Appendix: Concrete Examples

Anonymous Author(s)

ACM Reference Format:
Anonymous Author(s). 2022. Appendix: Concrete Examples. In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 3 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 EFFORT DUPLICATION (ED)

In this section we will present concrete examples for ED.

1.1 ED Example 1

This is simple ED example. `Variant1` representing the upstream–`nerdvegas/rez` and `variant2` representing the fork– `mottosso/bleeding-rez` have diverged between `2019-05-13` and `2020-11-06` (`latest_date`), adding `1,229` and `457` unique commits in the upstream and fork, respectively. One of the upstream's pull requests number `647`, has one commit–`25469bc` where one file `cmd.py` was changed to fix the `build break` for `issue #558`. The pull request was merged on `2019-06-22`.

Listing 1 is the code extract of the buggy lines for the buggy file `cmd.py`[1] at commit–`3a5afd8` (the commit before the one that patched the bug). Listing 2 is the code extract of the patched lines for the patched file[2] at commit–`25469bc`. Listing 3 contains the `diff` file[3] for the patch between the commit–`25469bc` that touched the file and its parent commit–`3a5afd8` (containing the buggy file) that contains one hunk. Listing 2 shows the fixed lines of the patched file. Listing 4 is the code extract of the patched lines for the patched file `cmd.py`[4] at the `latest_commit–4a723fb` in the target repository.

Listing 1: Buggy lines in file `cmd.py` at commit `3a5afd8` of the source repository

```
1       # http://ss64.com/nt/syntax-esc.html
2       _escape_re = re.compile(r'(?<!\^)[&<>]|(?<!\^)\^(?![&<>\^])')
3       _escaper = partial(_escape_re.sub, lambda m: '^' + m.group(0))
```

Listing 2: Patched lines in `cmd.py` at commit `25469bc` of the source repository

```
1       # http://ss64.com/nt/syntax-esc.html
2       _escape_re = re.compile(r'(?<!\^)[&<>]|(?<!\^)\^(?![&<>\^])|(\|)')
3       _escaper = partial(_escape_re.sub, lambda m: '^' + m.group(0))
```

[1]buggy_file in source (ED1): https://raw.githubusercontent.com/nerdvegas/rez/-3a5afd8/src/rezplugins/shell/cmd.py
[2]Patched file in source (ED1): https://raw.githubusercontent.com/nerdvegas/rez-/25469bc/src/rezplugins/shell/cmd.py
[3]diff_file (ED): https://patch-diff.githubusercontent.com/raw/nerdvegas/rez-/pull/647.patch
[4]latest_commit file in target (ED1): https://raw.githubusercontent.com/mottosso/-bleeding-rez/4a723fb/src/rezplugins/shell/cmd.py

Conference'17, July 2017, Washington, DC, USA

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

Listing 3: diff file for PR–647 from the source repository

```
1       @@ -24,7 +24,7 @@
2
3        # http://ss64.com/nt/syntax-esc.html
4   -    _escape_re = re.compile(r'(?<!\^)[&<>]|(?<!\^)\^(?![&<>\^])')
5   +    _escape_re = re.compile(r'(?<!\^)[&<>]|(?<!\^)\^(?![&<>\^])|(\|)')
6        _escaper = partial(_escape_re.sub, lambda m: '^' + m.group(0))
```

Listing 4: patched lines in file `cmd.py` in the `latest_commit–4a723fb` of target repository

```
1       # http://ss64.com/nt/syntax-esc.html
2       _escape_re = re.compile(r'(?<!\^)[&<>]|(?<!\^)\^(?![&<>\^])|(\|)')
3       _escaper = partial(_escape_re.sub, lambda m: '^' + m.group(0))
```

We can see that Line 2 in Listing 2 is identical to Line 5 that is added in the patch of the upstream in Listing 9 and Line 2 of the patched file in Listing 4. Such a case in the target repository can be classified as effort duplication, because the patch that is applied to the upstream is also applied to the divergent fork.

1.2 ED Example 2

This example has a bigger patch and is an effort duplication case in the upstream. This patch originates in the fork and is duplicated in the upstream.

`Variant1` representing the fork–`BlueBrain/spack` and `variant2` representing the upstream–`spack/spack` have diverged between `2018-03-12` and `2021-07-30` (`current_date`), adding `521` and `24` unique commits in the upstream and divergent fork, respectively. One of the divergent fork's pull requests number `1243`, has one commit `7d8f420` where one file `python.py` was changed for `hotfix: undo our modifications to the Python build system` for `issue #1243`. The pull request was merged on `2021-07-30`.

Listings 5 and 6 are the code extracts of the buggy lines for the two hunks from the buggy file `python.py` [5] at commit–`79a4051` (the commit before the one that patched the bug). Listing 7 and 8 are the code extract of the patched lines for the two hunks from the patched file [6] at commit–`7d8f420`. Listing 9 contains the `diff` file between the commit `7d8f420` that last touched the file and its parent commit `7d8f420` (containing the buggy file) that contains two hunks. Listings 10 and 11 are the code extract of the patched lines for the two hunks from the `latest_commit` file `python.py` [7] at the `latest_commit–01216a0` in the target repository.

Listing 5: Buggy lines for hunk 1 in file `python.py` at commit `79a4051` of the source repository

```
1           self.setup_py('install', *args)
2
3       def install_args(self, spec, prefix):
4           """Arguments to pass to install."""
5           python_version = self.spec['python'].version.up_to(2)
6           python_string = 'python{0}'.format(python_version)
7           path_to_install_lib = os.path.join(
8               prefix, 'lib', python_string, 'site-packages')
9
10          args = ['--prefix={0}'.format(prefix),
```

[5]buggy_file in source (ED2): https://raw.githubusercontent.com/BlueBrain/spack/79a4051/lib/spack/spack/build_systems/python.py
[6]patched_file in source (ED2): https://raw.githubusercontent.com/BlueBrain/spack/7d8f420/lib/spack/spack/build_systems/python.py
[7]latest_commit file in target (ED2): https://raw.githubusercontent.com/spack/spack/d02d683/lib/spack/spack/build_systems/python.py

```
11                  '--install-lib={0}'.format(path_to_install_lib)]
12
13          if ('py-setuptools' == spec.name or          # this is setuptools,
                or
14              'py-setuptools' in spec._dependencies and  # it's an immediat
```

Listing 6: Buggy lines for hunk 2 in file python.py at commit 79a4051 of the source repository

```
1           args += ['--root=%s' % prefix,
2                    '--install-purelib=%s' % pure_site_packages_dir,
3                    '--install-platlib=%s' % plat_site_packages_dir,
4                    '--install-scripts=bin',
5                    '--install-data="",
6                    '--install-headers=%s' % inc_dir
7                    ]
8
9           return args
```

Listing 7: Patched lines for hunk 1 in file python.py at commit 7d8f420 of the source repository

```
1   self.setup_py('install', *args)
2
3       def install_args(self, spec, prefix):
4           """Arguments to pass to install."""
5           args = ['--prefix={0}'.format(prefix)]
6
7           if ('py-setuptools' == spec.name or          # this is setuptools,
                or
8               'py-setuptools' in spec._dependencies and  # it's an immediate
                    dep
9               'build' in spec._dependencies['py-setuptools'].deptypes):
```

Listing 8: Patched lines for hunk 2 in file python.py at commit 7d8f420 of the source repository

```
1           args += ['--root=%s' % prefix,
2                    '--install-purelib=%s' % pure_site_packages_dir,
3                    '--install-platlib=%s' % plat_site_packages_dir,
4                    '--install-scripts=bin',
5                    '--install-data=',
6                    '--install-headers=%s' % inc_dir
7                    ]
8
9           return args
```

Listing 9: Diff file for PR–1243 from the source repository

```
1   @@ -212,13 +212,7 @@
2
3        def install_args(self, spec, prefix):
4            """Arguments to pass to install."""
5   -        python_version = self.spec['python'].version.up_to(2)
6   -        python_string = 'python{0}'.format(python_version)
7   -        path_to_install_lib = os.path.join(
8   -            prefix, 'lib', python_string, 'site-packages')
9   -
10  -        args = ['--prefix={0}'.format(prefix),
11  -                '--install-lib={0}'.format(path_to_install_lib)]
12  +        args = ['--prefix={0}'.format(prefix)]
13
14          # This option causes python packages (including setuptools) NOT
15          # to create eggs or easy-install.pth files.  Instead, they
16  @@ -258,7 +252,7 @@
17                       '--install-purelib=%s' % pure_site_packages_dir,
18                       '--install-platlib=%s' % plat_site_packages_dir,
19                       '--install-scripts=bin',
20  -                    '--install-data="",
21  +                    '--install-data=',
22                       '--install-headers=%s' % inc_dir
23                       ]
```

Listing 10: Patched lines for hunk 1 in file python.py in the latest_commit–01216a0 of target repository

```
1       def install_args(self, spec, prefix):
2           """Arguments to pass to install."""
3           args = ['--prefix={0}'.format(prefix)]
4
5           if ('py-setuptools' == spec.name or          # this is setuptools,
                or
6               'py-setuptools' in spec._dependencies and  # it's an immediate
                    dep
```

Listing 11: Patched lines for hunk 2 in file python.py in the latest_commit–01216a0 of target repository

```
1           args += ['--root=%s' % prefix,
2                    '--install-purelib=%s' % pure_site_packages_dir,
3                    '--install-platlib=%s' % plat_site_packages_dir,
4                    '--install-scripts=bin',
5                    '--install-data=',
6                    '--install-headers=%s' % inc_dir
7                    ]
8
9           return args
10
11      def install_lib(self, spec, prefix):
```

For hunk 1 we can see that Line 3 in Listing 10 is identical to Line 12 that is added in the patch of the upstream in Listing 9 and Line 5 of the patched file in Listing 7. The removed Lines 5-11 in Listing 9 is not found in the `latest_commit` file. This hunk is a case of ED.

For hunk 2 we can see that Line 5 in Listing 10 is identical to Line 21 that is added in the patch of the upstream in Listing 9 and Line 5 that of the patched file in Listing 8. The removed Line 20 in Listing 9 is not found in the `latest_commit` file. This hunk is a case of ED as well.

Both hunks from the patch are classified as ED, thus the file and the patch in the target repository can also be classified as effort duplication (ED), because the patch that is applied to the fork is also applied to the upstream.

2 SPLIT (SP)

In this section we will present a concrete example for SP.

`Variant1` representing the upstream–`datastax/python-driver` and `variant2` divergent fork–`YugaByte/cassandra-python-driver` have diverged between `2018-04-06` and `2020-11-17` (`current_date`), adding only `39` unique commits in the upstream. One of the upstream's pull request number `1095`, has one commit `dc3f2f8` where two files `CHANGELOG.rst` and `asyncorereactor.py` are changed for the fix `"PYTHON-1266: Fix asyncore race condition cause logging exception on shutdown"` for `issue #1095`. The pull request was merged on `2020-10-15`. The file `CHANGELOG.rst` is a written in a language that the tool cannot classify, but the file `asyncorereactor.py` can be classified.

Listings 12 and 13 are the code extracts of the buggy lines for the two hunks from the buggy file `asyncorereactor.py` [8] at commit–`653ef97` (the commit before the one that patched the bug). Listing 14 and 15 are the code extract of the patched lines for the two hunks from the patched file [9] at commit–`dc3f2f8`. Listing 16 contains the `diff` file between the commit `dc3f2f8` that last touched the file and its parent commit `653ef97` (containing the buggy file) that contains two hunks. Listings 17 and 18 are the code extract of the patched lines for the two hunks from the `latest_commit` file `asyncorereactor.py` [10] at the `latest_commit–d343e65` in the target repository.

[8] buggy_file in source (SP): https://raw.githubusercontent.com/datastax/python-driver/653ef97/cassandra/io/asyncorereactor.py
[9] patched_file in source (SP): https://raw.githubusercontent.com/datastax/python-driver/dc3f2f8/cassandra/io/asyncorereactor.py
[10] latest_commit file in target(SP): https://raw.githubusercontent.com/YugaByte/cassandra-python-driver/d343e65/cassandra/io/asyncorereactor.py

Listing 12: Buggy lines for hunk 1 in file `asyncorereactor.py` at commit `79a4051` of the source repository

```
from cassandra.connection import Connection, ConnectionShutdown,
        NONBLOCKING, Timer, TimerManager


# TODO: Remove when Python 2 is removed
class LogWrapper(object):
    """ PYTHON-1228. If our logger has disappeared, there's nothing we can
            do, so just execute nothing """
    def __init__(self):
        self._log = logging.getLogger(__name__)

    def __getattr__(self, name):
        try:
            return getattr(self._log, name)
        except:
            return lambda *args, **kwargs: None


log = LogWrapper()

_dispatcher_map = {}
```

Listing 13: Buggy lines for hunk 2 in file `asyncorereactor.py` at commit `79a4051` of the source repository

```
                while not self._shutdown:
                    try:
                        self._loop_dispatcher.loop(self.timer_resolution)
                        self._timers.service_timeouts()
                    except Exception:
                        log.debug("Asyncore event loop stopped unexepectedly",
                                  exc_info=True)
                        break
                self._started = False

            log.debug("Asyncore event loop ended")
```

Listing 14: Patched lines for hunk 1 in file `asyncorereactor.py` at commit `7d8f420` of the source repository

```
except ImportError:
    from cassandra.util import WeakSet  # noqa

import asyncore

from cassandra.connection import Connection, ConnectionShutdown,
        NONBLOCKING, Timer, TimerManager


log = logging.getLogger(__name__)

_dispatcher_map = {}

def _cleanup(loop):
```

Listing 15: Patched lines for hunk 2 in file `asyncorereactor.py` at commit `7d8f420` of the source repository

```
                    try:
                        self._loop_dispatcher.loop(self.timer_resolution)
                        self._timers.service_timeouts()
                    except Exception:
                        try:
                            log.debug("Asyncore event loop stopped unexepectedly
                                      ", exc_info=True)
                        except Exception:
                            # TODO: Remove when Python 2 support is removed
                            # PYTHON-1266. If our logger has disappeared, there
                                    's nothing we
                            # can do, so just log nothing.
                            pass
                        break
                self._started = False

            log.debug("Asyncore event loop ended")
```

Listing 16: Diff file for PR-1243 from the source repository

```
@@ -36,20 +36,7 @@
  from cassandra.connection import Connection, ConnectionShutdown,
          NONBLOCKING, Timer, TimerManager


- # TODO: Remove when Python 2 is removed
- class LogWrapper(object):
-     """ PYTHON-1228. If our logger has disappeared, there's nothing we can
-         do, so just execute nothing """
-     def __init__(self):
-         self._log = logging.getLogger(__name__)
-
-     def __getattr__(self, name):
-         try:
-             return getattr(self._log, name)
-         except:
-             return lambda *args, **kwargs: None
-
-
- log = LogWrapper()
+ log = logging.getLogger(__name__)

  _dispatcher_map = {}

@@ -262,7 +249,13 @@
                      self._loop_dispatcher.loop(self.timer_resolution)
                      self._timers.service_timeouts()
                  except Exception:
-                     log.debug("Asyncore event loop stopped unexepectedly",
-                               exc_info=True)
+                     try:
+                         log.debug("Asyncore event loop stopped
+                                   unexpectedly", exc_info=True)
+                     except Exception:
+                         # TODO: Remove when Python 2 support is removed
+                         # PYTHON-1266. If our logger has disappeared,
+                                 there's nothing we
+                         # can do, so just log nothing.
+                         pass
                      break
              self._started = False
```

Listing 17: Patched lines for hunk 1 in file `asyncorereactor.py` in the `latest_commit-01216a0` of target repository

```
except ImportError:
    ssl = None  # NOQA

from cassandra.connection import Connection, ConnectionShutdown,
        NONBLOCKING, Timer, TimerManager

log = logging.getLogger(__name__)

_dispatcher_map = {}

def _cleanup(loop_weakref):
```

Listing 18: Patched lines for hunk 2 in file `asyncorereactor.py` in the `latest_commit-01216a0` of target repository

```
                while not self._shutdown:
                    try:
                        self._loop_dispatcher.loop(self.timer_resolution)
                        self._timers.service_timeouts()
                    except Exception:
                        log.debug("Asyncore event loop stopped unexepectedly",
                                  exc_info=True)
                        break
                self._started = False

            log.debug("Asyncore event loop ended")
```

For hunk 1 we can see that Line 6 in Listing 17 is identical to Line 19 that is added in the patch of the upstream in Listing 16 and Line 9 of the patched file in Listing 7. The removed Lines 5-18 in Listing 16 is not found in the `latest_commit` file. This hunk is case of effort duplication.

For hunk 2 we can see that Line 6 in Listing 17 is identical to Line 27 that is removed in the patch of the upstream in Listing 16 and Line 6 of the buggy file in Listing 13. The added Lines 28-34 of in Listing 16 is not found in the `latest_commit` file. This hunk is a case of missed opportunity.

The file `asyncorereactor.py` is classified as a split because the patch that is applied to the upstream is partially applied to the divergent fork. The patch in its whole is also classified as SP since one file has an interesting class, we classify the patch as SP as well.