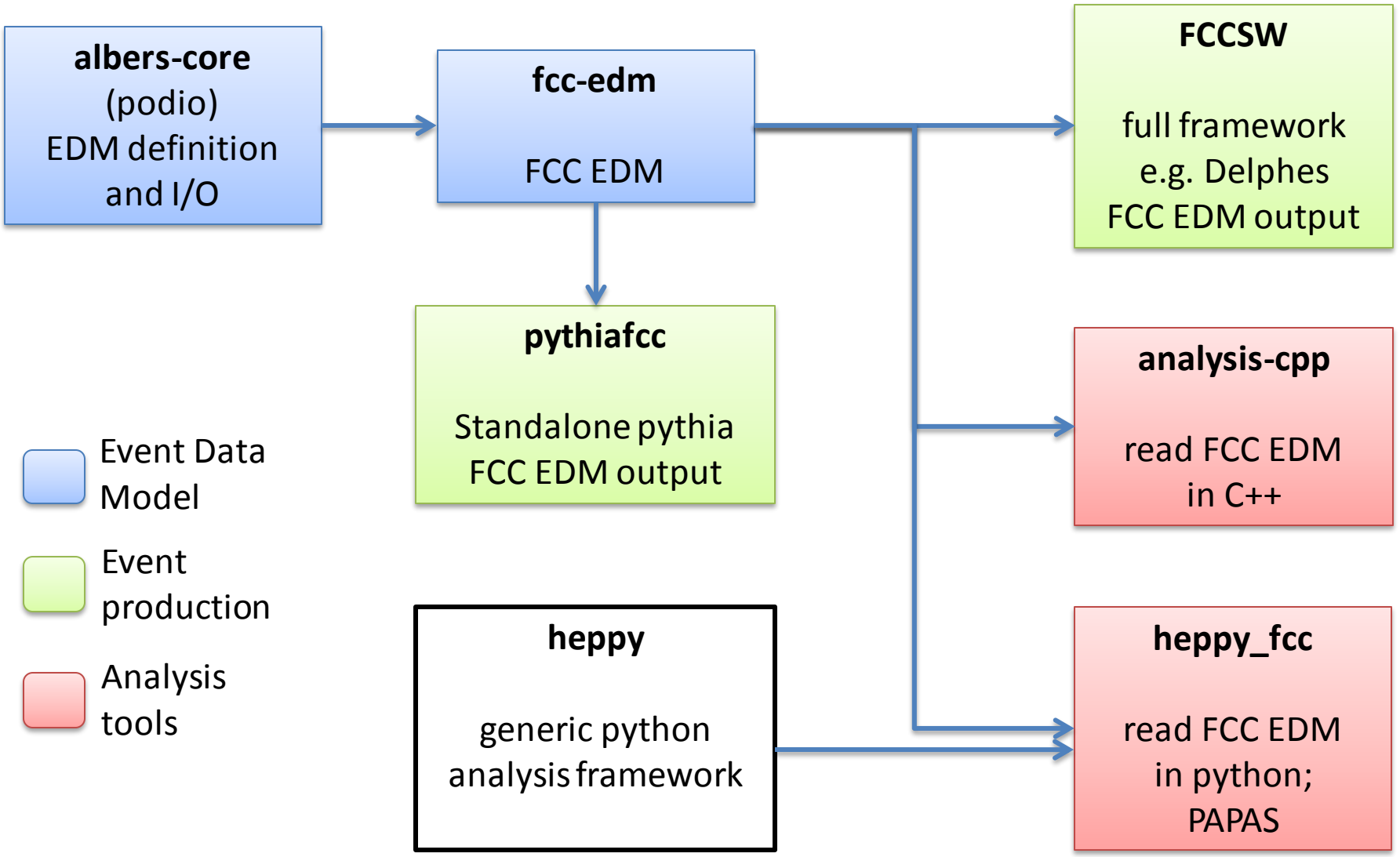
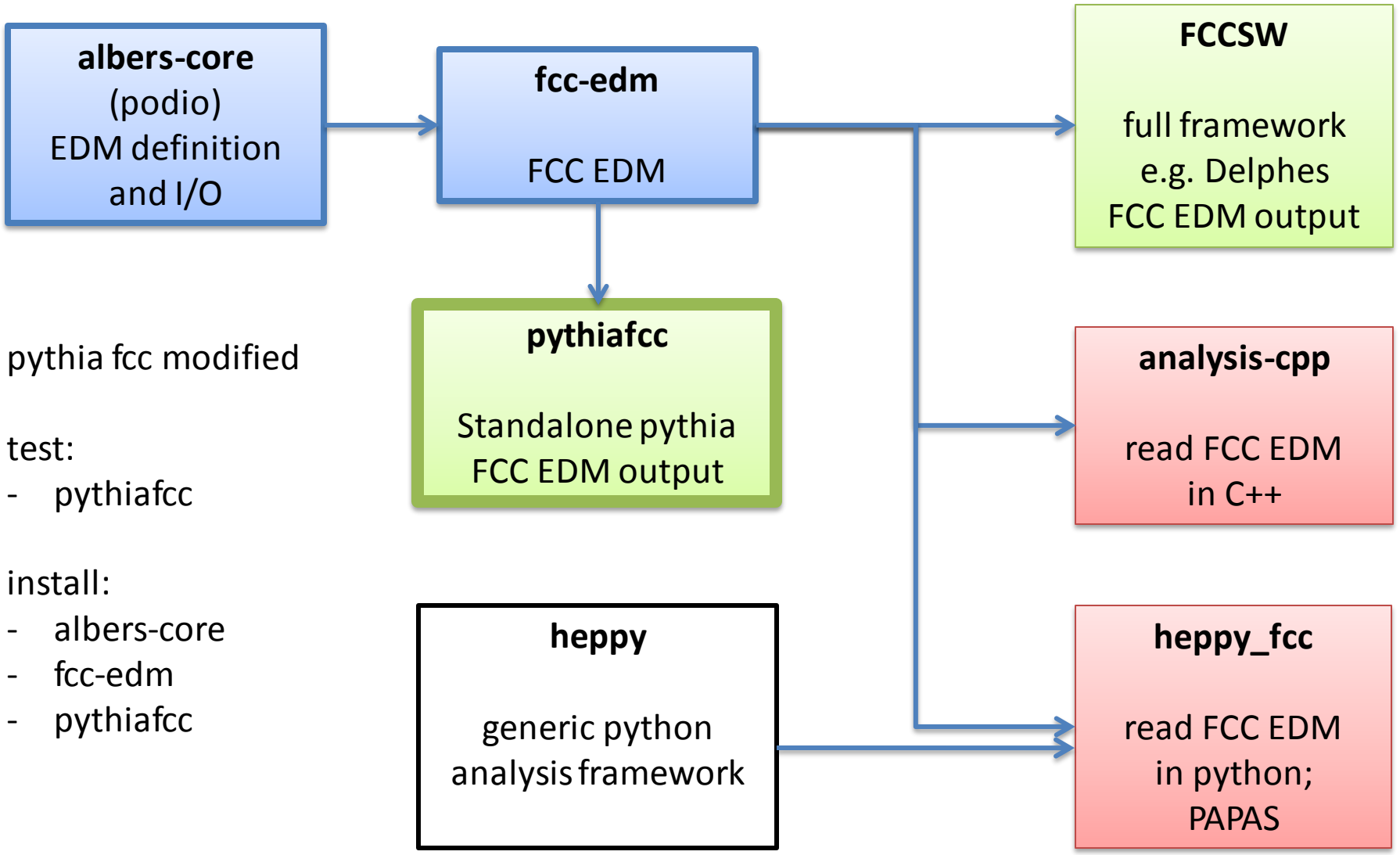
A python-based testing infrastructure

Colin Bernet (IPNL)

The FCC Softare



Testing the FCC Softare

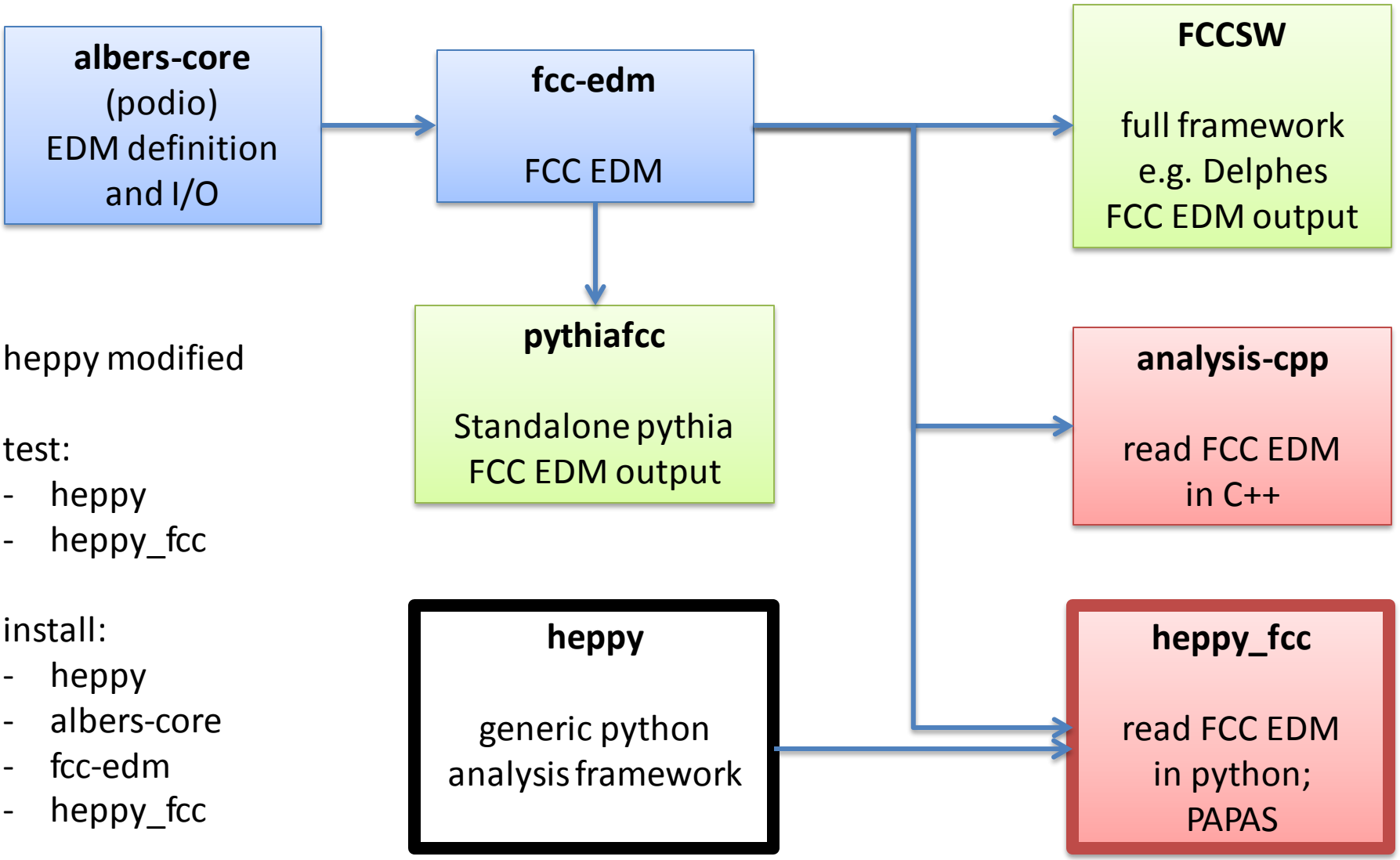


pythia fcc modified

test:
- pythiafcc

install:
- albers-core
- fcc-edm
- pythiafcc

Testing the FCC Softare

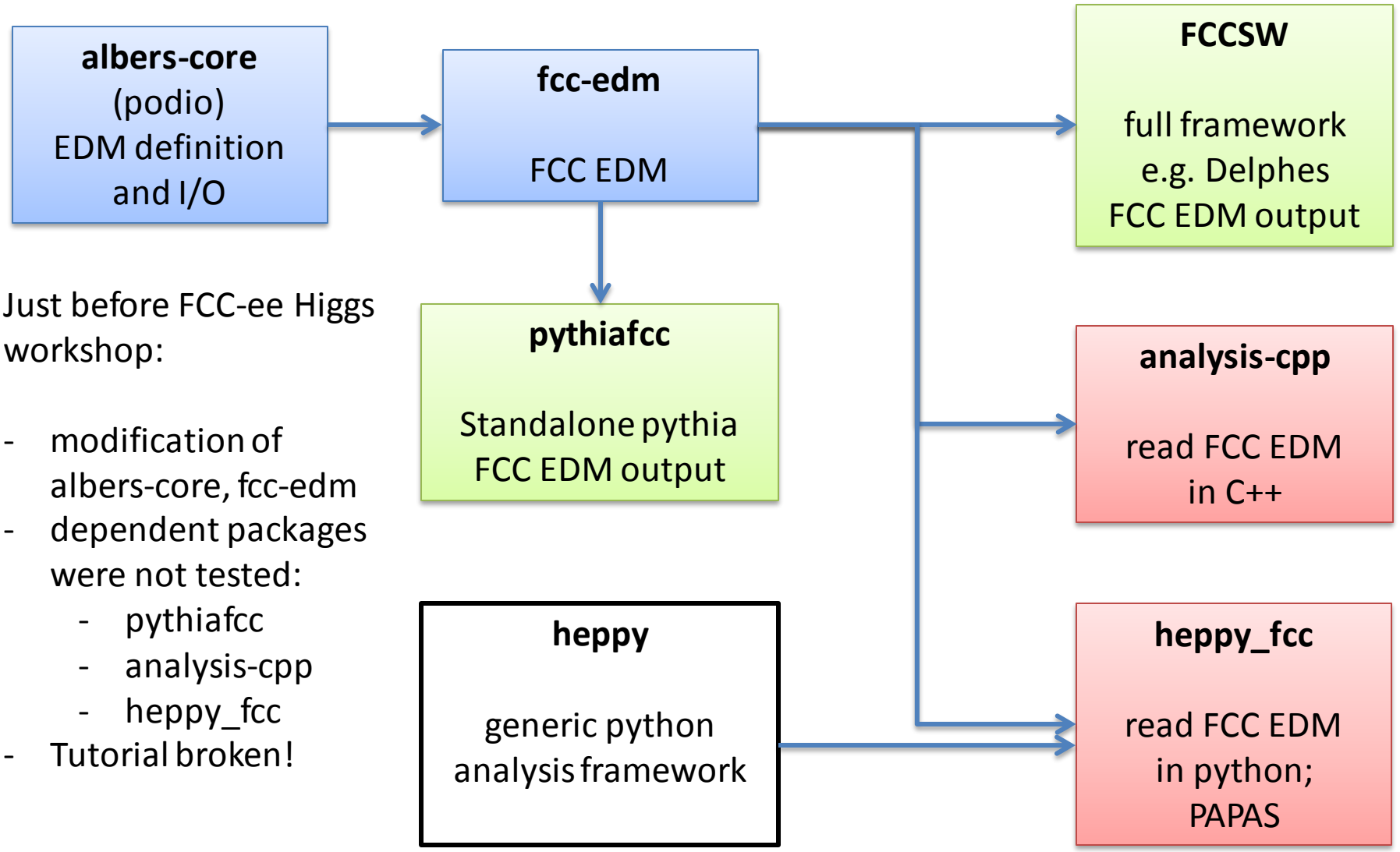


heppy modified

test:
- heppy
- heppy_fcc

install:
- heppy
- albers-core
- fcc-edm
- heppy_fcc

Testing Issues



Just before FCC-ee Higgs workshop:

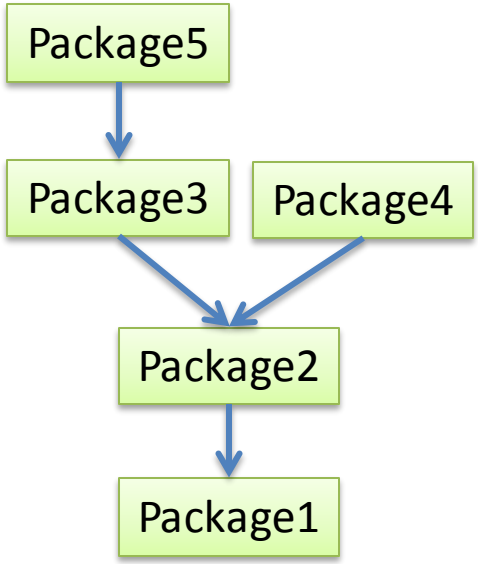
- modification of albers-core, fcc-edm
- dependent packages were not tested:
  - pythiafcc
  - analysis-cpp
  - heppy_fcc
- Tutorial broken!

Why no test?

- Several interdependent packages
  - dependencies not easy to follow
- Several platforms
  - need access to lxplus6, unbuntu, mac os 10.9
- The testing procedure is fully manual:
  - for each platform
    - install all packages
    - follow all tutorial instructions
    - if a problem is spotted, fix it, and repeat the loop
  - pain in the neck, nobody can be blamed for lack of testing at the moment.

What we need – 1

- An automatic dependency tracking system
  - package2 modified
    - test 2
      - install 5 then 3, 4, 2
    - test 1
      - install 1



What we need – 2

- A github robot
  - https://github.com/cms-sw/cmssw/pul
- To:
  - monitor PRs
  - trigger tests
  - send reports

jmduarte commented 4 hours ago

Changes for Razor Hbb HLT DQM for 74X. Backport of #11644.

jmduarte added some commits 4 hours ago

`changes for razor hbb hlt dqm 74x` 123197c

`changes for razor hbb hlt dqm 74x` 4576631

cmsbuild added this to the **Next CMSSW_7_4_X** milestone 4 hours ago

cmsbuild added **dqm-pending** **pending-signatures** **tests-pending** **orp-pending** **comparison-pending** labels 4 hours ago

cmsbuild commented 4 hours ago — Owner

A new Pull Request was created by **@jmduarte** (Javier Duarte) for CMSSW_7_4_X.

RazorHbb HLT DQM for 74X

It involves the following packages:

HLTriggerOffline/SUSYBSM

@cmsbuild, **@danduggan**, **@vanbesien**, **@deguio** can you please review it and eventually sign? Thanks.
You can sign-off by replying to this message having '+1' in the first line of your reply.
You can reject by replying to this message having '-1' in the first line of your reply.
If you are a L2 or a release manager you can ask for tests by saying 'please test' in the first line of a comment.
@degano you are the release manager for this.
You can merge this pull request by typing 'merge' in the first line of your comment.

What we need – 3

- Build machines
  - with all the platforms we support
- Benedikt can tell us a few words ☺

Available Tools

- Main Continuous Integration (CI) tools:
  - Jenkis, Circle CI, Travis CI, …
- I have no idea about Jenkis or Circle CI
- Travis CI could be used
  - easy and free for open source projects
  - but build machines hosted by Travis…
    - VMs need to be set up
      - install the whole FCC software + dependencies everytime?
    - lack of flexibility?
- People welcome to investigate
- Decided to write a small CI tool in python

fcc-spi

Tools

- python
  - unittest
    - python's unit testing framework
  - networkx
    - graph module (test dependency)
- github API
  - pygithub?
  - direct use of the REST API?

the fcc-spi module

- core
  - core tools
    - stack: handles dependencies
    - call_script: calls bash scripts
- fcc_tests
  - fcc test suite
- example.spitest
  - test suite for test packages
- ~500 lines of code

Example Test Module

```python
import unittest
import os
import shutil
from core.call_script import call_script

import spitest_4
import spitest_5

workdir = 'spitest-3'

requirements = (spitest_4, spitest_5)

def install():
    script = '''
rm -rf {workdir}
git clone https://github.com/HEP-FCC-TEST/spitest-3 {workdir} 2> /dev/null
cd {workdir}
source exe.sh
'''.format(workdir=workdir)
    result, env = call_script(script)

def erase():
    shutil.rmtree(workdir)

class TestPackage1(unittest.TestCase):

    @classmethod
    def setUpClass(cls):
        install()

    def setUp(self):
        self.cwd = os.getcwd()
        os.chdir(workdir)

    def tearDown(self):
        os.chdir(self.cwd)

    def test_install(self):
        self.assertTrue(os.path.isfile('done.txt'))
        result, env = call_script('echo $SPITEST3')
        self.assertEqual(env['SPITEST3'], 'done')

if __name__ == '__main__':
    unittest.main()
```

install once for all tests

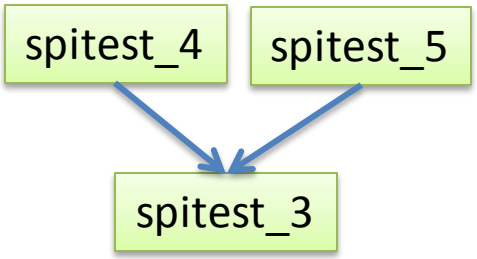
go to workdir before each test

go back after each test

only one test in this example

Needed:

- workdir:
  - directory where the package will be installed
- requirements:
  - dependencies



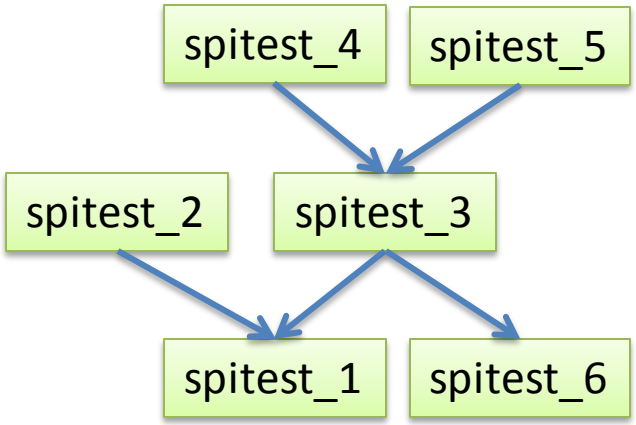
- install function

TODO:

- cloning should be done by the framework.
- handle branches

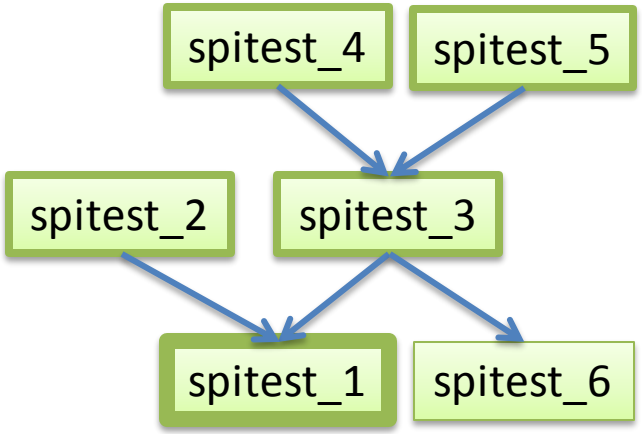
Stack of tests

```
import core

import spitest_1
import spitest_2
import spitest_3
import spitest_4
import spitest_5
import spitest_6

loader = unittest.TestLoader()
suite = unittest.TestSuite()

modules  = [spitest_1, spitest_2, spitest_3, spitest_4, spitest_5, spitest_6]

def validate(module):
    if not hasattr(module, 'workdir') and \
        module.workdir.is_instance(basestring):
        raise ValueError('blah')

map(validate, modules)

stack = core.Stack(modules)


def test(module):
    for dep in stack.depend_on_with_dependencies(module):
        tests = loader.loadTestsFromModule(dep)
        suite.addTests(tests)
    unittest.TextTestRunner(verbosity=2).run(suite)


if __name__ == '__main__':
    import sys
    modname = sys.argv[1]
    module = vars()[modname]
    print modname, 'modified'
    test(module)
```



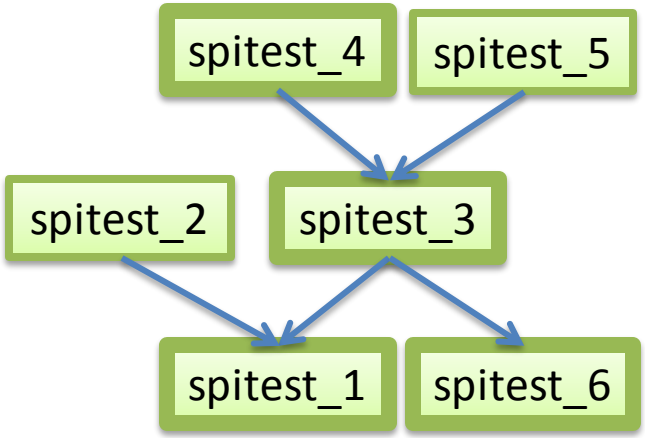
Stack of tests: spitest_1 modified

```
[spitest]$ python suite.py 'spitest_1'
spitest_1 modified
test_install (spitest_5.TestPackage1) ... ok
test_install (spitest_4.TestPackage1) ... ok
test_install (spitest_3.TestPackage1) ... ok
test_install (spitest_2.TestPackage1) ... ok
test_install (spitest_1.TestPackage1) ... ok
test_spitest2 (spitest_1.TestPackage1) ... ok
test_spitest3 (spitest_1.TestPackage1) ... ok

----------------------------------------------------------------------
Ran 7 tests in 6.752s

OK
```



Stack of tests: spitest_4 modified

```
[spitest]$ python suite.py 'spitest_4'
spitest_4 modified
test_install (spitest_4.TestPackage1) ... ok
test_install (spitest_5.TestPackage1) ... ok
test_install (spitest_3.TestPackage1) ... ok
test_install (spitest_6.TestPackage1) ... ok
test_install (spitest_2.TestPackage1) ... ok
test_install (spitest_1.TestPackage1) ... ok
test_spitest2 (spitest_1.TestPackage1) ... ok
test_spitest3 (spitest_1.TestPackage1) ... ok

----------------------------------------------------------------------
Ran 8 tests in 9.179s

OK
```



Todo:

- allow install without test when test becomes CPU intensive
- log all results (printouts, files, etc)

First FCC test

```
def install():
    script = '''
rm -rf {workdir}
git clone https://github.com/HEP-FCC/albers-core {workdir} 2> /dev/null
cd {workdir}
source init_macos.sh
mkdir build
cd build
cmake -DCMAKE_INSTALL_PREFIX=../install ..
make -j 4 install
cd ..
'''.format(workdir=workdir)
    result, env = call_script(script)

class TestAlbersCore(unittest.TestCase):

    @classmethod
    def setUpClass(cls):
        install()

    def setUp(self):
        self.cwd = os.getcwd()
        os.chdir(workdir)

    def tearDown(self):
        os.chdir(self.cwd)

    def test_1_write(self):
        s_write = '''
albers-write
'''
        result, env = call_script(s_write)
        self.assertTrue( os.path.isfile('example.root') )

    def test_2_read(self):
        s_read = '''
albers-read
'''
        result, env = call_script(s_read)
```

```
[fcc_tests]$ python suite.py albers_core
albers_core modified
/Users/cbernet/Code/FCC/fcc-spi/fcc_tests/albers-core/build/CMakeFiles
/Users/cbernet/Code/FCC/fcc-spi/fcc_tests/albers-core/install/cmake
CMake Warning (dev):
  Policy CMP0042 is not set: MACOSX_RPATH is enabled by default.  Run "cmake
  --help-policy CMP0042" for policy details.  Use the cmake_policy command t
  set the policy and suppress this warning.

  MACOSX_RPATH is not specified for the following targets:

   albers
   exampledatamodel

This warning is for project developers.  Use -Wno-dev to suppress it.

In file included from /Users/cbernet/Code/FCC/fcc-spi/fcc_tests/albers-core/
/Users/cbernet/Code/FCC/fcc-spi/fcc_tests/albers-core/albers/EventStore.
      field 'm_writer' is not used [-Wunused-private-field]
    Writer* m_writer;
            ^
1 warning generated.
test_1_write (albers_core.TestAlbersCore) ... ok
test_2_read (albers_core.TestAlbersCore) ... skipping bugged first event
ok
CMake Warning (dev):
  Policy CMP0042 is not set: MACOSX_RPATH is enabled by default.  Run "c
  --help-policy CMP0042" for policy details.  Use the cmake_policy comma
  set the policy and suppress this warning.

  MACOSX_RPATH is not specified for the following targets:

   datamodel
   utilities

This warning is for project developers.  Use -Wno-dev to suppress it.

In file included from /Users/cbernet/Code/FCC/fcc-spi/fcc_tests/fcc-edm/
:
/Users/cbernet/Code/FCC/fcc-spi/fcc_tests/fcc-edm/utilities/DummyGenerat
      field 'm_njets' is not used [-Wunused-private-field]
  unsigned m_njets;
           ^
1 warning generated.
test_1_write (fcc_edm.TestFCCEDM) ... ok
test_2_read (fcc_edm.TestFCCEDM) ... skipping bugged first event
ok

----------------------------------------------------------------------
Ran 4 tests in 25.614s

OK
```

TODO:

- suppress and log warnings

Github robot: Tasks

- look for new pull requests in all packages
  - every x minutes
- run the test suite on the new PR:
  - install dependencies
  - install package
    - merge PR locally
  - test package
  - test dependent packages
- comment on the PR page
  - green light or indications on what to do to fix thing
  - if green light : a human admin will still have to review and merge the PR

Github robot: which API?

- pygithub
  - can log to my account, browse my repositories, etc from a python script
  - but I don’t manage to read pull requests!
- REST interface
  - wrap it myself in python
  - should be easy
  - but not familiar with REST interfaces yet

TODO

- write the missing tests
  - FCCSW, pythiafcc, analysis-cpp, heppy, heppy_fcc
  - connect to documentation (twiki, README.md)
- suppress stderr printouts, save detailed logs, etc.
- manage multiple platforms
- move boilerplate code from tests to central modules
- github robot for full automatization
- set up on the build machines at CERN