with or without GUI





- a PDF report is generated

- YAML is a human-readable data-serialization language (close to JSON)

- GUI can be deactivated to work on computer without a graphical interface

- The PDF report is a clear, dated and readable account of the executed tests and their result

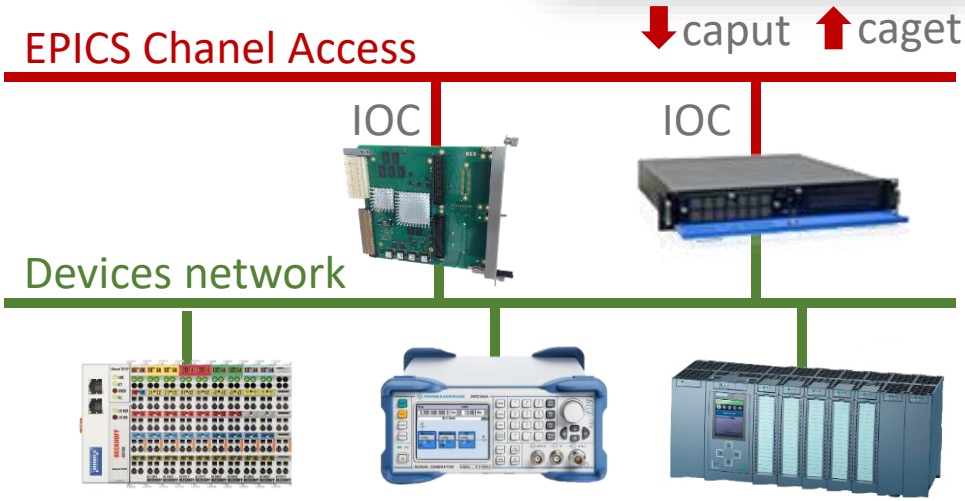
Principle: EPICS PV testing through CA and other vocabulary

- WeTest sets and checks the values of EPICS **Process Variables** (PVs) via the **Channel Access** (CA)

- WeTest executes **subtests** contained in a **suite**
- A suite is composed of one or several YAML file called **scenarios**
- **Tests** are defined in these scenarios, each test generates one or several **subtests**

- A subtest can **set** the value of a PV, **wait** for a given delay, **read** the value of a PV and **check** it against expected values

- Subtest result is one of : **skipped**, **success**, **failure** or **error**
- The subtest fails if it can not write the value or if the value read does not match the expected value
- A subtest errors when it is not written properly



Writing a scenario file : Example – Partical use

```yaml
macros:
    - GAS_RACK_TYPE:     "DISTRIBUTION"
      RACK:              "A"
      NBLM_NUMBER:       1
      GAS_LINE:          1
    - TEST_TITLE: "nBLM - set gas line ${GAS_LINE} for nBLM ${NBLM_NUMBER} on ${
      Sec-Sub:           "FEB-050Row"
      DIS:               "PBI"
      DELAY:             0.5 # sec, S7PLC delay is quite high in IOC config
      # gas config
      FLOW:                          1     # L/h
      # interlocks
      REGULATION_DELAY:              1     # seconds (should be more than DELAY_S
      DELAY_SHUNT_INTERLOCK:         0     # min
      ## pressure
      LOW_PRESSURE_BYPASS:           0     # bypass is disabled
      LOW_PRESSURE_THRESHOLD:        1000  # mBar
      LOW_PRESSURE_TIMER:            10    # sec

config:
   type:        functional
   on_failure: pause
   name:        ${TEST_TITLE}
   prefix:      ""
   delay:       ${DELAY}

tests:
   # check connection
   - name: "check connection status with gas PLC (situated in the main rack)"
     getter: "${Sec-Sub}:${DIS}-PLC-Line${GAS_LINE}:ReadyR.SEVR"
     values: [0]


   # stop regulation
   - name: "stop regulation"
     setter: "${Sec-Sub}:${DIS}-PLC-Line${GAS_LINE}:StartReg"
     getter: "${Sec-Sub}:${DIS}-PLC-Line${GAS_LINE}:StartRegR"
     values: [0]

   # check hardware faults
   - name: "check hardware connection (sensors and valves)"
     commands:
       - name : "pressure sensor"
         getter: "${Sec-Sub}:${DIS}-PLC-Line${GAS_LINE}:PrsHardErrR"
         value: 0
       - name: "manual/auto mismatch valve"
         getter: "${Sec-Sub}:${DIS}-MV-$(RACK)$(GAS_LINE)0:MismatchErrR"
         value: 0
```

- One of the scenario written to test a gas line on ESS nBLM project

```yaml
     - name: "flow control valve"
       getter: "${Sec-Sub}:${DIS}-FCV-$(RACK)$(GAS_LINE)0:HardErrR"
       value: 0
     - name: "input flow sensor"
       getter: "${Sec-Sub}:${DIS}-FT-$(RACK)$(GAS_LINE)0:HardErrR"
       value: 0
     - name: "output flow sensor"
       getter: "${Sec-Sub}:${DIS}-FT-$(RACK)$(GAS_LINE)9:HardErrR"
       value: 0

# ack all warning/interlokcs
- name: "acknowledge all interlocks"
  setter: "${Sec-Sub}:${DIS}-PLC-Line$(GAS_LINE):Ack"
  getter: "${Sec-Sub}:${DIS}-PLC-Line$(GAS_LINE):AckR"
  values: [1] # ack all interlocks

# set flow
- name: "set flow to ${FLOW} L/h"
  setter: "${Sec-Sub}:${DIS}-PLC-Line${GAS_LINE}:FlwSP"
  getter: "${Sec-Sub}:${DIS}-PLC-Line${GAS_LINE}:FlwSPR"
  values:
    - ${FLOW}

# set interlocks
## pressure
### low (warning)
- name: "set interlocks - low pressure"
  commands:
    - name:   "bypass"
      setter: "${Sec-Sub}:${DIS}-PLC-Line${GAS_LINE}:PrsLoErrBypass"
      getter: "${Sec-Sub}:${DIS}-PLC-Line${GAS_LINE}:PrsLoErrBypassR"
      value:  ${LOW_PRESSURE_BYPASS}
    - name:   "threshold - ${LOW_PRESSURE_THRESHOLD} mBar"
      setter: "${Sec-Sub}:${DIS}-PLC-Line${GAS_LINE}:PrsLoErrTrsh"
      getter: "${Sec-Sub}:${DIS}-PLC-Line${GAS_LINE}:PrsLoErrTrshR"
      value:  ${LOW_PRESSURE_THRESHOLD}
    - name:   "timer - ${LOW_PRESSURE_TIMER} sec"
      setter: "${Sec-Sub}:${DIS}-PLC-Line${GAS_LINE}:PrsLoErrTim"
      getter: "${Sec-Sub}:${DIS}-PLC-Line${GAS_LINE}:PrsLoErrTimR"
      value:  ${LOW_PRESSURE_TIMER}
### high pressure (warning)
### very high pressure (interlock)

## flow
## mismacth
## gap flow
## manual/auto
```

- Another scenario calls it multiple time to substitute the different gas lines identifiers

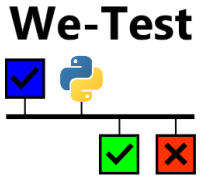
Writing a scenario file : YAML syntax

```yaml
config:
    type: functional
    on_failure: continue
    name: Type test
    prefix: SL-MBT-RBN1
    delay: 2

tests:

    - name: "Nominal range with finally"
      delay: 0.5
      setter: :OLightCmdCalc
      getter: :OLightCmdCalc
      message: One line message !
      range:  { start: 1, stop: 10, step: 2.3}
      finally:
        setter: -B1-GF1:RFLevel-Offset
        value: 12.5

    - name: 'Nominal test enumerate'
      setter: :ADCS1-STAT-RB
      getter: :ADCS1-STAT-RB
      values: ${ENUM_VALUES}
      skip: True
      ignore: False

    - name: 'Nominal test waveform (char, int...)'
      setter: :UIncPwrMes-BPTR
      getter: :UIncPwrMes-BPTR
      commands:
        - name: "Char waveform as string"
          setter: :ADCS1-INFO
          getter: :ADCS1-INFO
          value: "abcd"
        - name: "Float waveform"
          setter: :UIncPwrMes-BPTR
          getter: :UIncPwrMes-BPTR
          value: [1, 2, 3.14, "inf", "-INF"]
        - name: "Int waveform"
          setter: :Test-Int-Waveform
          getter: :Test-Int-Waveform
          value: [1, 2, "A"]
        - name: "testing small delta and big margin"
          delta: 1
          margin: 5
          set_value: [95, 190]
          get_value: [100, 200]
```

- Comments start with a hashtag (#)
- Uses indentation for multiline content
- Composed of **maps** and **sequences** (aka. **dictionaries** and **lists**)
- Lists are composed of items starting with a dash (-)
- Maps are composed of items with a key and a value separated by a colon (:)
- Needs a space after dash and colon
- One-line form is available using:
  - square brackets ([]) for list
  - curly brackets ({}) for maps
  - comas (,) to separate items
- String are found automatically or using quotes (“”)
- Some strings are read as Booleans: True, False, Yes, No

Writing a scenario file : Tests block

```yaml
tests:

  - name: "Nominal range with finally"
    delay: 0.5
    setter: :OLightCmdCalc
    getter: :OLightCmdCalc
    message: One line message !
    range:  { start: 1, stop: 10, step: 2.3}
    finally:
      setter: -B1-GF1:RFLevel-Offset
      value: 12.5

  - name: 'Nominal test enumerate'
    setter: :ADCS1-STAT-RB
    getter: :ADCS1-STAT-RB
    values: ${ENUM_VALUES}
    skip: True
    ignore: False

  - name: 'Nominal test waveform (char, int...)'
    setter: :UIncPwrMes-BPTR
    getter: :UIncPwrMes-BPTR
    commands:
      - name: "Char waveform as string"
        setter: :ADCS1-INFO
        getter: :ADCS1-INFO
        value: "abcd"
      - name: "Float waveform"
        setter: :UIncPwrMes-BPTR
        getter: :UIncPwrMes-BPTR
        value: [1, 2, 3.14, "inf", "-INF"]
      - name: "Int waveform"
        setter: :Test-Int-Waveform
        getter: :Test-Int-Waveform
        value: [1, 2, "A"]
      - name: "testing small delta and big margin"
        delta: 1
        margin: 5
        set_value: [95, 190]
        get_value: [100, 200]
```



- **tests** block lists all the tests to execute
  each **test** generates one or more **subtests**
- A test has a **test kind** (range, values or commands)
  and a setter and/or getter PV (EPICS Process Variable)
- **Some fields are common to all tests kind**
  name, message, prefix, use_prefix, delay, setter, getter, margin, delta, ignore, skip, on_failure, retry
- **range: to test many numerical values**
  start, stop, step, lin, geom, includes_start, include_stop, sort
- **values: hand-picked values and not only numerical**
  list of values (float, int, bool, enum, string or waveform)
- **commands: set all the parameters of each subtest**
  name, message, delay, setter, getter, set_value, get_value, value, ignore, skip, on_failure and retry
- **finally: sets a value after at the end of the test**

Writing a scenario file : Example - Use Cases

tests hardware range of values

```yaml
- name: "Testing power supply whole range"
  delay: 0.5
  prefix: "-POWER-SUPPLY-01"
  setter: ":VSet"
  getter: ":VMes"
  range:  { start: 1, stop: 1000, geom: 10}
  delta: 2
  margin: 1
  message: |
  tests values 1, 2.15, 4.64, 10, 21, 46.41, 100, 215.44, 464.14 and 1000
  accepts +/-2 and +/-1%
```

tests min a average computation off a waveform

```yaml
- name: "Testing waveform statistics computation"
  prefix: "-RF-CH1"
  commands:
    - name: "Send known data"
      setter: ":VMes"
      set_value: [1, 2, 4.5, 12, 3.6, 48, 2.32, 15, 23, -16, 0, 0, 0]
    - name: "Check average"
      getter: ":AvgMes"
      get_value: 9.54
      delta: 0.01
    - name: "Check min"
      getter: ":MinMes"
      get_value: -16
```

wait to reach given condition

```yaml
- name: "Wait for temperature"
  message: stuck here until temperature of ${TEMP}°C is reached
  delay: 2
  prefix: "-PT042"
  getter: ":TempMes"
  values:
    - ${TEMP}
  retry: -1 # same as INF
  on_failure: continue
```

test multiple enum values

```yaml
- name: "Check conditionning start"
  delay: 1
  prefix: ":COND"
  setter: "-Start"
  getter: "-Status"
  values: [START, PAUSE, STOP, START, STOP]
```

test controller refusal

```yaml
- name: "Try value out of possible range"
  delay: 0.5
  prefix: "-POWER-SUPPLY-01"
  setter: ":VSet"
  getter: ":VMes"
  commands:
    - name: "below zero stay at zero"
      set_value: -12
      get_value: 0
    - name: "above 1000 stay at 1000"
      set_value: 2000
      get_value: 1000
```

Writing a scenario file : Config block and Version

```yaml
# This is a comment

version: {major: 1, minor: 1, bugfix: 0}

macros:
  - ENUM_VALUES: ["SWITCHING_OFF", "OFF", 1]

config:
    type: functional
    on_failure: continue
    name: Type test
    prefix: SL-MBT-RBN1
    delay: 2

tests:

    - name: "Nominal range with finally"
      delay: 0.5
      setter: :OLightCmdCalc
      getter: :OLightCmdCalc
      message: One line message !
      range:  { start: 1, stop: 10, step: 2.3}
      finally:
        setter: -B1-GF1:RFLevel-Offset
        value: 12.5
```

- **config** block configures the scenario's tests block
- **name** in the config block corresponds to the scenario name
- **type** determines if the scenario's tests are "functional" or "unit".
  - Unit tests are shuffled to be executed in random order (yet the subtests stay in the same order within each test)
- **prefix** will be prepended to each test's prefix
- other fields define the default fallback value to use when these fields are not defined in a test.
  - use_prefix, delay, ignore, skip, on_failure and retry

- **version**, at the root of the file, is mandatory and used to check that the file is compatible with the version WeTest reading it.

Writing a scenario file : Include block and suite name

```yaml
include:
    - /path/to/scenario_1
    - relative/path/to/scenario_2
    - tests
    - scenario_3
    - [scenario_macro, SKIP: no, MAX_VAL: 12]
    - path: scenario_macro
      skip: Yes
      MAX_VAL: ${MACRO_FLOAT}
      SCENARIO_TITLE: Title from include
```

```yaml
include:
    - start_regulation.yaml
    - trig_interlock_pressure.yaml
    - start_regulation.yaml
    - trig_interlock_flow.yaml
```

- **include block enables to insert other scenarios before or after the tests defined in this file**
- **Included items must specify a file name or path**
  filepath resolution order is: absolute path, then relative to execution location, then relative to current scenario location
- **Tests position in the include sequence is by default at the end, but can be changed by using the "tests" include keyword**
- **A scenario file can have no tests and config block and only an include block (as well as a version).**
- **macros can be changed in the include line**
  (see next slide for more information about macros)

- **name, at the root of the file, is used as the suite name.**
  Only the name defined in the topmost file (the one that is first read by WeTest and includes the other scenarios) is used, the other ones are ignored. This field is optional.

Writing a scenario file : Macros block

```yaml
macros:
# here as a sequence,
# but could also be directly a dictionary
    - MACRO_FOR_MACRO: final value
      SCENARIO_TITLE:  "Testing macros"
      MACRO_REDEFINED: "used value"
    - VERSION_MIN: 1
      MACRO_FLOAT: 1.06
      MACRO_INT: 6
      MACRO_STR_1: a string macro
      MACRO_STR_2: another string macro
      MACRO_REDEFINED: "ignored value"
      MACRO_USING_MACRO: ${MACRO_FOR_MACRO}
      SKIP_TEST_STR: no
      MACRO_DICT: {start: 2, stop: 12, step: 2}
      MACRO_LIST: [2, 4, 6, 8]
      MACRO_TEST:
        name: "command test as a macro"
        setter: ":OLightCmdCalc"
        getter: ":OLightCmdCalc"
        skip: $(skip)
        commands:
          - name: "command name"
            value: 2
      WAVEFORM_VALUES:
        - [10, 11, 12]
        - [13, 14, 15]
        - [16, 17, 18]
```

- **macros** block enables to define a kind of variable with a name (string) and a value
- macro value can be a simple type by also maps and lists

- macros are set in the macros block
- macros can also be set in the **include** block
  - and have priority over macros defined in the included file

- The value of a macro is substituted anywhere in the file where **$(MACRO_NAME)** or **${MACRO_NAME}** is used
- macros can be used in the macros block
  - but should be defined before hand (previous item in the macros block as a list, or from include call)

- Limitations
  - dash (-) and colon (:) in a string value, are in some specific cases interpreted respectively as a list or as a map
  - it is not possible to force a value to be interpreted as a string (however one can use the macro in a string with a space for instance)

Writing a scenario file : Generic scenarios

```yaml
# unit test for vertical test at Saclay

macros:
  - area:        "CEA"
    device:      "PBI-nBLM"
    DELAY:       0.2   # sec
    CRATE_IDX:   "1"
    HV_SLOT:     "02"
    LV_SLOT:     "10"

name: "nBLM - vertical integration tests - unit tests (range and limits of all d

include:

    # gas
    # main rack
    # line 1
    - path:           '/home/ceauser/e3-3.15.5/e3-nblmioc/m-epics-nblm/misc/wete
      GAS_LINE:       1
      RACK:           "A"
      TEST_TITLE:     "gas - MEBT-DTL-1 line"
      Sec-Sub:        "FEB-050Row"
      DELAY:          0.5 # sec, S7PLC delay is quite high in IOC config
    # line 2
    # line 3

    # Low voltage
    ## slot 10
    ### channel 0
    - path:           'wetest/generic/weTest_unit_generic_power_supply.yaml'
      TEST_TITLE:     "LV (+8V) - crate ${CRATE_IDX}, slot ${LV_SLOT}, channel 0
      Sec-Sub:        "${area}"
      Dis-Dev-Idx:    "${device}"
      DELAY:          ${DELAY}
      # voltage
      IGNORE_VOLTAGE:   False
      SET_VOLTAGE:    "C${CRATE_IDX}-S${LV_SLOT}-CH000-V0Set"
      GET_VOLTAGE:    "C${CRATE_IDX}-S${LV_SLOT}-CH000-V0Set-RB"
      MAX_VOLTAGE:    15  # Volts
      MIN_VOLTAGE:    5   # Volts
      STEP_VOLTAGE:   3   # Volts
      OVER_VOLTAGE:   16  # Volts
      UNDER_VOLTAGE:  4   # Volts
      FINAL_VOLTAGE:  0   # Volts
      # current
      IGNORE_CURRENT:   False
      SET_CURRENT:    "C${CRATE_IDX}-S${LV_SLOT}-CH000-I0Set"
      GET_CURRENT:    "C${CRATE_IDX}-S${LV_SLOT}-CH000-I0Set-RB"
      MAX_CURRENT:    5   # Amp
```

- **Generic files** are shipped with WeTest for instance to test a power supply

```yaml
      MIN_CURRENT:    0   # Amp
      STEP_CURRENT:   1   # Amp
      OVER_CURRENT:   6   # Amp
      UNDER_CURRENT:  -1  # Amp
      FINAL_CURRENT:  0   # Amp
      # on/off
      IGNORE_ON_OFF:    False
      SET_ON_OFF:     "C${CRATE_IDX}-S${LV_SLOT}-CH000-Pw"
      GET_ON_OFF:     "C${CRATE_IDX}-S${LV_SLOT}-CH000-Pw-RB"
      FINAL_ON_OFF:   0 # off
      # ramp-up
      IGNORE_RAMP_UP:   False
      SET_RAMP_UP:    "C${CRATE_IDX}-S${LV_SLOT}-CH000-RUpTime"
      GET_RAMP_UP:    "C${CRATE_IDX}-S${LV_SLOT}-CH000-RUpTime-RB"
      MAX_RAMP_UP:    200   # msec
      MIN_RAMP_UP:    1     # msec
      STEP_RAMP_UP:   50    # msec
      OVER_RAMP_UP:   201   # msec
      UNDER_RAMP_UP:  -1
      MARGIN_RAMP_UP: 1     # %
      FINAL_RAMP_UP:  200   # msec
      # ramp-down
      IGNORE_RAMP_DOWN:   False
      SET_RAMP_DOWN:    "C${CRATE_IDX}-S${LV_SLOT}-CH000-RDwTime"
      GET_RAMP_DOWN:    "C${CRATE_IDX}-S${LV_SLOT}-CH000-RDwTime-RB"
      MAX_RAMP_DOWN:    200   # msec
      MIN_RAMP_DOWN:    1     # msec
      STEP_RAMP_DOWN:   50    # msec
      OVER_RAMP_DOWN:   201   # msec
      UNDER_RAMP_DOWN:  -1
      MARGIN_RAMP_DOWN: 1   # %
      FINAL_RAMP_DOWN:  200 # msec
### channel 1
    - path:           'wetest/generic/weTest_unit_generic_power_supply.yaml'
      TEST_TITLE:     "LV (-8V) - crate ${CRATE_IDX}, slot ${LV_SLOT}, channel 0
      Sec-Sub:        "${area}"
      Dis-Dev-Idx:    "${device}"
      DELAY:          ${DELAY}
      # voltage
      IGNORE_VOLTAGE:   False
      SET_VOLTAGE:    "C${CRATE_IDX}-S${LV_SLOT}-CH001-V0Set"
      GET_VOLTAGE:    "C${CRATE_IDX}-S${LV_SLOT}-CH001-V0Set-RB"
      MAX_VOLTAGE:    15  # Volts
      MIN_VOLTAGE:    5   # Volts
      STEP_VOLTAGE:   3   # Volts
      OVER_VOLTAGE:   16  # Volts
      UNDER_VOLTAGE:  4   # Volts
      FINAL_VOLTAGE:  0   # Volts
```

- Here the generic file is included for each of the twenty lines of ESS nBLM, after an initial setup

Execution and GUI: CLI options

- **Currently WeTest can only be run from CLI**
- **Options relative to tests definition**
  - one or multiple YAML scenario files can be given using positionnal arguments or -s, --scenario option
  - -m, --macros to change macro value
- **Options relative to PV**
  - by default PVs list is created from subtests list
  - -d, --db option to enter EPICS DB directory or files
  - -n, --naming to choose naming rules validation (ESS, RDS-81346, SARAF, None)
  - -D, --no-pvs to deactivate PV monitoring
- **Options relative to report**
  - -O, --no-pdf-output to deactivate report creation
  - -o, --pdf-output to change output report name (default to wetest-results.pdf)
- **Other options**
  - -G, --no-gui to show execution and results only in CLI
  - -p, --force-play to start testing even if some PVs are deconnected
  - -P, --no-auto-play to wait for user input before running tests even if all PVs are connected
- **Examples**

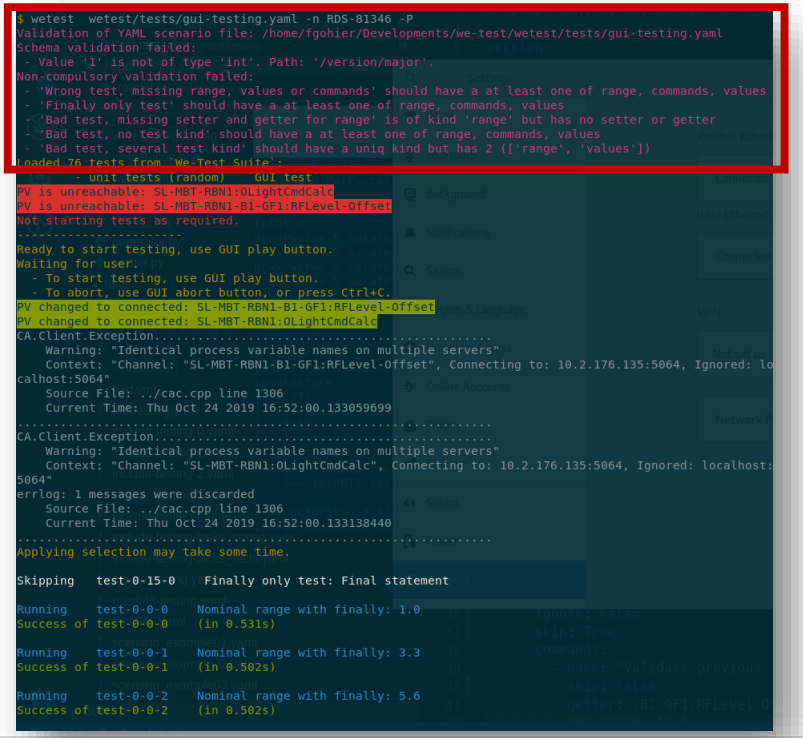
```
wetest -h
wetest ./path/to/scenario
wetest -d /path/to/DB -n SARAF -ps ./path/to/scenario -o report_name
wetest ./path/to/scenario -DPGO --macros SKIP=True SCENARIO_TITLE="Title from CLI"
```

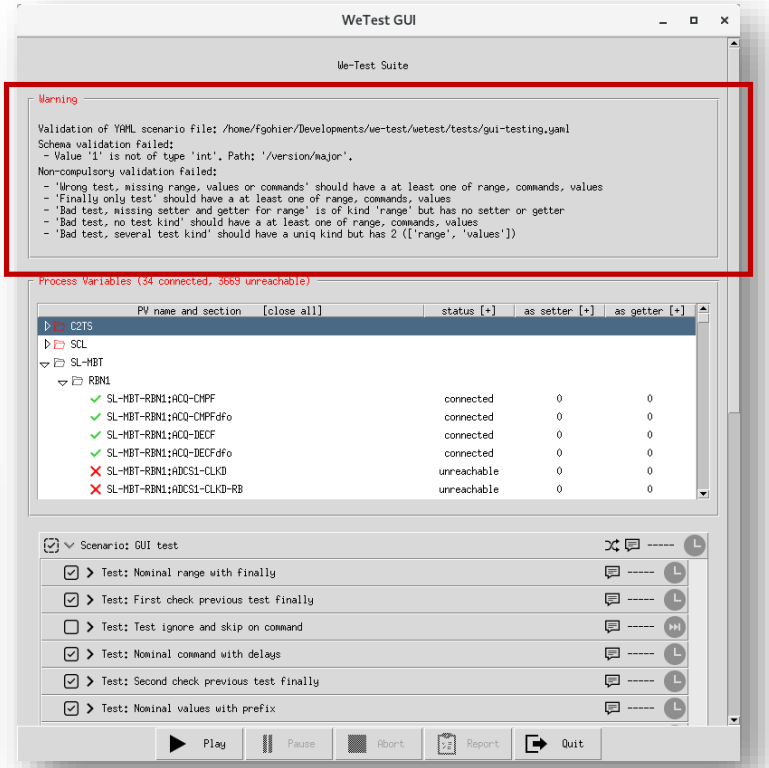
```
$ wetest -h
usage: wetest [-h] [-s TEST_FILE [TEST_FILE ...]]
              [-m MACRO=VALUE [MACRO=VALUE ...]] [-d DB_PATH [DB_PATH ...] |
              -D] [-n {ESS,RDS-81346,SARAF,None}] [-G] [-p | -P]
              [-o OUTPUT_FILE | -O]
              [TEST_FILE [TEST_FILE ...]]

WeTest is a testing facility for EPICS modules. Tests are described in a YAML
file, and executed over the Channel Access via Pyepics library. It also
enables to monitor PVs (extracted from the tests and from specified DB).

positional arguments:
  TEST_FILE             One or several scenario files (executed before
                        scenarios from --scenario).

optional arguments:
  -h, --help            show this help message and exit
  -s TEST_FILE [TEST_FILE ...], --scenario TEST_FILE [TEST_FILE ...]
                        One or several scenario files (executed after
                        positional arguments).
  -m MACRO=VALUE [MACRO=VALUE ...], --macros MACRO=VALUE [MACRO=VALUE ...]
                        Override macros defined in file.
  -d DB_PATH [DB_PATH ...], --db DB_PATH [DB_PATH ...]
                        EPICS DB files and directory to extract additional PVs
                        from.
  -D, --no-pv           Run withtout monitoring any PVs.
  -n {ESS,RDS-81346,SARAF,None}, --naming {ESS,RDS-81346,SARAF,None}
                        Specifies naming convention to display PV name
                        (defaults to None).
  -G, --no-gui          Do not open a GUI.
  -p, --force-play      Start runnning tests automatically even with
                        disconnected PV.
  -P, --no-auto-play    Tests will not start running automatically.
  -o OUTPUT_FILE, --pdf-output OUTPUT_FILE
                        Specify PDF output file name (otherwise defaults to
                        wetest-results.pdf).
  -O, --no-pdf-output   Do not generate the PDF report with tests results.
```

Execution and GUI: Warning section

Warning section

- shows only when scenario files contains errors
- collapsable by clicking on warning outer border
- first check for basic schema validation
  (fields names and type)
- second check for higher level validation
  (complex types or relations between fields)
- third is no-go validation
  (WeTest can not work with these inputs)





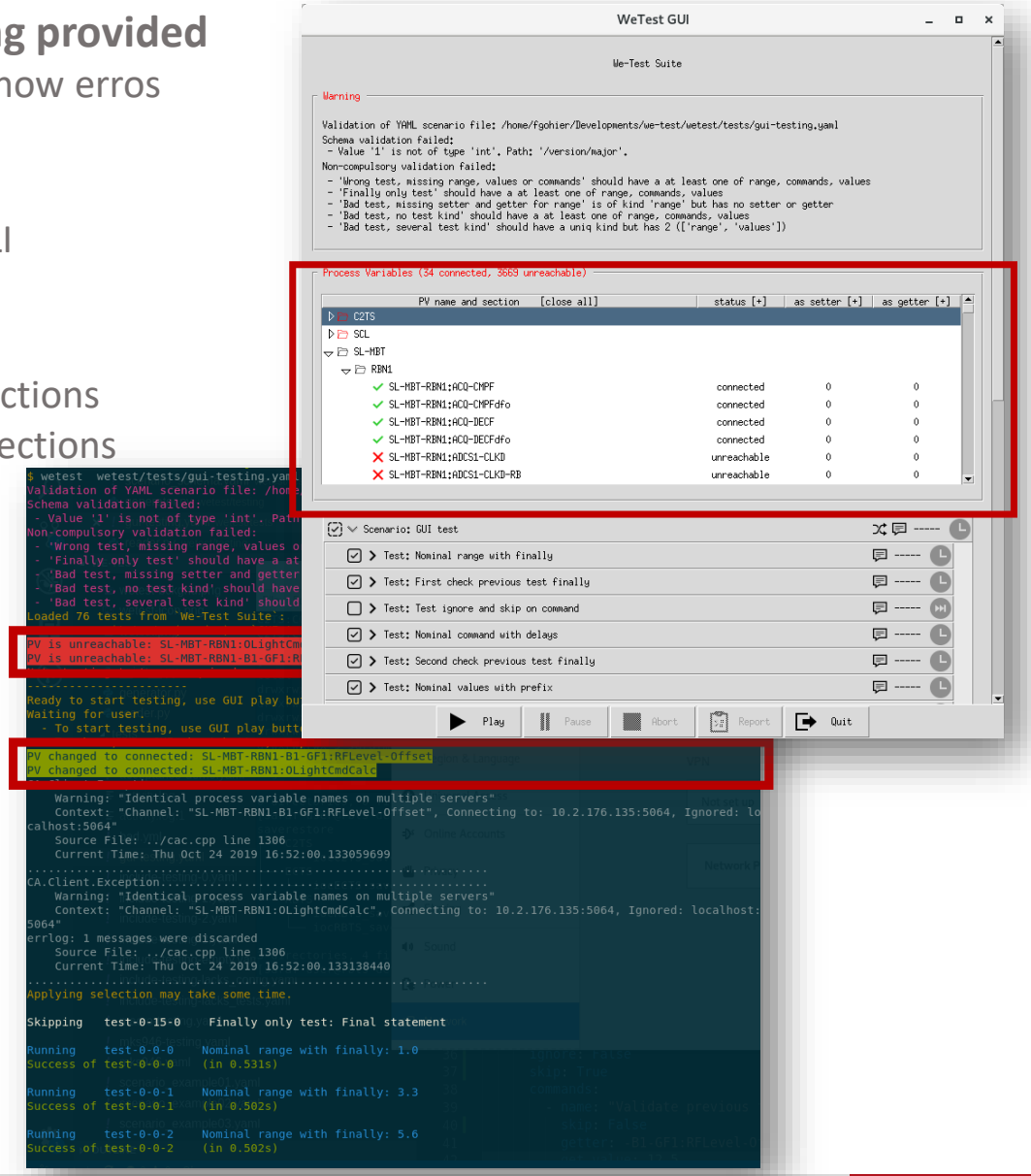
Execution and GUI: PVs checking and monitoring

- **PVs are checked against to the naming provided**
  an invalid naming section is created to show erros

- **PVs connection status is monitored**
  much more convenient in GUI than in CLI

- **Additionnal features available in GUI**
  - move around and collapse/expand sections
  - click first column title to collapse all sections
  - click other column title to apply filter
  - filter to show only disconnected
  - filter to show only tested PVs

- **Two display modes**
  - short mode: table only appears when there are disconnected PVs
  - full mode: table shows in anycase and uses three-time more space
  - toogle mode by clicking on Process Variables outer border



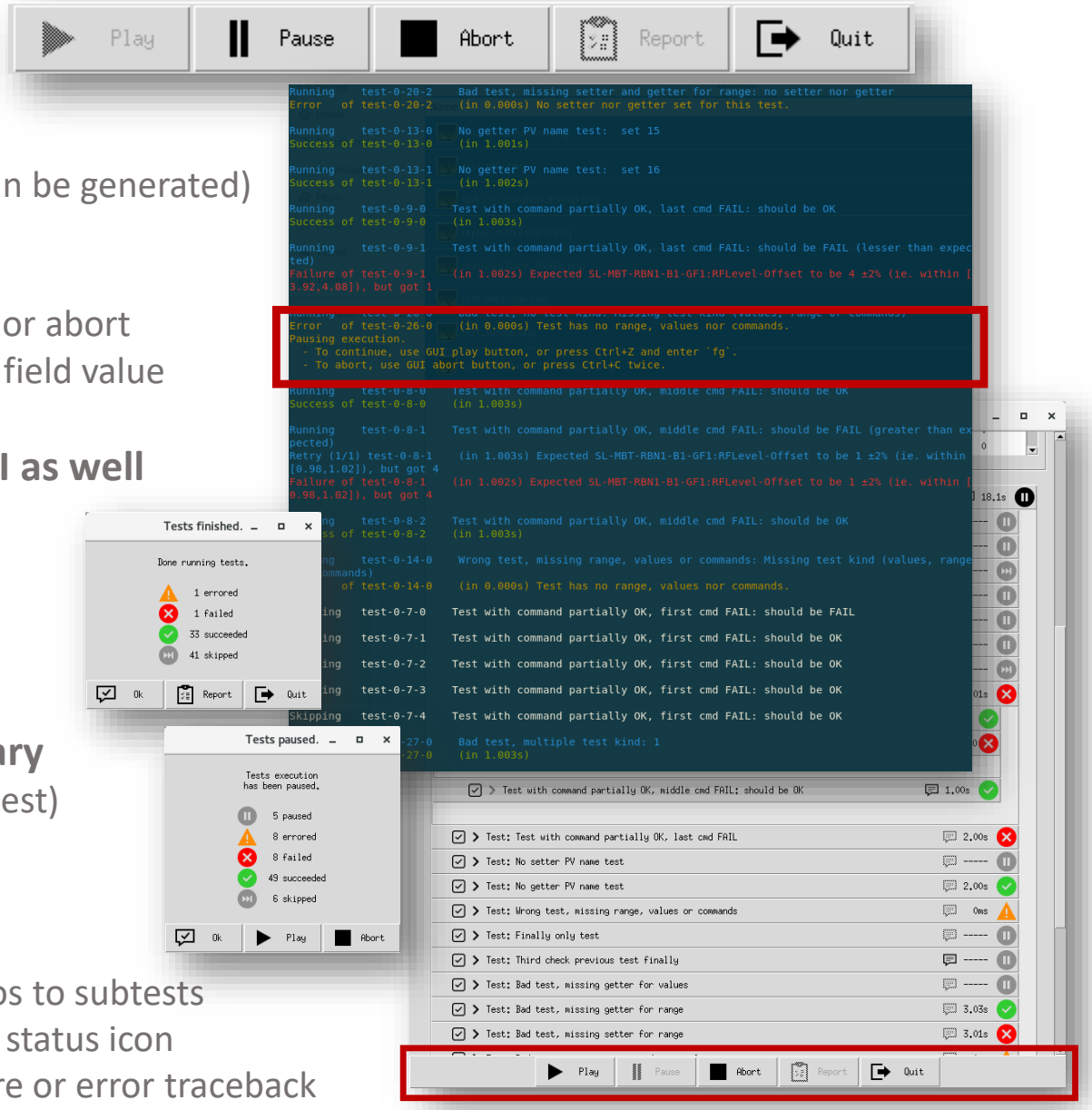
Execution and GUI: Execution and Results

- **Control bar at GUI bottom**
  - start or resume testing
  - pause the tests temporarily
  - abort the testing (no report can be generated)
  - open latest report
  - quit GUI and abort tests
  - note that tests can also pause or abort automatically with on_failure field value

- **Pause and Abort available in CLI as well**
  - start using Ctrl+D and Enter
  - abort using Ctrl+C
  - pause using Ctrl+Z
  - resume using Ctrl+Z and `fg`

- **GUI pop-up with results summary**
  - when pausing (either user or test)
  - when aborted or finished

- **GUI tests treeview**
  - collapse/expand from scenarios to subtests
  - collapse/expand all by clicking status icon
  - expand subtest to see its failure or error traceback



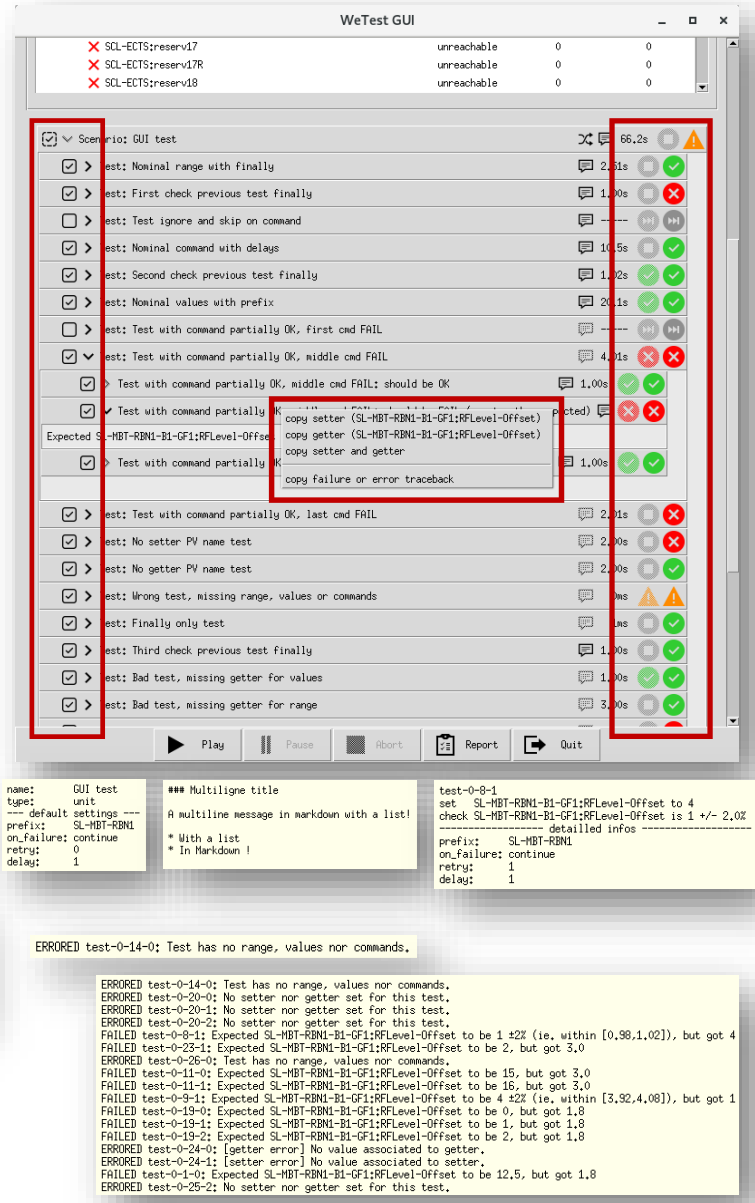
Execution and GUI: Additional GUI features

- **Selection (preset according to skip field)**
  - can select which subtest to execute
  - can select whole or partial scenario and test as well

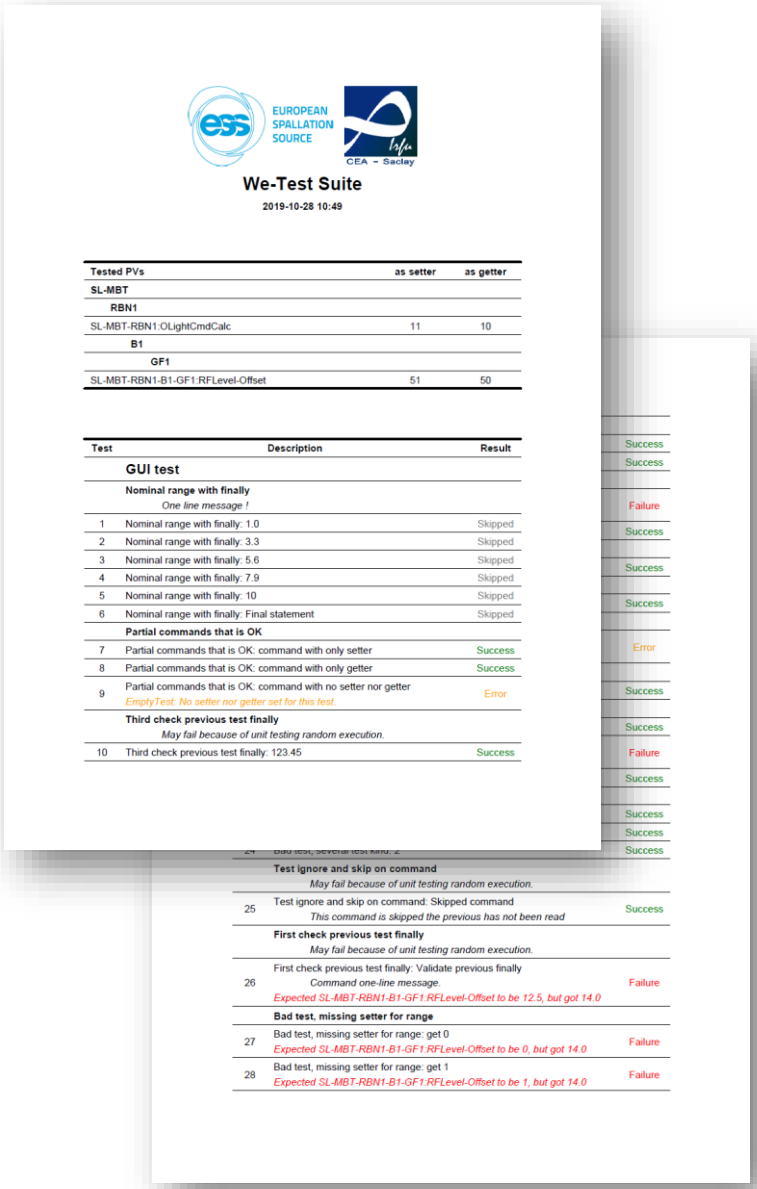
- **Replay**
  - previous icon status still shown with its traceback
  - selection change during run is only applies at next run

- **Multiple tooltips**
  - hover above scenario to see config
  - hover above test to see message
  - hover above subtest to see set and expected values and other parameters
  - hover above test duration to see retry details and previous run duration
  - hover above status icon to see traceback summary

- **Copy to clipboard**
  - by right-clicking on subtest
  - enable to copy setter, getter or all PV names (to use for instance with caget or camonitor in CLI)
  - enable to copy traceback if any



Tests results Report



EUROPEAN SPALLATION SOURCE

CEA – Saclay

We-Test Suite

2019-10-28 10:49

Tested PVs	as setter	as getter
SL-MBT		
RBN1		
SL-MBT-RBN1:OLightCmdCalc	11	10
B1		
GF1		
SL-MBT-RBN1-B1-GF1:RFLevel-Offset	51	50

Test	Description	Result
	GUI test	
	Nominal range with finally	
	One line message !	
1	Nominal range with finally: 1.0	Skipped
2	Nominal range with finally: 3.3	Skipped
3	Nominal range with finally: 5.6	Skipped
4	Nominal range with finally: 7.9	Skipped
5	Nominal range with finally: 10	Skipped
6	Nominal range with finally: Final statement	Skipped
	Partial commands that is OK	
7	Partial commands that is OK: command with only setter	Success
8	Partial commands that is OK: command with only getter	Success
9	Partial commands that is OK: command with no setter nor getter	Error
	EmptyTest: No setter nor getter set for this test.	
	Third check previous test finally	
	May fail because of unit testing random execution.	
10	Third check previous test finally: 123.45	Success

Success
Success
Failure
Success
Success
Success
Error
Success
Success
Failure
Success
Success
Success
Success

	Test ignore and skip on command	
	May fail because of unit testing random execution.	
25	Test ignore and skip on command: Skipped command	Success
	This command is skipped the previous has not been read	
	First check previous test finally	
	May fail because of unit testing random execution.	
26	First check previous test finally: Validate previous finally	Failure
	Command one-line message.	
	Expected SL-MBT-RBN1-B1-GF1:RFLevel-Offset to be 12.5, but got 14.0	
	Bad test, missing setter for range	
27	Bad test, missing setter for range: get 0	Failure
	Expected SL-MBT-RBN1-B1-GF1:RFLevel-Offset to be 0, but got 14.0	
28	Bad test, missing setter for range: get 1	Failure
	Expected SL-MBT-RBN1-B1-GF1:RFLevel-Offset to be 1, but got 14.0	

- **A PDF report generated automatically**
  see CLI options to change name or deactivate

- **PV table shows tested PVs**
  - sorted by section if a naming option was provided
  - shows number of time PV is used as setter or getter

- **Informations displayed**
  - Suite, scenario and test title and messages
  - Subtest title, message, status and traceback
- **Informations NOT displayed (yet ?)**
  subtest number of retries, duration and execution date

- **Four possible test status**
  - success when test matches expected outcome
  - failure when test outcome different that expected (the device did not react as expected)
  - error when test did not execute properly (something is wrong with the test written)
  - skipped when test not executed (test skipped in file or not selected in GUI, since the report is not generated at all when tests are aborted)

DEMO

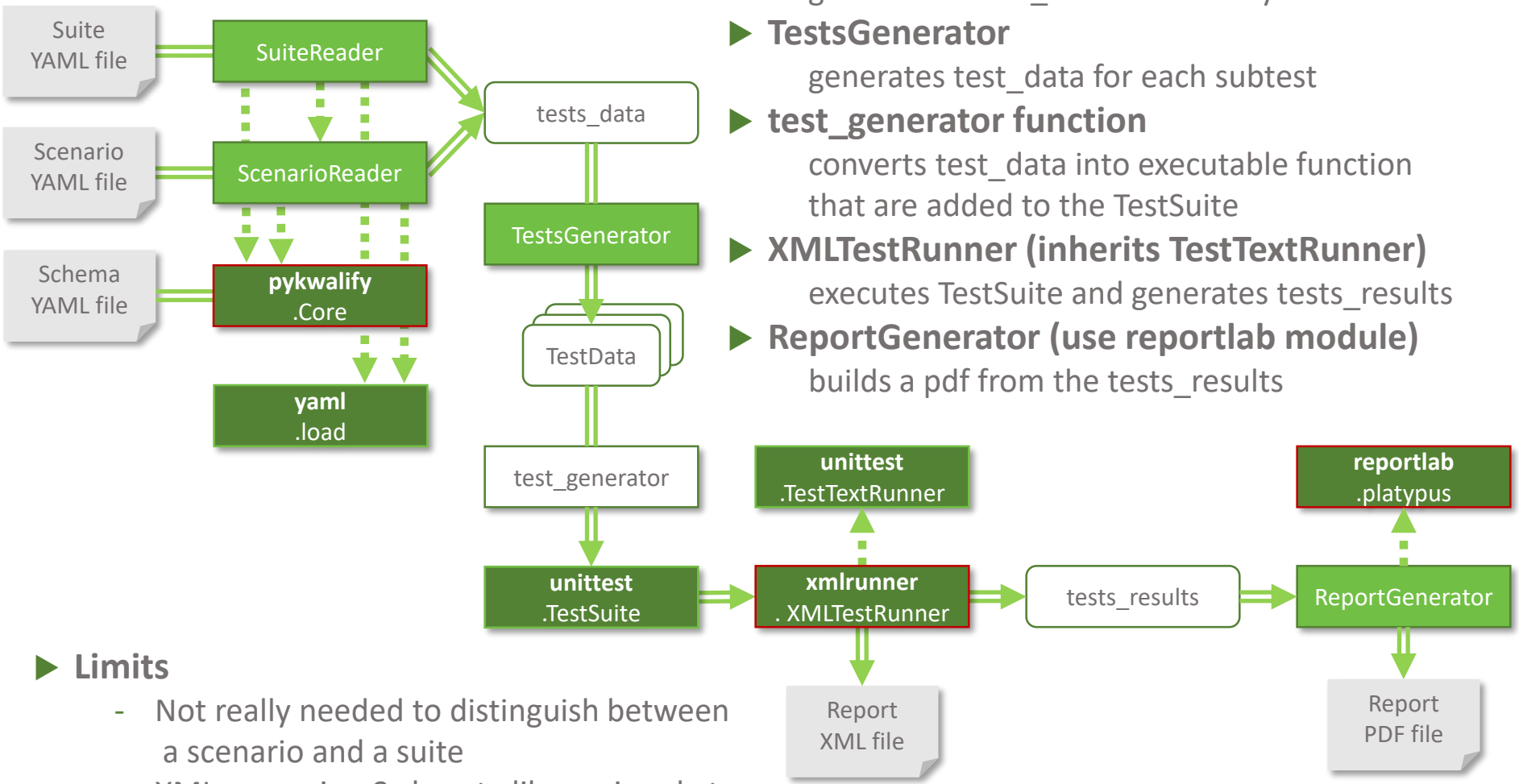
```
$ wetest  wetest/tests/gui-testing.yaml -n RDS-81346 -P
Validation of YAML scenario file: /home/fgohier/Developments/we-test/wetest/tests/gui-testing.yaml
Schema validation failed:
 - Value '1' is not of type 'int'. Path: '/version/major'.
Non-compulsory validation failed:
 - 'Wrong test, missing range, values or commands' should have a at least one of range, commands, values
 - 'Finally only test' should have a at least one of range, commands, values
 - 'Bad test, missing setter and getter for range' is of kind 'range' but has no setter or getter
 - 'Bad test, no test kind' should have a at least one of range, commands, values
 - 'Bad test, several test kind' should have a uniq kind but has 2 (['range', 'values'])
Loaded 76 tests from `We-Test Suite`:
    - unit tests (random)   `GUI test`
PV is unreachable: SL-MBT-RBN1:OLightCmdCalc
PV is unreachable: SL-MBT-RBN1-B1-GF1:RFLevel-Offset
Not starting tests as required.
----------------------
Ready to start testing, use GUI play button.
Waiting for user.
  - To start testing, use GUI play button.
  - To abort, use GUI abort button, or press Ctrl+C.
PV changed to connected: SL-MBT-RBN1-B1-GF1:RFLevel-Offset
PV changed to connected: SL-MBT-RBN1:OLightCmdCalc
CA.Client.Exception.....................................................
    Warning: "Identical process variable names on multiple servers"
    Context: "Channel: "SL-MBT-RBN1-B1-GF1:RFLevel-Offset", Connecting to: 10.2.176.135:5064, Ignored: lo
calhost:5064"
    Source File: ../cac.cpp line 1306
    Current Time: Thu Oct 24 2019 16:52:00.133059699
.....................................................................
CA.Client.Exception.....................................................
    Warning: "Identical process variable names on multiple servers"
    Context: "Channel: "SL-MBT-RBN1:OLightCmdCalc", Connecting to: 10.2.176.135:5064, Ignored: localhost:
5064"
errlog: 1 messages were discarded
    Source File: ../cac.cpp line 1306
    Current Time: Thu Oct 24 2019 16:52:00.133138440
.....................................................................
Applying selection may take some time.

Skipping    test-0-15-0    Finally only test: Final statement

Running     test-0-0-0     Nominal range with finally: 1.0
Success of test-0-0-0      (in 0.531s)

Running     test-0-0-1     Nominal range with finally: 3.3
Success of test-0-0-1      (in 0.502s)

Running     test-0-0-2     Nominal range with finally: 5.6
Success of test-0-0-2      (in 0.502s)
```

- Implementation Architecture
- File reading and validation
- CLI and GUI communication
- Naming validation
- PDF Report generation
- Other changes planned
- WeTest development workflow
- Sharing WeTest

Part II – Behind the scene

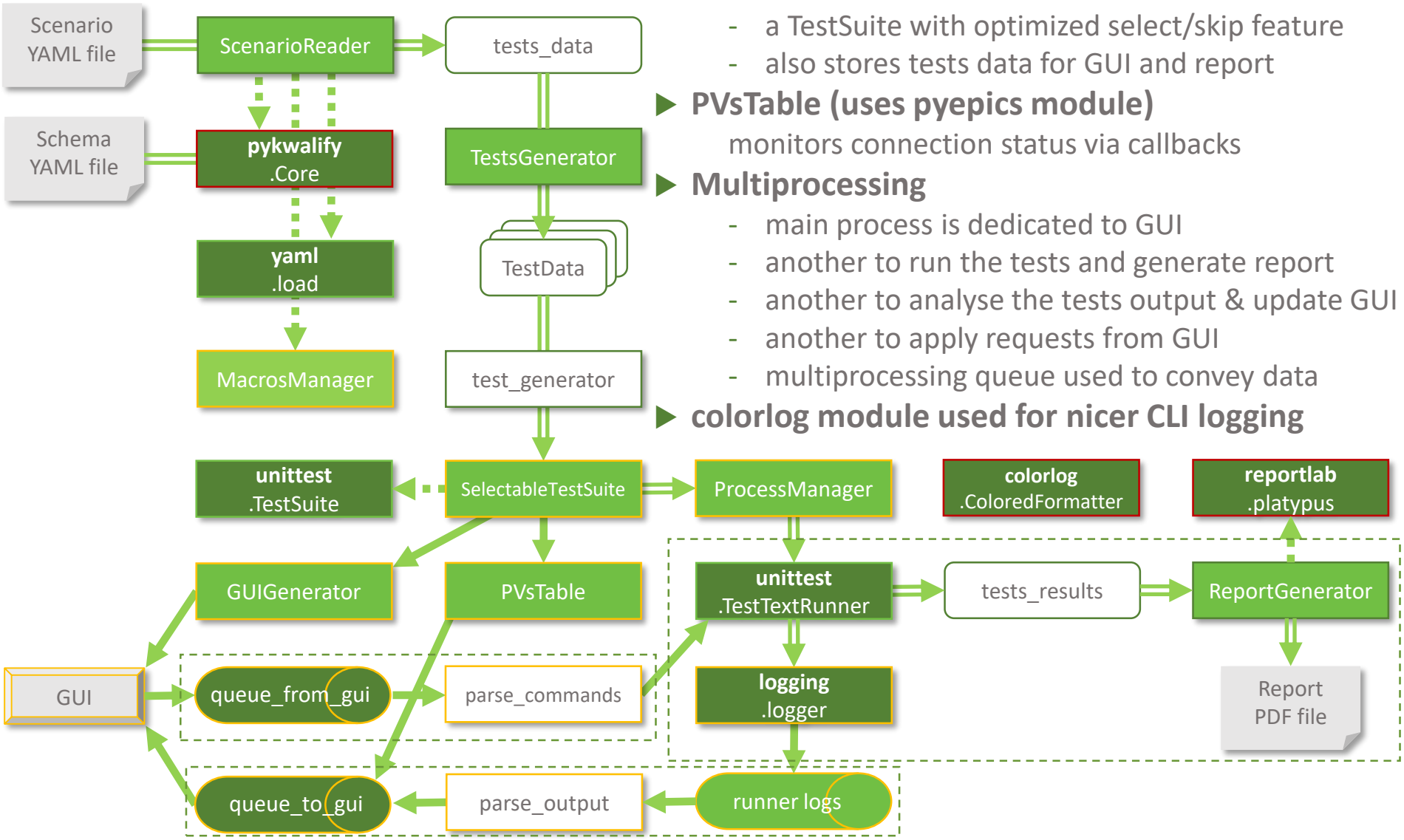
Implementation Architecture: Previous version (before 2019)

- **Readers (use pykwalify and yaml modules)**
  generate a tests_data dictionnary with file content
- **TestsGenerator**
  generates test_data for each subtest
- **test_generator function**
  converts test_data into executable function that are added to the TestSuite
- **XMLTestRunner (inherits TestTextRunner)**
  executes TestSuite and generates tests_results
- **ReportGenerator (use reportlab module)**
  builds a pdf from the tests_results



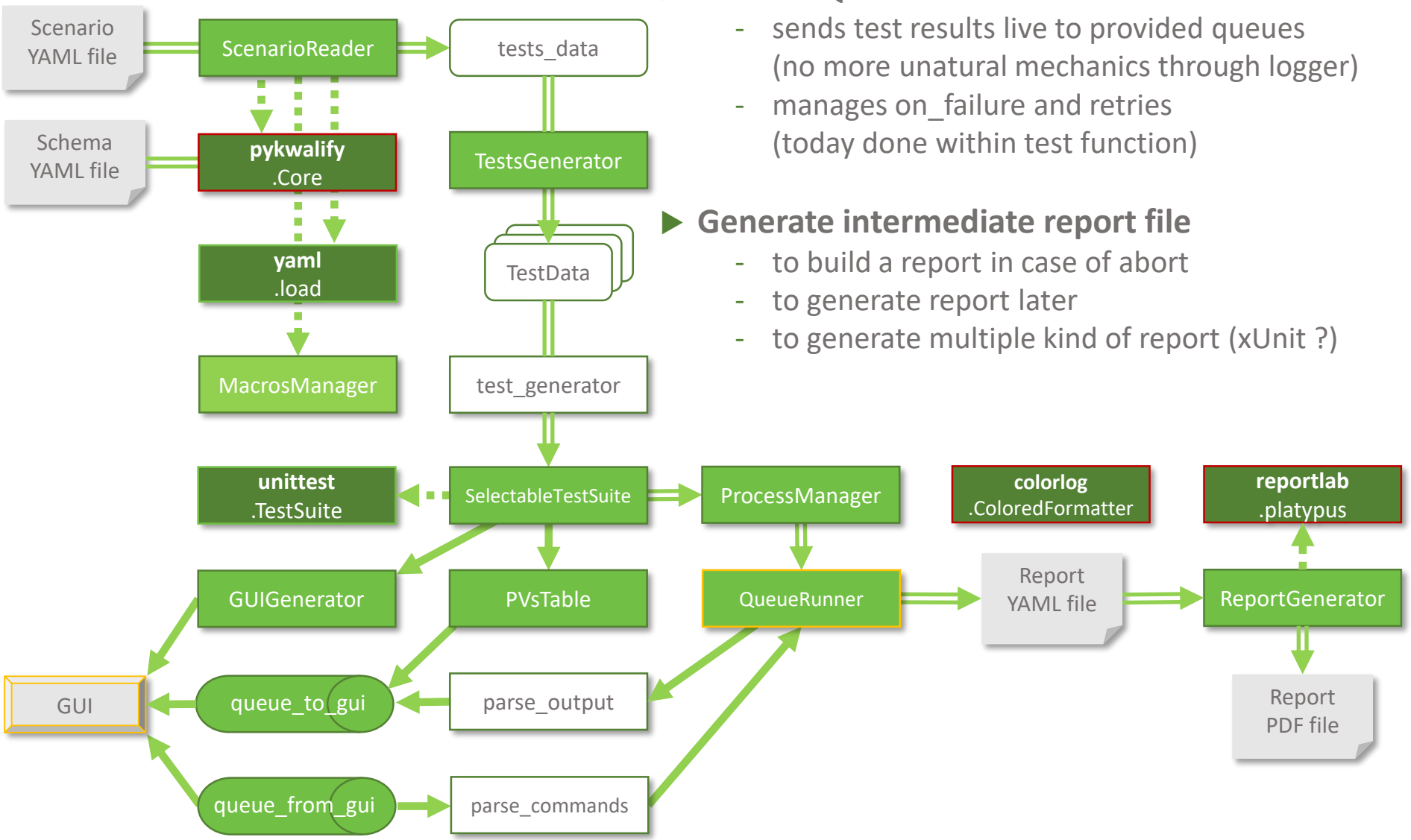
- **Limits**
  - Not really needed to distinguish between a scenario and a suite
  - XMLrunner is a 3rd-party libary aimed at generating xUnit XML file that we don't use

Implementation Architecture: Current version (end of 2019)



- **SelectableTestSuite**
  - a TestSuite with optimized select/skip feature
  - also stores tests data for GUI and report
- **PVsTable (uses pyepics module)**
  monitors connection status via callbacks
- **Multiprocessing**
  - main process is dedicated to GUI
  - another to run the tests and generate report
  - another to analyse the tests output & update GUI
  - another to apply requests from GUI
  - multiprocessing queue used to convey data
- **colorlog module used for nicer CLI logging**

Implementation Architecture: Future version



- **Custom QueueRunner**
  - sends test results live to provided queues (no more unatural mechanics through logger)
  - manages on_failure and retries (today done within test function)
- **Generate intermediate report file**
  - to build a report in case of abort
  - to generate report later
  - to generate multiple kind of report (xUnit ?)

File reading and validation

File reading

- read using base module yaml
- transform the files into a python dictionary
- dictionnary is then processed to generate all the corresponding subtests

File validation

- a schema is defined to check file content using pykwalify
- more complex tests are implemented to test composite fields type and fields compatibility

Macros substitution

- done using regex.substitute
- therefore every is converted into a string
- and is then converted back into float, integers, boolean, dict, list or left as a string if not possible

```
type: map
required: yes
mapping:
    "version":
        desc: "Version is useful te check compatibility of a file and a WeTest versio
        type: map
        required: yes
        mapping:
            "major":  { type: int, required: yes }
            "minor":  { type: int, required: yes }
            "bugfix": { type: int, required: yes }

    "name":
      type: str
      desc: Top name (top name from included scenario are ignored)

    "macros":
        desc: "Macros can be defined and will be substituted throughout the file"
        type: any  # actually either a map or a sequence of map

    "include":
        desc: "A list of scenario files to include, if not specified `tests` will be
        type: seq
        sequence:
            - desc: "Path to a scenario file"
              type: str
            - desc: "Path to a scenario file and macros"
              type: map
              allowempty: yes  # the map can have keys which are not present in the s
              mapping:
                  "path":  { type: str, required: yes }
            - desc: "Path to a scenario file and macros"
              type: seq
              sequence:
                  - desc: "Path to a scenario file"
                    type: str
                    required: yes
                  - desc: "A dictionnary with one or several macro"
                    type: map
                    allowempty: yes

    "config":
        desc: "This configure the `tests` block"
        type: map
        mapping:
            "name":
              desc: Scenario Name
              type: str
              required: yes
            "type":
               type: str
               enum: ['unit', 'functional']
               desc: When `unit` is chosen, tests will be shuffled for random execut
            "prefix":     { type: str   }
            "use_prefix": { type: bool  }

            "delay":      { type: float }
            "ignore":     { type: bool  }
```

CLI and GUI communication

Tkinter

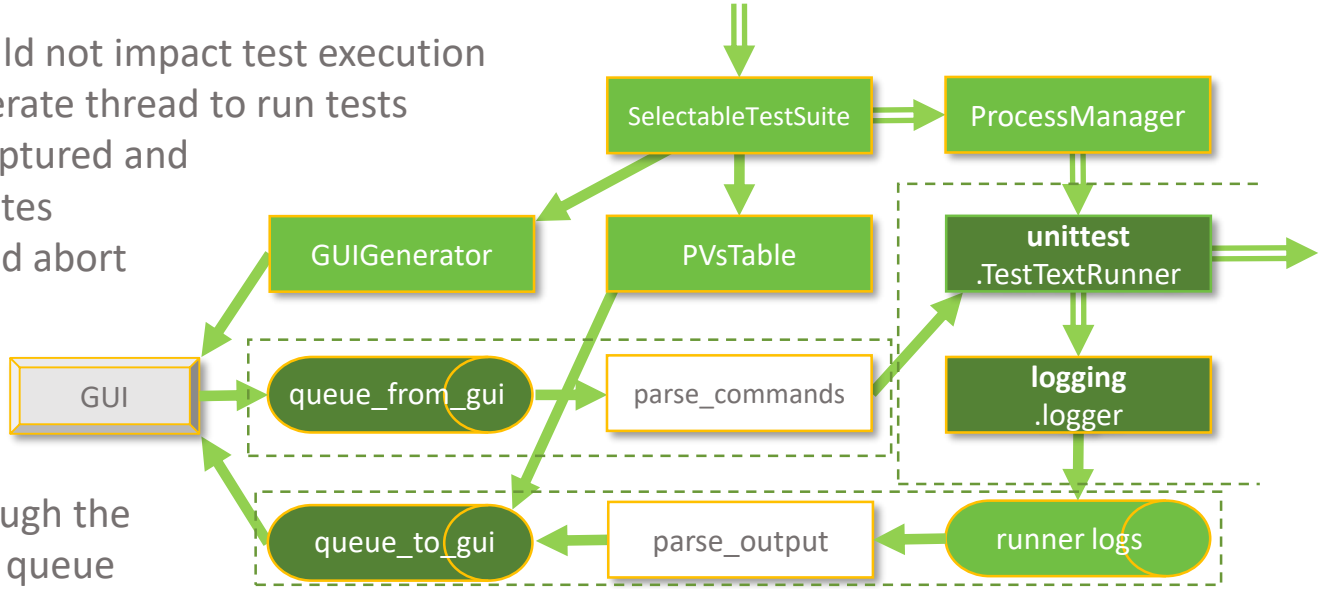
- supposed to be installed by default with python installation
- when not already install should be easilly installable as standard OS package
- requires the GUI to be run in the main thread

CLI only mode

- is the historic mode for WeTest
- this mode is still available using option to disable GUI generation
- however WeTest always require Tkinter to be installed (bug #135)

Multithreading

- GUI refreshment should not impact test execution therefore using a seperate thread to run tests
- tests logs in CLI are captured and convert into GUI updates
- GUI pause, resume and abort are convert into actions over the test-running process
- PVs changes sent through the same multiprocessing queue to update GUI



Naming validation

Naming validation

- Naming interface defined with sort, split and ssplit (short split) methods to implement
- Each naming implements the Naming class
- The sort method is used to ensure the PVs are always presented in the same order
- The split and ssplit methods are expected to return a list of the PV sections
- If a PV name is not valid the split method is supposed to return a NamingError
- Naming check can be as strict as desired
- Multiple naming strictness for the same project possible as long as the implemented class has a different name.

TODO

- enable user to provide custom naming without modifing WeTest code (a sort of plugin) (feature #136)
- provided function will need to implement the Naming interface

PDF Report generation

PDF Report construction

- The PDF construction is based on the 3rd-party module reportlab
- Basic reportlab funtionnalities are used as much as possible to format the page, the paragraphs, the tables and images.

- Client en CEA logo are inserted on top of the document (client logo ought to be changeable easily, feature #71)

- The first table extracts the PVs from the SelectableTestsSuite and show how many time each PV is tested as a setter and as a getter.
- The PVs table is generated using the Naming provided in order to present PVs by section, and to show invalid PVs names.

- Tests data stored in the SelectableTestsSuite are recovered and associated to the tests results outputed by the TestsTextRunner
- Each test result is then formatted in a table entry
- Separated by scenario titles and tests titles and messages
- When a test failed or errored its trace is displayed, after being shorten when it maches one of the custom Exception (AssertionError, InconstistantTest, EmptyTest)

- For more information about reportlab see Reportlab PDF Generation User Guide: https://www.reportlab.com/docs/reportlab-userguide.pdf

Other Changes Planned

GUI major changes planned

- choose all of CLI option from a GUI (feature #134)
- sort by execution duration, by execution order and test results (feature #121)
- pattern filter for PV name (feature #116)
- show number or retry (feature #104) and execution date (feature #138)
- traceback text should be wrapped to fit windows width (feature #126)
- warn when the Report button aims at an older report (feature #115)

Macros major changes planned

- macros block should only apply to current file and not to included files (bug #139)
- macros starting with underscore (_) should be considered private to file and not changable
- enable user to choose macro type (using yaml basic features ?) (feature #141)

Scenario Schema major changes planned

- more mathematical comparison (precision, greater than, lesser than, different) (feature #68)
- dynamic finally value (only enter PV name and its value will be restored after test) (#52)

Other open bugs

- system slowdown are sometime observed when WeTest has been running for a while (#132)
- WeTest subprocess have been observed to keep running when WeTest is closed (#129)

WeTest development workflow

Using Gitlab

- as much as possible the issues are fixed and commit one at the time
- branches rarelly used (except for major changes) since only one developper

Testing

- automatic tests have not been used nor updated for years now (issue #74)
- tests are done manually using home-made mockioc script:
  https://drf-gitlab.cea.fr/epics-tools/mockioc
  (which accept an EPICS DB an emulate the IOC using PCASpy module)

Update required

- write a proper documentation (today only this presentation an README for installation)
- update tests (using pytest and doctest when appropriate)
- make WeTest python 3 compatible
- make WeTest work on Windows
- use Continuous Integration with Tox and Gitlab-CI

Sharing WeTest

- **WeTest plebiscited by users**
  - WeTest well received for delievery in 2017 (Catania) and in 2019 (Saclay)
  - Answers to an actually need
  - GUI and macros make it much nicer to use and watch
  - Retries and pause on failure make it usable to test more complex process
  - ESS Control team showed interest in using it and maybe participate
  - Makes writting and running tests more interesting
  - Automatic testing with WeTest is key for continuous integration of EPICS modules (with actual hardware or emulation via PCASpy or ESS Kameleon simulator)

- **Required before releasing**
  - proper documentation
  - proper code testing to avoid regression
  - proper licensing
  - open-sourcing code on Github (with transfert of all the issues)
  - easy installation via pip and conda

Thank you for your attention

Last updated Novembre 5th 2019

Francis GOHIER