
CERN

Continuous Integration for PLC-Based Control Systems

Brad Schofield, João Borrego
CERN BE-ICS
05 October 2019

What is Continuous Integration?

- Software development methodology, focusing on:
  - Frequent commits to a repository
  - Automation of build process
  - Automated testing to detect regressions, and check functionality

Motivation: Why CI for PLCs?

- At CERN, we have a widely used framework for industrial control systems: UNICOS
- Use of CI in developing this framework helps detect problems earlier
- We also want to be able to test applications, both new developments and refactoring of existing ones

Challenges

- How to automate the build process?
  - Must use proprietary engineering tools, different for each PLC supplier (Step 7, TIA Portal, Unity etc)
- How to implement tests?
  - Usually don’t want to change the program to implement tests ‘natively’
  - Want to be able to write tests in an easy way

Approach: Testing

- Write tests in Python

  - Common to all PLC types
  - Can take advantage of nice testing packages like unittest, pytest
  - We can easily make abstractions (ie operate on process objects rather than bits in a DB)
- Sounds nice, but how to talk to the PLC?

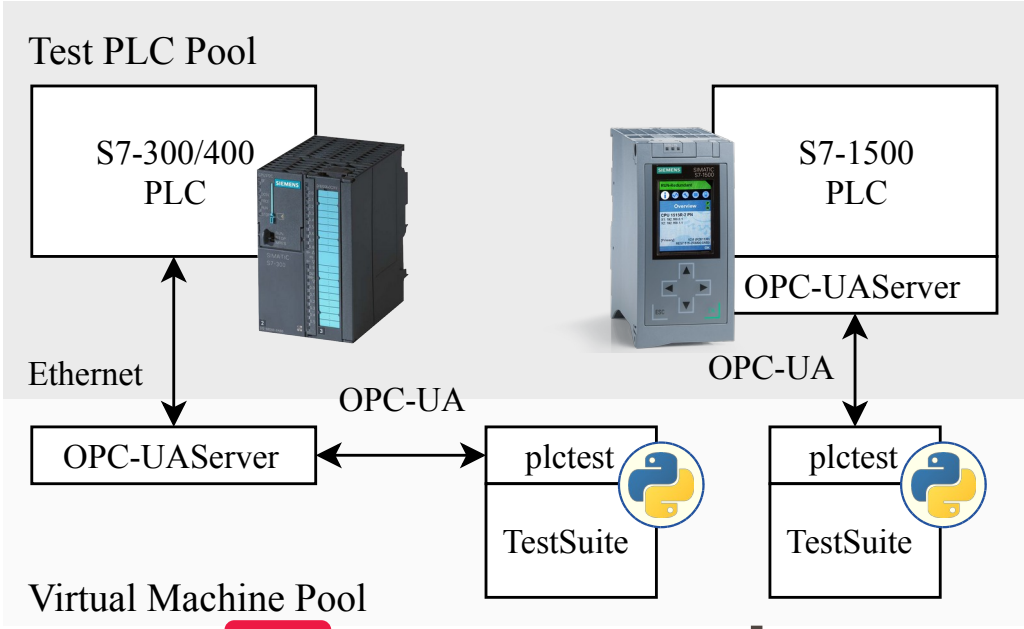
Approach: Communication

- We use OPC UA

  - Open process control communication protocol
  - Gives access to PLC variables without altering program
  - Supported by many PLC types
  - Simple to interface in Python (python-opcua package)

OPC-UA Testing Architecture

- Siemens S7-300/400 uses Simatic NET OPC UA server on VM
- Siemens S7-1500 can use onboard server or Simatic NET
- Test suite can run anywhere with Python



openstack®

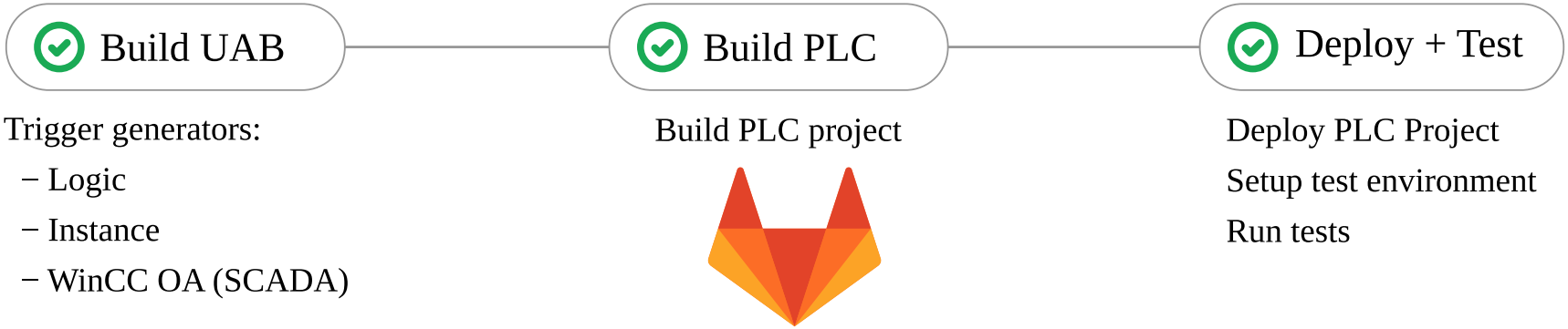
What about the build stage?

- We need to automate tasks usually done in the engineering tools (Step 7, TIA Portal)
  - Import sources
  - Compile
  - Download HW & SW to test target PLCs
- Need to create command line tools based on APIs!

Command line engineering tools

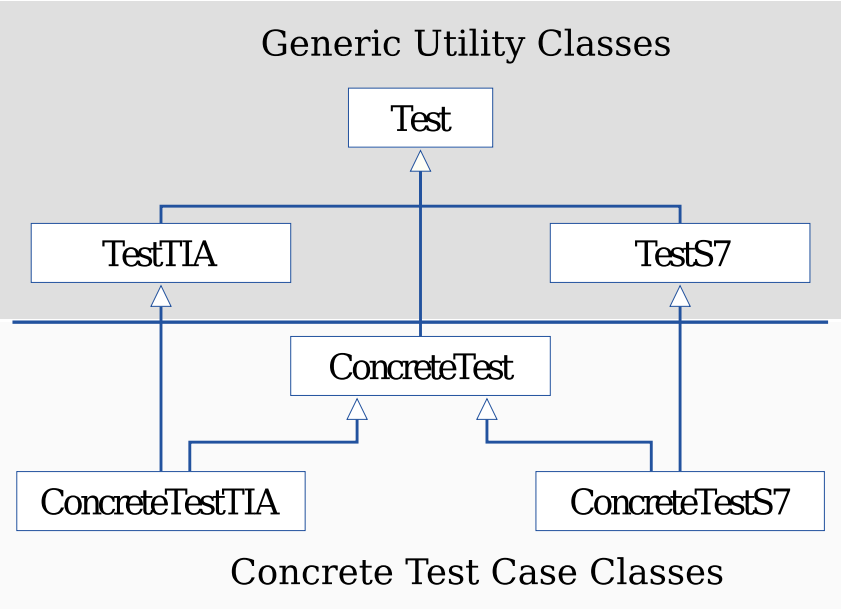
- Together with other colleagues at CERN, we have developed tools based on the C# APIs for Step 7 and TIA Portal
  - Can now import, compile and download to a PLC
  - Allows easy scripting of an automated pipeline
- Gitlab CI growing in use rapidly at CERN, we choose to start there

CI Pipeline (UNICOS)



- GitLab CI pipeline from specification file to test results
- Automatic UNICOS project build with Maven and UAB, deployment with STEP 7 and TIA Portal C# APIs, testing via Python and OPC-UA

Python testing package: py-plc-test



- Wraps OPC UA comms, and knowledge of internal structure
- Provides an abstraction layer to interact with UNICOS objects on higher level
- Can even write one set of tests and instantiate them for different PLC types!

UNICOS Object Abstraction

- OPC-UA provides similar, but not identical interface for different PLC types. UNICOS abstraction means we don't need to worry about this
- Single test description for multiple PLC types

High-level	Internal Implementation
`on_off.set_mode("manual")`	`on_off.set_attribute("AuIhMMo", False)`
	`on_off.reset_register("ManReg01")`
	`on_off.set_attribute("ManReg01.MMoR", False)`

Example: functional test of a UNICOS process object

```
def test_orders_1do_fs_off(self):
    """ORDERS - OnOff with FailSafe Off - On output"""
    on_off = self.on_off
    on_off.configure(fs_pos_on=False, hf_on=True, hf_off=True, pulse=False, hld=True,
                     hld_cmd=False, anim=True, out_off=False, en_rstart=True, rstart_fs=False)

    on_off.set_attributes({"AuOnR": True, "AuOffR": False})
    self.assertEqual(True, on_off.get_attribute("OutOnOV"))

    # Check commands in manual mode
    self.set_mode_assert(on_off, "manual")
    self.assertEqual(True, on_off.get_attribute("OutOnOV"))
    on_off.set_status(False)
    self.assertEqual(False, on_off.get_attribute("OutOnOV"))
    on_off.set_status(True)
    self.assertEqual(True, on_off.get_attribute("OutOnOV"))
    on_off.set_status(False)
    self.assertEqual(False, on_off.get_attribute("OutOnOV"))
```

Demo!

- Let’s look at the pipeline in more detail

Thanks for your attention!