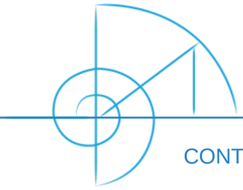
Safety Solutions Designer

DEC 2020

AIX
LYON
PARIS
STRASBOURG

WWW.CLEARSY.COM

Framework for automated testing CLEARSY Test Framework



CONTACT@CLEARSY.COM

Contents

CLEARSY Test Framework

CLEARSY Test Framework is a software workshop designed to perform **black box** oriented **functional tests** of a system.

Main principles

A test stimulates inputs and checks outputs.

Complex conditions may be defined for the checking.

A set of test constitutes a scenario.

Scenarii are described into XML files open to user.

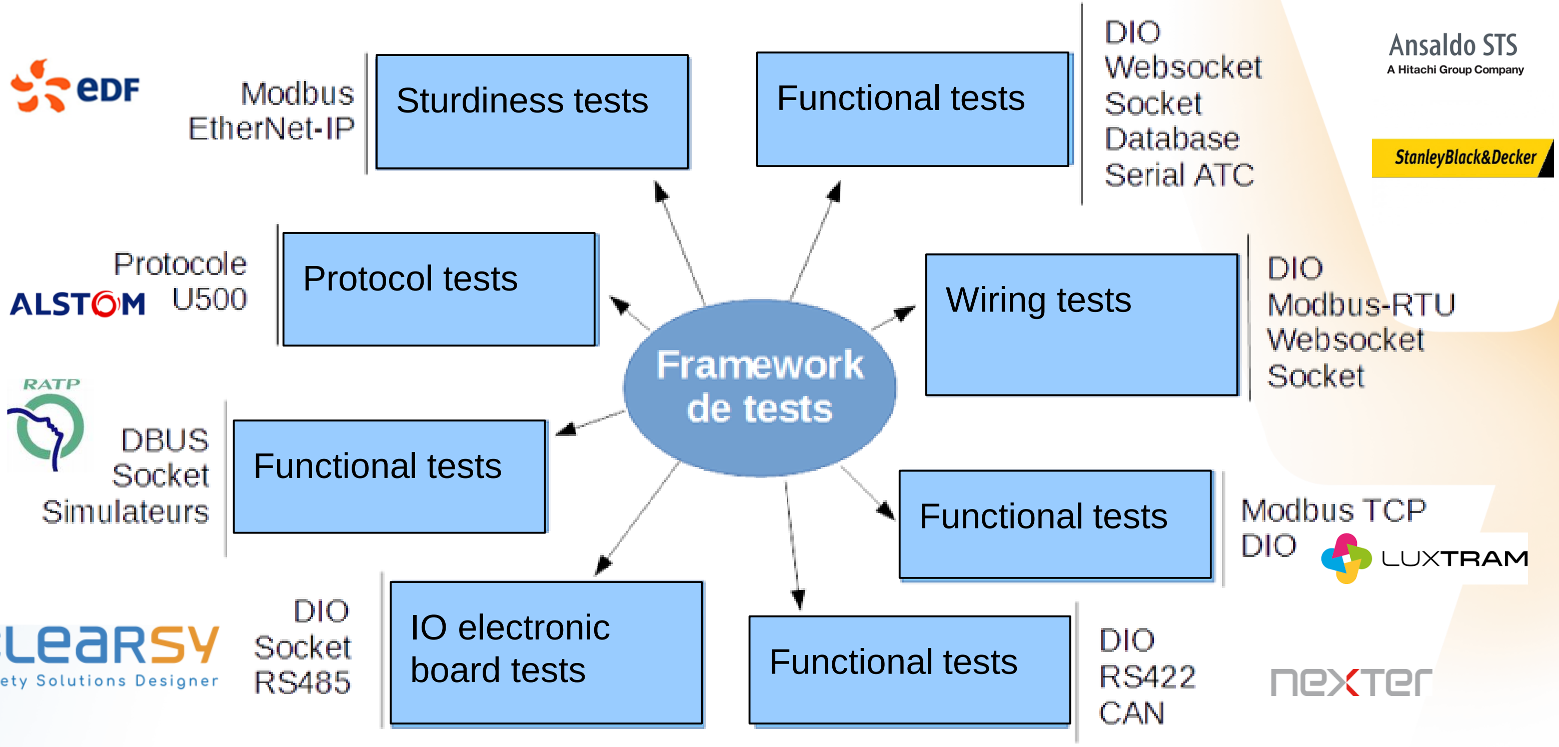
The ordering run of a set of scenarii is possible in order to perform non regressive tests.

Features

- Fits for industrial market :
  Qualified T2 CEI 61508 and EN50128 (for the generic part)
- Flexibility: fits to every test specificity
- Runs on a standard PC
- Windows and Linux
- XML and Xunit standards
- Generation of user formatted reports (docx)

- since 2014
- used on 20+ projects

Fields of use : Interfaces - References



Contents

Test structure

A test follows this pattern:

- According to a context (beginning status)
- If the Framework runs define actions to one or several interfaces
- Then it should detect define consequences on one or several interfaces

Components

CLEARSY Test Framework comprises:

- A **graphical user interface** which allows to launch tests and show results
- A **Runner** (the orderer, the core), available as a runtime
- **Mocks**, modules which extend Runner functionalities

  A Mock typically implements the behaviour of a protocol

  Mocks are instantiated and controller by the Runner

  A Mock is dedicated to each interface to test

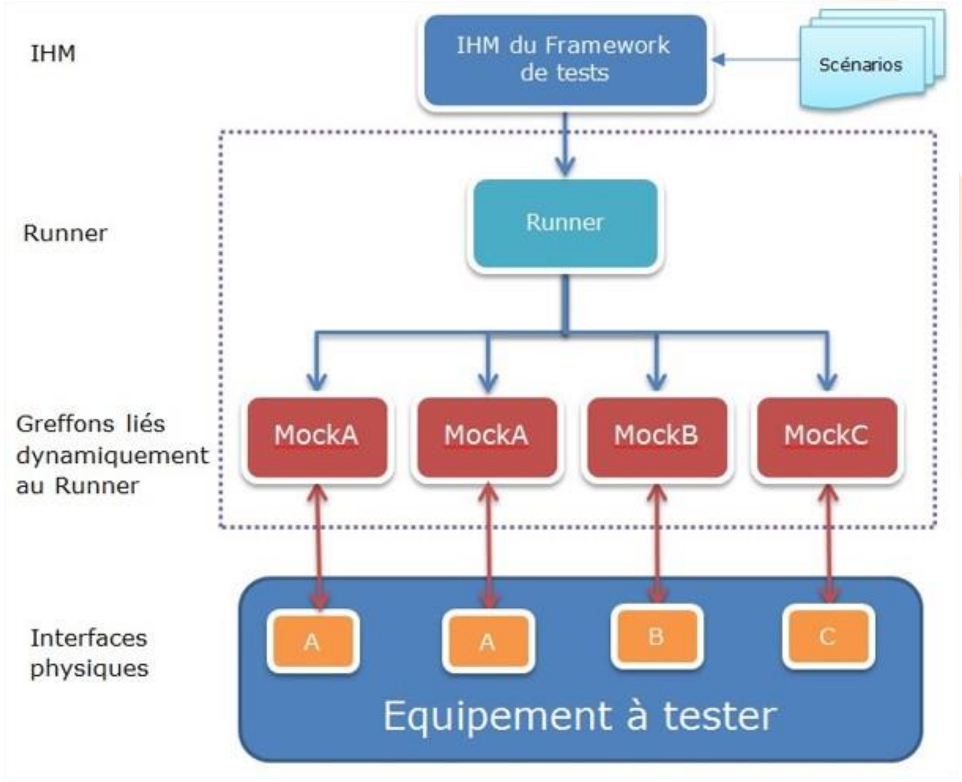
Environment:

Multi plateform : **Windows** or **Linux**

Architecture

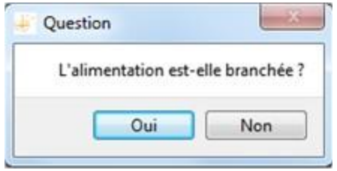
Mocks are dynamically linked to Runner exe

Physical IO



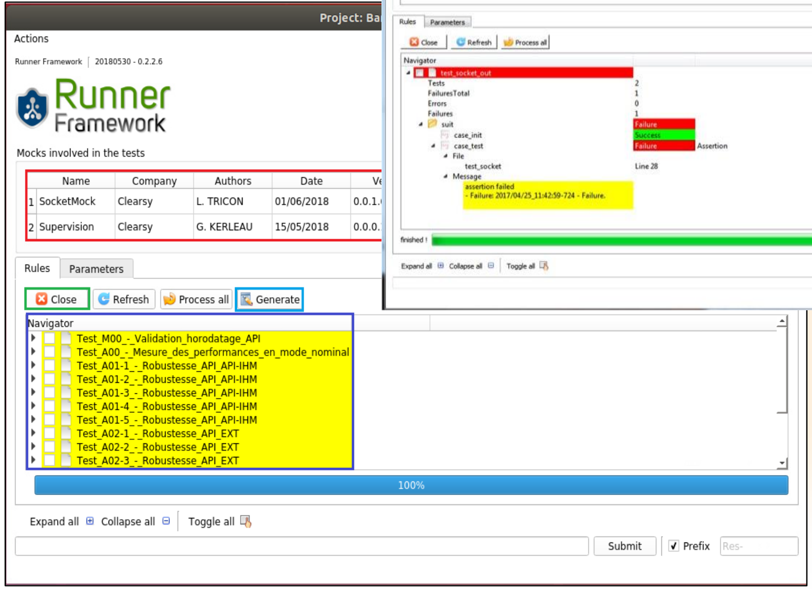
Graphical interface

- Loads the scenarios
- Launches the tests
- Shows user’s pop-ups required by tests


Question

L'alimentation est-elle branchée ?

- Generates tests reports


Project: Ba

Actions

Runner Framework | 20180530 - 0.2.2.6

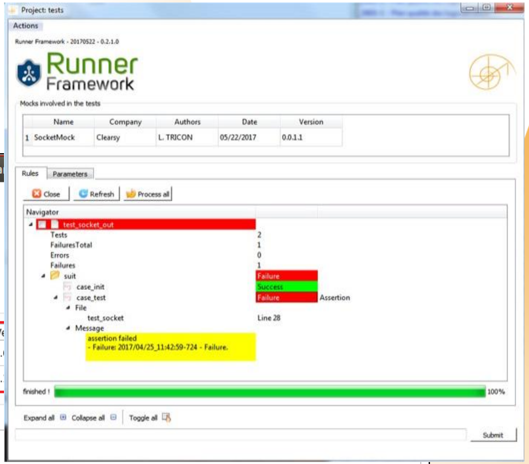
Runner Framework

Mocks involved in the tests

	Name	Company	Authors	Date	Ve
1	SocketMock	Clearsy	L. TRICON	01/06/2018	0.0.1.
2	Supervision	Clearsy	G. KERLEAU	15/05/2018	0.0.0.

Navigator
- Test_M00_-_Validation_horodatage_API
- Test_A00_-_Mesure_des_performances_en_mode_nominal
- Test_A01-1_-_Robustesse_API_API-IHM
- Test_A01-2_-_Robustesse_API_API-IHM
- Test_A01-3_-_Robustesse_API_API-IHM
- Test_A01-4_-_Robustesse_API_API-IHM
- Test_A01-5_-_Robustesse_API_API-IHM
- Test_A02-1_-_Robustesse_API_EXT
- Test_A02-2_-_Robustesse_API_EXT
- Test_A02-3_-_Robustesse_API_EXT

100%


Project: tests

Actions

Runner Framework - 20170522 - 0.2.1.0

Runner Framework

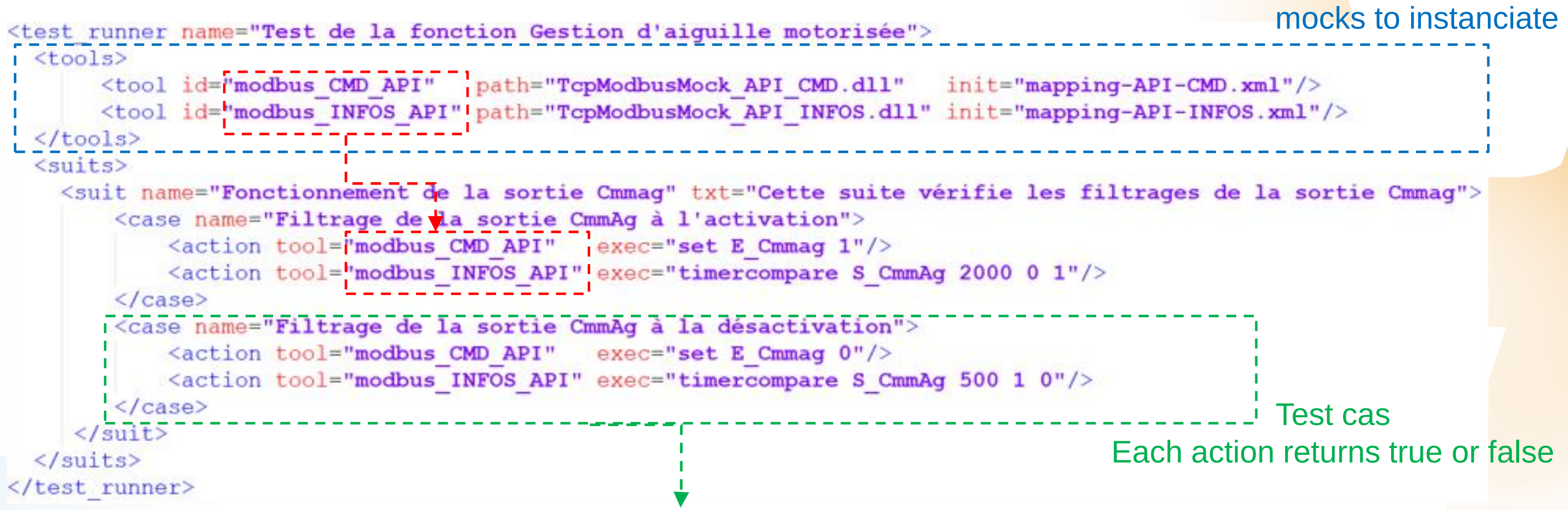
Mocks involved in the tests

	Name	Company	Authors	Date	Version
1	SocketMock	Clearsy	L. TRICON	05/22/2017	0.0.1.1

```
Navigator
test_socket_out
    Tests                         2
    FailuresTotal                 1
    Errors                        0
    Failures                      1
    suit                          Failure
        case_init                 Success
        case_test                 Failure   Assertion
            File
                test_socket       Line 28
            Message
                assertion failed
                - Failure: 2017/04/25_11:42:59-724 - Failure.
finished !                                  100%
```

Test example (XML file)

Programmation of a test by an user (from an usual XML editor).
This test carries out two mocks « TCPModbus » which provide Modbus commands and informations (***set*** et ***timercompare***).



```
<test_runner name="Test de la fonction Gestion d'aiguille motorisée">
  <tools>
       <tool id="modbus_CMD_API"    path="TcpModbusMock_API_CMD.dll"    init="mapping-API-CMD.xml"/>
       <tool id="modbus_INFOS_API" path="TcpModbusMock_API_INFOS.dll" init="mapping-API-INFOS.xml"/>
  </tools>
  <suits>
    <suit name="Fonctionnement de la sortie Cmmag" txt="Cette suite vérifie les filtrages de la sortie Cmmag">
        <case name="Filtrage de la sortie CmmAg à l'activation">
            <action tool="modbus_CMD_API"   exec="set E_Cmmag 1"/>
            <action tool="modbus_INFOS_API" exec="timercompare S_CmmAg 2000 0 1"/>
        </case>
        <case name="Filtrage de la sortie CmmAg à la désactivation">
            <action tool="modbus_CMD_API"   exec="set E_Cmmag 0"/>
            <action tool="modbus_INFOS_API" exec="timercompare S_CmmAg 500 1 0"/>
        </case>
     </suit>
  </suits>
</test_runner>
```

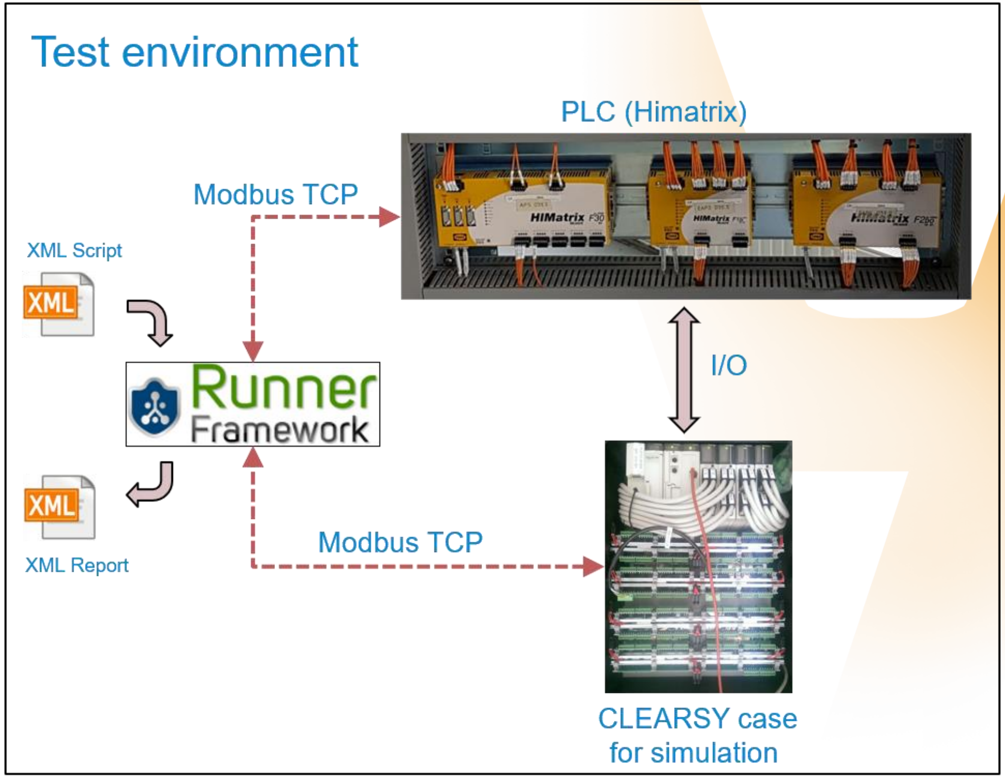
set	updates the variable E_CmmAg with 0 value
timercompare	checks that the S_CmmAg variable has been changing after 500 ms from 1 to 0 value

Use case: SIL4 railway interlocking

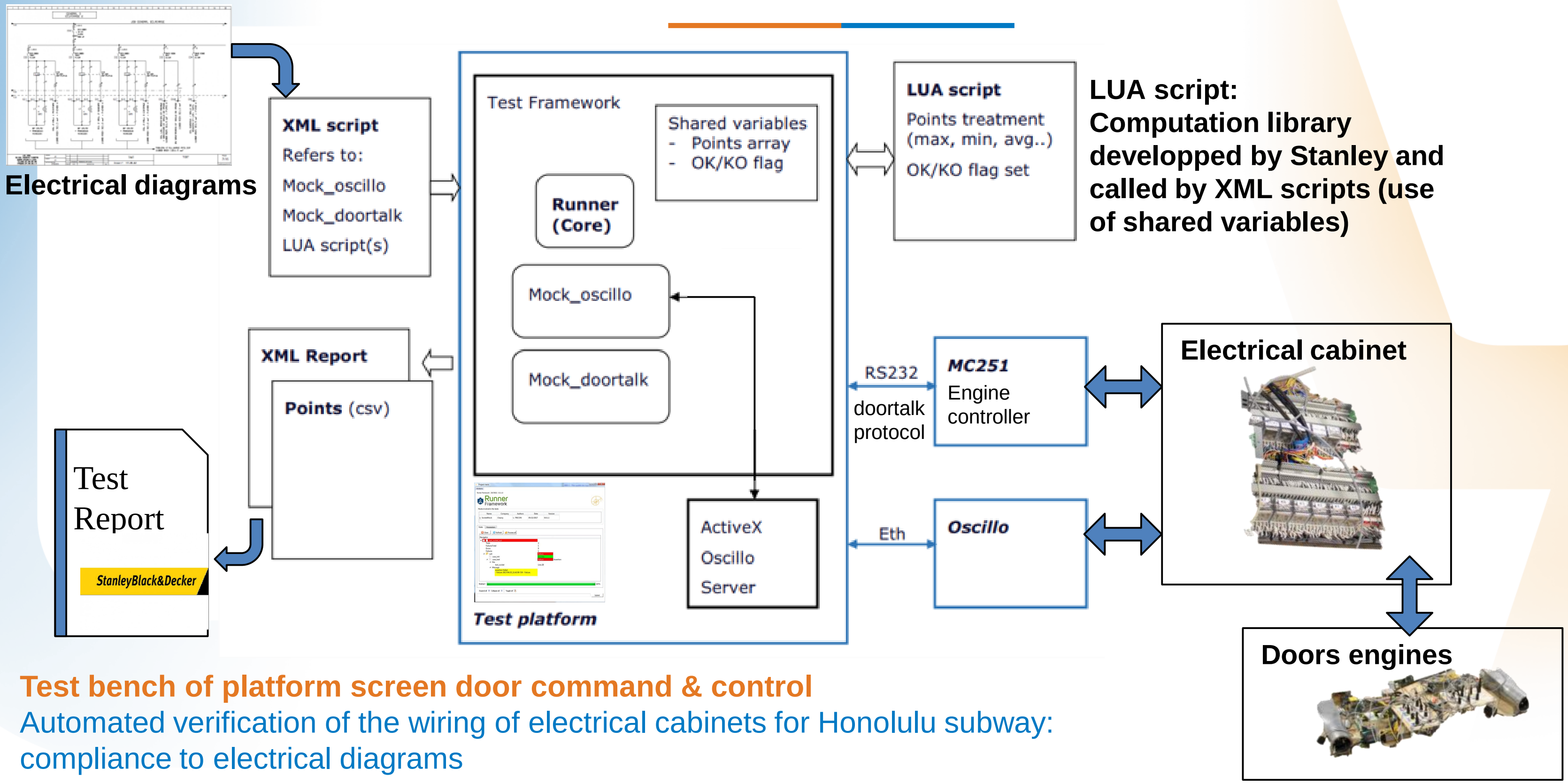
PLC signalling software test

Simulation of environment (lights, switches sensors, track circuits…)

and set of functional tests in compliance with EN50128:SIL4



Use case: Electrical cabinet testing



LUA script: Computation library developped by Stanley and called by XML scripts (use of shared variables)

Test bench of platform screen door command & control

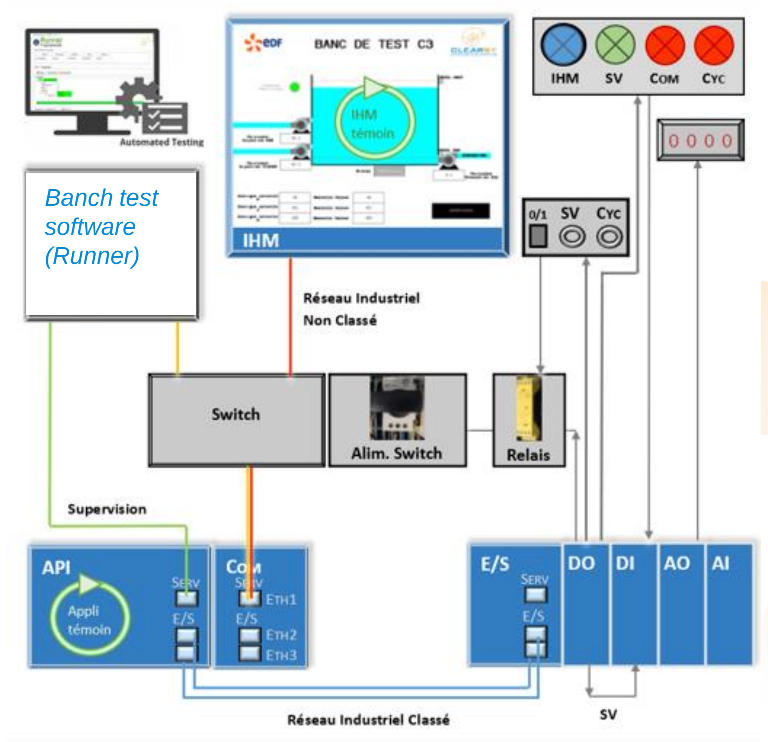
Automated verification of the wiring of electrical cabinets for Honolulu subway: compliance to electrical diagrams

Use case: C3 level communication qualification - Cybersecurity

Qualification of the robustness of a safe class PLC
regardness to pollution coming from an unclassed system (HMI)

Usable for cybersecurity tests (denial of service)

Qualification test case



Contents

Our offer

- The generic solution (plug and play) comprises
  - Available protocols: Modbus/TCP, Modbus/RTU, CAN
  - Standard test conditions:

    Operations = < > on boolean values, integers or strings
- We develop your specific solution
  - Add on required protocols : interfaces et applicative layer
  - Add on test conditions
  - Custom formatted reports (docx)
- Proposed services
  - Co-definition of the needs
  - Development of the specific solution

    Delivery of executable and User Manual (includes installation, test langage and examples)
  - Development and delivery of a complete test bench

    or Support to the testbench development
  - Development of tests *or* Support to test development

CLEARSY Test benches home made: examples

Test benches realized from specification from technical specifications provided by end user customer or defined by CLEARSY

Test bench for RATP

Test bench for EDF

Advantages

- No user license if CLEARSY delivers the whole test bench
- Customized test bench defined from customer requirements and not a customization of the needs to an existing tool !
  - customization of interfaces, protocols, tests, reports
- Customer self sufficiency for tests creation & change
- Test bench reusable on other projects without limitation
- An unique contact on the project for hardware and software issues
- Short time-to-market
  - Average project duration: from 3 to 6 monthes

Contact

contact@clearsy.com

www.clearsy.com

https://www.clearsy.com/outils/bancs-tests/

Paris
Strasbourg
Lyon
Aix en Provence