

AMITY UNIVERSITY
UTTAR PRADESH

AMITY SCHOOL OF ENGINEERING & TECHNOLOGY

PROGRESS REPORT

B. Tech (CSE)

Project Title: Linter for Enforcing Downstream Checks

Academic Session: 2020-21

Project Guide: Dr. Pooja Singh

Group Number: 85

Section: 8CSE8

S.No	Enrolment no.	Name	Signature
1	A2305217638	Samyak Jain	<i>Samyak</i>
2	A2305217557	Utkarsh Gupta	<i>Utkarsh</i>

Progress made so far:

We've made good progress; the 3 “cops” that we've worked on so far are:

1. The problem encountered while checking for git files in the gemspec, manually.

Avoid using `git ls-files` to produce lists of files. Downstream (OS packagers) often need to build your package in an environment that does not have git (on purpose). Packages in Debian are built in a clean environment ([sbuild](#), [schroot](#), et al), and whilst doing so, the build fails with:

```
Invalid gemspec in [<gem_name>.gemspec]: No such file or directory - git
```

2. The problem encountered while using `require_relative`.

Avoid using `require_relative` with the relative path from your test code (spec/ or tests/) to lib/. Use `require` instead.

Using `require_relative` from the tests into the `lib` directory makes that impossible, but it also makes the test look less like a client-code that would use the code in your gem. Therefore, we recommend that the test code uses the main library code without `require_relative`.

Therefore, when one uses a relative path, we end up getting a `LoadError`, stating:

```
cannot load such file - /[PKGBUILDDIR]/foo.
```

3. The third cop deals with the “require” calls. When using “require” with relative path, it results in the same problem which has been described above. Therefore, it’d be in best interest to use the absolute path, which is not even hard to pull off.

As a quick example,

Instead of using:

```
require “../lib/foo/bar”
```

Consider using:

```
require “foo/bar”
```

This is much shorter, neater, and helps fix our problem as well. So a win-win, indeed.

A quick overview from each week:

Weeks	Samyak Jain	Utkarsh Gupta
1	Initializing with the problem regarding usage of git-ls files.	Initializing with the problem regarding usage of <code>require_relative</code> .
2	Checking for git files in the gemspec, manually. Starting the documentation for the same.	Omitting the usage of <code>require_relative</code> , manually. Starting the documentation for the same.
3	Workaround to the git files: Avoid using git ls-files to produce lists of files. Downstreams (OS packagers) often need to build your package in an environment that does not have git (on purpose). Instead, use some pure	Workaround to the <code>require_relative</code> files: Use <code>require</code> instead. Using <code>require_relative</code> from the tests into the <code>lib</code> directory makes that impossible, but it also makes the test look less like client-code that would use the code in your gem.

	Ruby alternatives, like Dir or Dir.glob.	Therefore, we recommend that test code uses the main library code without <code>require_relative</code> .
4	<p>Initializing git-ls file cop under the parent directory structure of rubocop-packaging.</p> <p>Constant testing for both the scenarios to collect all the possible errors.</p>	<p>Initializing <code>require_relative</code> cop under the parent directory structure of rubocop-packaging.</p> <p>Constant testing for both the scenarios to collect all the possible errors.</p>
5	<p>The documentation regarding the problem is properly written keeping best practices in mind.</p> <p>Drafting a style guide to mention all the workarounds and achievements.</p>	<p>The documentation regarding the problem is properly written keeping best practices in mind.</p> <p>Drafting a style guide to mention all the workarounds and achievements.</p>
6	<p>Started writing AST formation for the problem.</p> <p>AST for the cop is almost under process and might be finished by next week.</p>	<p>Started writing AST formation for the problem.</p> <p>AST for the cop is almost under process and might be finished by next week.</p>
7	The AST with regards to git files problems is complete now and is tested properly	The AST with regards to <code>require_relative</code> problems is complete now and is tested properly. Needs a little more tweaking due to heavy usage of <code>require_relative</code> .
8	For the proper training purposes, the linter was run across 100s of packages pertaining to the problem.	For the proper training purposes, the linter was run across 100s of packages pertaining to the problem.
9	Taking a deeper look at how the internals of the “require” function work in Ruby.	Initializing the solution of “usage of <code>require</code> with relative path”.

10	Checking and testing for plausible false-positives and AST formation; clubbing multiple ASTs could result in a cumbersome result.	Writing the (proposed) AST for such calls; since there are multiple ways of invoking a “require” call, try to catch as many as possible.
11	Writing tests for this cop as there could be multiple ways to invoke and catching them all should be the goal of this cop.	Writing the internal code (lib/) for the problem, using and importing the helper functions from rubocop/packaging/lib_helper_module.

- ❖ The cops are now in functional mode and are being tested across various ruby gems, some of the packages that were tested are as follows:

```

1. adlint*
2. asciiart*
3. asciidoctor*
4. atig*
5. batalert*
6. berkshelf-api*
7. bsfilter*
8. bundler*
9. camping*
10. capistrano*
11. chake*
12. chef*
13. chef-zero*
14. coderay*
15. ctioga2[U][O]
16. cucumber*
17. diaspora*
18. diaspora-installer*
19. dnsruby*
20. facterdb*
21. feed2imap*
22. foodcritic*
23. foremancli*
24. gem2deb*
25. gist*
26. gitlab-shell*
27. haproxyctl*
28. hiera*
29. hiera-eyaml*
30. hiki[U]
31. homesick*

```

```

class GemspecGit < Base
  # This is the message that will be displayed when RuboCop finds an
  # offense of using `git ls-files`.
  MSG = "Avoid using git to produce lists of files. " \
        "Downstreams often need to build your package in an environment " \
        "that does not have git (on purpose). " \
        "Use some pure Ruby alternative, like `Dir` or `Dir.glob`."

  def_node_search :xstr, <<~PATTERN
    (block
      (send
        (const
          (const {cbase nil?} :Gem) :Specification) :new)
        (args
          (arg _) `$(xstr (str #starts_with_git?)))
        )
      )
    )
  PATTERN
end

```

AST Regarding git-ls files in gemspec

```

class RequireRelativeHardcodingLib < Base
  include RuboCop::Packaging::LibHelperModule
  extend AutoCorrector

  # This is the message that will be displayed when RuboCop finds an
  # offense of using `require_relative` with relative path to lib.
  MSG = "Avoid using `require_relative` with relative path to lib. " \
        "Use `require` with absolute path instead."

  def_node_matcher :require_relative, <<~PATTERN
    (send nil? :require_relative
      (str #falls_in_lib?))
    )
  PATTERN
end

```

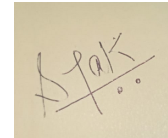
32. imagetooth*
33. itamae*
34. jekyll*
35. jekyll-theme-minima*
36. jgrep*
37. kwalify[U]
38. larch*
39. librarian-puppet*
40. lolcat*
41. metadata-json-lint*
42. mikutter*
43. mkalias*
44. nadoka*
45. nanoc*
46. ohai*
47. open-build-service*
48. origami-pdf*
49. passenger*

AST Regarding in require_relative

In the future, we are looking forward to testing more packages against the above implementation and implementing more solutions for newer problems that are being encountered.



SIGNATURE OF GUIDE



SIGNATURE