Effective Code Reviews: The edge between hard and soft skills

Vinícius Gubiani Ferreira
Vinícius Gubianí Ferreira
"A software QA activity in which one or several people check a program mainly by viewing and reading parts of its source code, and they do so after implementation or as an interruption of implementation. At least one of the persons must not be the code's author.

Wikipedia
Once upon a time ...
Often decides to be brutally honest
Due to that, had to find a new job
Or worst: keep his opinions for himself
Code reviews? Why?

- Knowledge Transmission
- Stimulate collective collaboration in projects
- Quick feedback / ensure changes are on the right track
- Stimulate contribution for new PRs and good practices
- Ensure we have good quality on the code product
3 Golden rules for healthy code reviews
Rule number 1:

Don't take any comment as a personal offense
Rule number 2:

Listen to feedbacks
Rule number 3:

Accept the fact that you might be wrong. Make mistakes is part of being human.
Good practices - Positive feedback

- Good job! / Awesome
- LGTM, IMHO, ...
- Use emojis, memes, ...
Examples
vinigfer on Oct 3

Maybe just `function` instead of `function_choice`? Seems a bit redundant this line. This is how I feel when I see lines like this 😊

```
21 <button type="button" id="btnButton">Button</button>
```

It be like that all the time
@ @ -308,7 +308,7 @@ def test_invalid_update_payment_command(self, mock_set
...

        result = self.handler.update_payment_method(update_payment_command)

-        self.assertEqual(result.data, None)
+        self.assertEqual(result.data, result.data)
you commented tests...

FIX YOUR CODE TO PASS UNIT TESTS

FIX YOUR UNIT TESTS TO PASS YOUR CODE
How to check if a PR is good? Which metrics should we use?
Code Quality Measurement:
WTFs/Minute

Good Code

Bad Code

WTF

WTF is this shit?

Dude, WTF

http://commadot.com
Good practices - Negative feedback

When you reject a PR, _always_ explain the reason you are doing so.
General rules about comments (especially in negative feedback)

- Be nice always! Have empathy with the PR author.
- Use collective words (us, can we, all) instead of individual terms (I, you, him).
- Be clear and straight to the point (but still being nice and respecting your fellow teammates).
- Raise questions: What if we do it this way? Looks more efficient because X, Y, Z.
Checklist - During development

- Isn't your PR getting too much big? Is it possible to break it into smaller PRs, easier/faster to review?
- Can I open a PR as a WIP?
- Place style changes (Black/PEP8/iSort) into separate commits
Checklist - Before opening a PR

- Did the tests passed on your machine?
- Is the PR title clear? What about the description? How about adding a print if a screen was added/changed?
- Is it a new feat or fix? Where are the new tests?
- Review your own PR carefully, as you would review somebody else's PR. Or as I like to think of ...
The next programmer is a psycho who know where you live
Reviewing PRs: What to look for?

- Tests or pipeline are failing: don't waste your time
- Typos: Comments, tests, variable names, classes, ...
- Words in another language that are not english
- Logic errors
- Performance optimizations
Reviewing PRs: What to look for?

- Follow style conventions, project architectural guidelines, and good practices adopted by the company and the community
- Commit messages / PR title
- The author doesn't give up: negotiate!
- Don't use push force! Use --force-with-lease
To sum it up

Code review comes down to people: find a way to express yourself and negotiate about tech subjects, without harming anybody.
Move to the Edge 🚀

Valeu! / Merci! / Thank you! / Gracias! / Vielen Dank! / Спасибо! / 谢谢啦! / ありがとう!
Have a question? Please contact me!

#talk-effective-code-reviews

vinigfer

vinicius-gubianiferreira

vini.g.fer@gmail.com