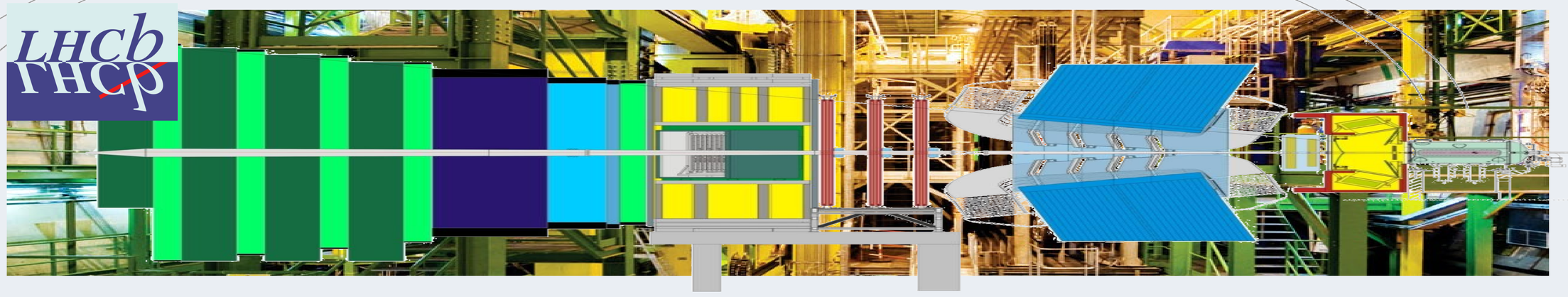

CERN Organisation
Europeenne pour la
Recherche Nucleaire
CH-1211 Geneve 23
Switzerland

Implementing a Domain Specific Language to configure and run LHCb Continuous Integration builds

M. Clemencic and B. Couturier on behalf of the LHCb Core Software Team

Abstract

The new LHCb nightly build system described at CHEP 2013 was limited by the use of JSON files for its configuration. JSON had been chosen as a temporary solution to maintain backward compatibility towards the old XML format by means of a translation function.
Modern languages like **Python** leverage on **meta-programming** techniques to enable the development of **Domain Specific Languages** (DSLs).
In this contribution we will present the advantages of such techniques and how they have been used to implement a DSL that can be used to both describe the configuration of the LHCb Nightly Builds and actually operate them.

The Use Case

LHCb Nightly Builds could be configured with XML and JSON.

```
<configuration>
    <general>...</general>
    <slot name="lhcb-head" use_cmake="true"
        description="head of everything against GAUDI_v26r1 and LCGCMT_74root6">
        <cmtprojectpath>
            <path value="/afs/cern.ch/lhcb/software/DEV/nightlies" />
            <path value="/afs/cern.ch/sw/Gaudi/releases" />
            <path value="/afs/cern.ch/sw/lcg/releases" />
            <path value="/afs/cern.ch/sw/lcg/app/releases" />
            <path value="/afs/cern.ch/lhcb/software/releases" />
        </cmtprojectpath>
        <platforms>
            <platform name="x86_64-slc6-gcc48-opt" />
            <platform name="x86_64-slc6-gcc48-dbg" />
            <platform name="x86_64-slc6-gcc49-opt" />
            <platform name="x86_64-slc6-gcc49-dbg" />
        </platforms>
        <projects>
            <project name="Gaudi" tag="GAUDI_v26r1" />
            <project name="Online" tag="ONLINE_HEAD"/>
            <project name="LHCb" tag="LHCB_HEAD">
                <change value="r183453" package="GaudiObjDesc" />
            </project>
            <project name="Lbcom" tag="LBCOM_HEAD" />
            <project name="Boole" tag="BOOLE_HEAD" />
            <project name="Rec" tag="REC_HEAD"/>
            <project name="Brunel" tag="BRUNEL_HEAD"/>
        </projects>
    </slot>
</configuration>
```

```
{"slot": "lhcb-head",
 "description": "head of everything against GAUDI_v26r1 and LCGCMT_74root6",
 "projects": [{"name": "Gaudi", "version": "v26r1"},
              {"name": "Online", "version": "HEAD"},
              {"name": "LHCb", "version": "HEAD",
               "overrides": {"GaudiObjDesc": "r183453"}},
              {"name": "Lbcom", "version": "HEAD"},
              {"name": "Boole", "version": "HEAD"},
              {"name": "Rec", "version": "HEAD"},
              {"name": "Brunel", "version": "HEAD"}],
 "default_platforms": ['x86_64-slc6-gcc48-opt",
                       "x86_64-slc6-gcc48-dbg",
                       "x86_64-slc6-gcc49-opt",
                       "x86_64-slc6-gcc49-dbg"],
 "USE_CMT": true,
 "env": [
   "CMTPROJECTPATH=/afs/cern.ch/lhcb/software/DEV/nightlies:/afs/cern.ch/sw/...
 ]
}
```

We needed something more flexible, easier to write and read, like a custom language, better tuned to our use case.

Python as a DSL engine

The Python programming language, thanks to metaclasses, decorators and descriptors, allows to write classes and functions that reduce a lot the need for boilerplate code, thus practically transforming it in an internal Domain Specific Language.

- **Descriptors**: member functions hidden behind object properties
- **Decorators**: wrap member functions to extend them (e.g. execute code before/after, modify passed arguments)
- **Metaclasses**: allow dynamic generation of code to automatically extend classes
- **Special methods**: methods used to provide the implementations of operators (e.g. +, -, [])

LHCb Nightly Builds New Configuration Language

Customization of Project behavior

Define Project prototypes

Custom defaults in Project prototypes

Define Slot prototypes

Full power of Python when needed

Instantiate configuration objects from basic building blocks

```
class Gaudi(Project):
    def checkout(self, **kwargs):
        Checkout.git('http://git.cern.ch/pub/gaudi.git',
                     self.version)

class LHCb(Project):
    pass

class LHCb_HEAD(LHCb):
    version = 'HEAD'
    overrides = {'GaudiObjDesc':'r183453'}

class CMakeSlot(Slot):
    buld_tool = 'cmake'

cmtprojectpath = ['/afs/cern.ch/lhcb/software/DEV/nightlies',
                  '/afs/cern.ch/sw/Gaudi/releases',
                  '/afs/cern.ch/sw/lcg/releases',
                  '/afs/cern.ch/sw/lcg/app/releases',
                  '/afs/cern.ch/lhcb/software/releases']

platforms = ['x86_64-slc6-%s-%s' % (i, j)
             for i in ('gcc48', 'gcc49')
             for j in ('opt', 'dbg')]

lhcb_head = CMakeSlot('lhcb-head',
                      description = 'head of everything against GAUDI_v26r1 '
                                    'and LCGCMT_74root6',
                      projects = [Gaudi('v26r1'),
                                  Project('Online', 'HEAD'),
                                  LHCb_HEAD(),
                                  Project('Lbcom', 'HEAD'),
                                  Project('Boole', 'HEAD'),
                                  Project('Rec', 'HEAD'),
                                  Project('Brunel', 'HEAD')],
                      env = ['CMTPROJECTPATH=%s' %
                             os.pathsep.join(cmtprojectpath)],
                      platforms = platforms)
```

Metaclasses:
- Use simple statements in class definition to trigger complex customization
- Context-enabled customizations (e.g. use the class name as name of the project)

Descriptors:
- Enable complex behavior on property assignment or retrieval

Decorators:
- Enrich methods and functions with pre/post actions

Special Methods:
- Implement rich semantics via operators

Configuration classes implement as well the core functionalities of the LHCb Nightly Build System (*checkout*, *build* and *test*):

Build and test the slot:

```
lhcb_head.checkout()
lhcb_head.build(jobs=8)
lhcb_head.test()
```

Build and test one project:

```
lhcb_head.Gaudi.checkout()
lhcb_head.Gaudi.build(jobs=8)
lhcb_head.Gaudi.test()
```

marco.clemencic@cern.ch

CHEP2015
OKINAWA, japan