Collaborative software development | Part 2

Lesson 4 of the workshop 'Version control and collaborative development for research Software"

Delft University of Technology, 07-10 October 2024

A course developed and delivered by Manuel Garcia Alvarez and Giordano Lipari

Lesson 4

3|0 Recap collaborative Workflows
3|1 Code Reviews
3|2 Contributing Guidelines
3|3 Other best practices

Episode
More on Collaborative Workflows

(Team) Exercise 1: Implement collaborative workflow [15 min]

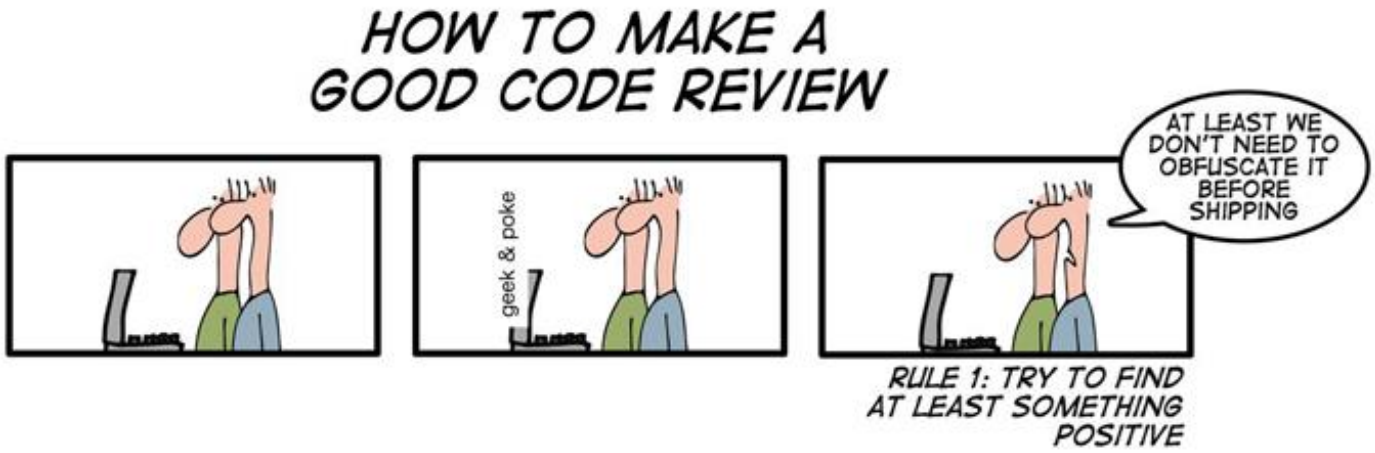
Working in teams, define and implement a workflow of your choice to collaborate in a repository.

1. [Administrator/Owner] create a repository for the team using the template: https://github.com/WorkshopGitcodev/collab-review
2. [Team] discuss and agree on which workflow to implement for this exercise.
3. [Team] each member choses one task from the `faircode-checklist.md` and make changes.
4. [Team] each member opens a pull request to the `main` branch with their changes. **Important: Do not merge**
5. [Team] each member makes some other changes to the repository, commits and push.
6. [Team] go back to your pull request and see how the latest changes affected your pull request. **Do not merge.**

Questions?

Episode

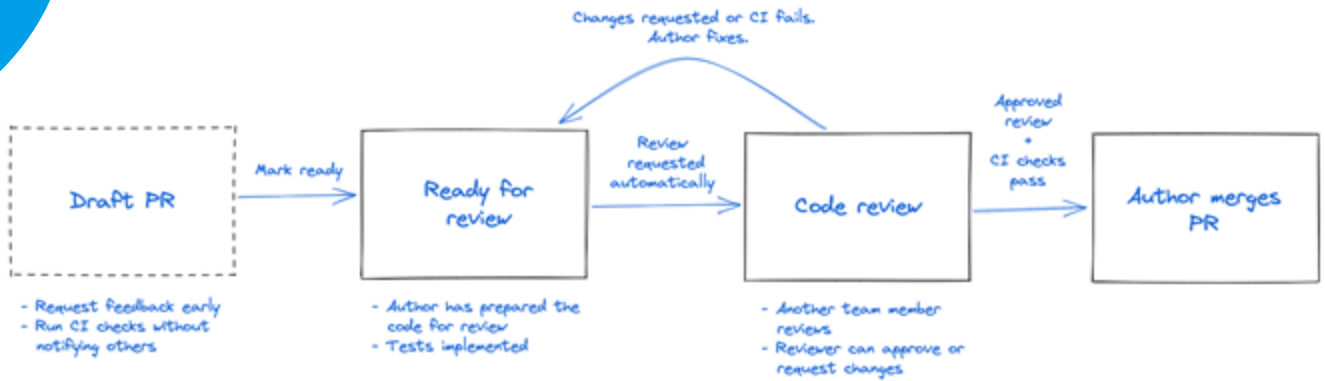
Managing Collaboration: Code Reviews


HOW TO MAKE A GOOD CODE REVIEW
geek & poke
AT LEAST WE DON'T NEED TO OBFUSCATE IT BEFORE SHIPPING
RULE 1: TRY TO FIND AT LEAST SOMETHING POSITIVE

Source: shorturl.at/jpsz7

Code Review

Code review is a step on a workflow of collaborative software development. Therefore, deciding on the overall process is key.



Code Review (Roles)

Code Review: What to focus on?

- Functionality: Does the code behave as the PR author likely intended? Does the code behave as users would expect?
- Software design: Is the code well-designed and fitted to the surrounding architecture?
- Complexity: Would another developer be able to easily understand and use the code?
- Tests: Does the PR have correct and well-designed automated tests?
- Naming: Are names for variables, functions, etc. descriptive?
- Comments: Are the comments clear and useful?
- Documentation: Did the author also update relevant documentation?

Source: shorturl.at/jpsz7

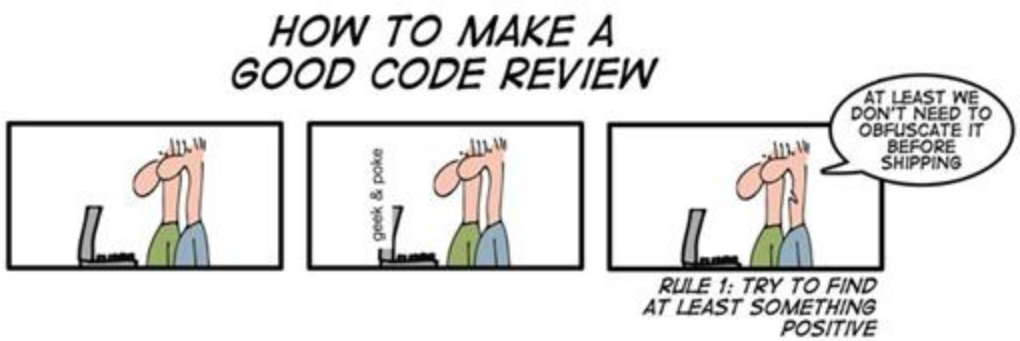
Code Review: Giving feedback

- Give feedback about the code, not about the author.
- Pick your battles.
- Accept that there are multiple correct solutions to a problem.
- You're in the same boat.
- PR authors are humans with feelings (except dependabot ).
- Provide reasons, not feelings, to support your position.
- Use the "Yes, and..." technique to keep an innovative atmosphere. It can be an ungracious pattern to dismiss fresh and fragile ideas in a draft PR stage.
- Keep the feedback balanced with positive comments. It's always delightful to receive praise from the reviewer (but don’t over do it).

Source: shorturl.at/jpsz7

Code Review: Explicit communication

- When reviewing a piece of code, be explicit about the action you request from the author. GitHub provides tools to be more explicit: for example, "Request changes" in a review.



Source: shorturl.at/jpsz7

Code Review: (Author) document code

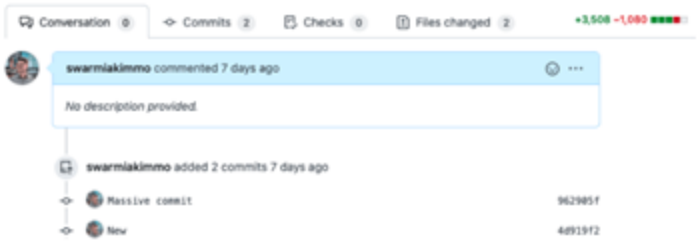
Document as much as possible in code: when receiving a comment or suggestion, aim for documenting the discussion in code. If the reviewer is not sure what the X function does, elaborate on the functionality ideally by renaming the function or writing a comment in the code.

This way, the next developer that reads the code will understand the functionality without reading the PR discussions.

Source: shorturl.at/jpsz7

Code Review: some don’ts

Avoid large PRs



Split work to smaller batches

Source: shorturl.at/jpsz7

Code Review: some don’ts

Avoid tedious manual checks

Delegate nit-picking to a computer

Source: shorturl.at/jpsz7

Questions?

(Team) Exercise 2. Code Review [10 min]

Practice reviewing code in pull requests on GitHub.

1. [Author] Assign one or two team members as reviewers in your pull request (PR).
2. [Reviewer] Reviews, discuss, and suggest changes the pull request(s) following recommendation mentioned so far.
3. [Author] Make changes to PR based on the reviewer(s) suggestions, and updates the PR.
4. [Reviewer] Approves the PR.
5. [Author] Merges the PR into the repository.

Episode
Collaborative Guidelines

(Individual) Exercise 3. Contributing guidelines [8 min]

Add contributing guidelines to your very first repository (workshop?), using the template available in: https://github.com/manuGil/fair-code

Episode
Other Best Practices

Licensing

- Always choose a licence for your opensource software.
- TU Delft Software Policy: researchers can get copyrights if publishing software by a pre-approve license: https://github.com/manuGil/fair-code

Software Citation

- Use a CITATION.cff with your software: CFF generation tool

(Individual) Exercise 4. Licensing and Citation [10 min]

Add a license and citation file to your first repository (workshop?)

1. Add an open-source license to the repository. Use the Creative Common license tool to decide which license to use: https://chooser-beta.creativecommons.org
    a. Check the GitHub documentation to know how to add a license file.
2. Use this tool to generate a CITATION file and add it to your repository:

   Search for the Internet for: cffinit

Software Releases

- A set of changes to a software ready for distribution.
- Version your software using semantic versioning:

Git version: **2.42.0**

MAJOR.MINOR.PATCH

Questions?

Summary

- Code reviews are essential to produce high quality software.
- Be mindful when giving feedback to someone else code.
- Collaborative guidelines let potential collaborators to know how they can contribute to a software.
- It is important to think about citing and licensing your software.
- Use semantic versioning when releasing software.

Questions?

Code & Data Office Hours 📣

- Book a 45 min. appointment with an RSE or DM from the DCC.
- Request a Code Check for your publication.

edu.nl/rjcqv

Book a spot
or visit
www.dcc.tudelft.nl

Best practices for Code review

Follow up readings:

- Github pull request reviews

- A complete guide to code reviews
- Code Review Guidelines for Humans
- Google’s Code Review Developer Guide
- Best Practices for Reviewing Pull Requests in GitHub