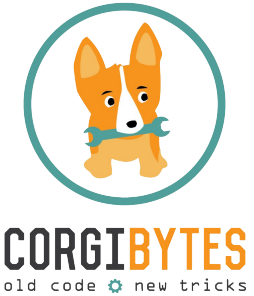
Writing Maintainable Rails Applications

Chris Cumming

Who am I? (i.e. Shameless Self Promotion)

Software Consultant

https://saturdaymp.com
chris.cumming@satudaymp.com



Virtual Chat Host

http://weeklydevchat.com/
https://www.legacycode.rocks/

Slacks

Chris C on

- Dev Edmonton (https://devedmonton.com/)
- Legacy Code Rocks (https://www.legacycode.rocks/)
- YegSec (https://www.yegsec.ca/)

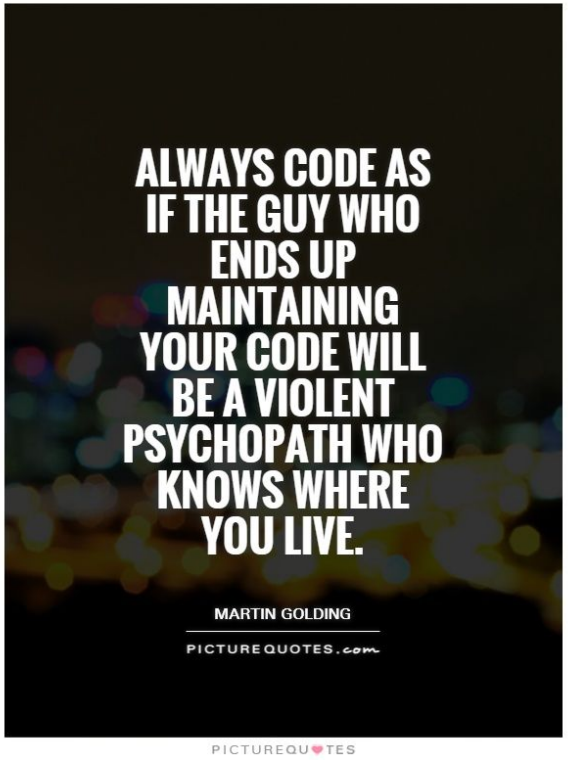
What is Maintainable Software?

"The ease with which a software system or component can be modified to correct faults, improve performance or other attributes, or adapt to a changed environment."

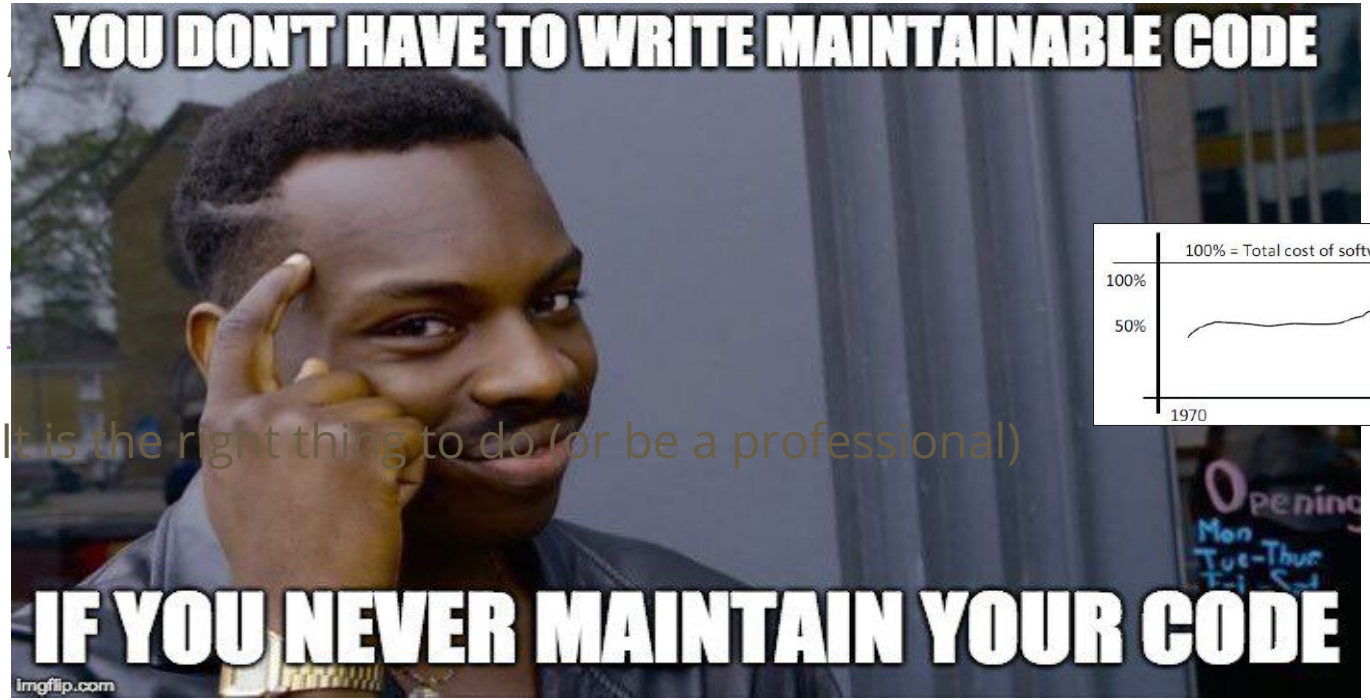
- IEEE

“Would you be willing to be paid by the release but have your payment reduced for every issue found?”

- Chris



Why Write Maintainable Applications?



It is the right thing to do (or be a professional)

Traits of Easy to Maintain Application

Well written and architected

Good test coverage

Uses guardrails

Problems are found early and are hard to ignore

Explicit

Feel confident making changes

Well Written and Architected

Rails has well defined file locations and naming scheme, follow them.

Rails has well known idioms, use them.

Keep MVC logic in the correct location.

Avoid fat controllers and models.

Don’t be afraid to put logic in non-controller/model classes.

Don’t do queries or complicated logic in views.

Good Test Coverage

Easy to write and run tests in Rails, no excuse not too.

Monitor test coverage (e.g. SimpleCov) percentage in CI and fail build if it goes down.

Don’t be afraid to test views, partials, routes, etc.

Only use Mocks as a last resort. VCR can mock 3rd party API calls.

Have both unit and system tests.

Uses Guardrails

Ruby and Rails does not have many guardrails, you need to add them.

Use a linter like Standard or RuboCop. Fail build if linter finds an issue.

Use a type checker like Sorbet. Fail build type checker finds an issue.

Use IDE code inspections.

PR reviews.

Problems are Found Early and are Hard to Ignore

Run linter, type checker, tests, etc before committing code.

Visual diff your changes before committing.

CI should run linter, type checker, tests, test coverage, etc. Should notify developer if a check fails.

Don’t do PR reviews until all checks pass.

Explicit

Ruby on Rails is not great at being explicit.

Use Sorbet, or other type checker.

Avoid metaprogramming (defining methods on the fly) and duck typing.

Systematically remove code.
https://thepugautomatic.com/2020/11/systematically-removing-code/

Use tools to find dead code.
https://www.hexdevs.com/posts/how-to-remove-dead-code-ruby/

Use Comments.

Feel Confident Making Changes

Do you feel comfortable making the change?

Will take time to build up your confidence.

You will never catch all issues. Add additional checks for bugs that slip through the cracks.

Questions?

Email:

chris.cumming@satudaymp.com

Slacks

Chris C on

- Dev Edmonton (https://devedmonton.com/)
- Legacy Code Rocks (https://www.legacycode.rocks/)
- YegSec (https://www.yegsec.ca/)

SATURDAY MORNING PRODUCTIONS

WEEKLY DEV CHAT

CORGIBYTES

old code ⚙ new tricks

Resources/References - 1

Buggy Code: 10 Common Rails Programming Mistakes
https://www.toptal.com/ruby-on-rails/top-10-mistakes-that-rails-programmers-make

SimpleCov
https://github.com/simplecov-ruby/simplecov

RSpec
https://rspec.info/

Sorbet
https://sorbet.org/

RubyMine
https://www.jetbrains.com/ruby/

Resources/References - 2

Systematically remove code.
https://thepugautomatic.com/2020/11/systematically-removing-code/

Use tools to find dead code.
https://www.hexdevs.com/posts/how-to-remove-dead-code-ruby/

Standard:
https://github.com/testdouble/standard