Open Source Best Practices (OSBP)

Series of lectures by Yegor Bugayenko
To be presented to students of Innopolis University in 2023–2024
The entire set of slide decks is in the yegor256/osbp GitHub repository

ABSTRACT:
At the course, students will learn how to interact with other programmers in open source GitHub repositories, such that pull requests integrate seamlessly, reputation grows, popularity of repositories increases, and satisfaction of being an open source contributor materializes. This skill may help students in their work with proprietary repositories too, especially when teams are remotely distributed.

What is the goal?

This course introduces some best practices of GitHub-based software development, which eventually may help students gain 10K stars at their own repositories.

Who is the teacher?

Yegor is developing software for more than 30 years, being a hands-on programmer (see his GitHub account with 4.5K followers: @yegor256) and a manager of other programmers. At the moment, he is a director of an R&D laboratory in Huawei. His recent conference talks are in his YouTube channel. He also published a few books and wrote a blog about software engineering and object-oriented programming. He previously taught a few courses in Innopolis University (Kazan, Russia) and HSE University (Moscow, Russia), for example, SSD16 (2021), EQSP (2022), PPA (2023), COOP (2023), PMBA (2023), and SQM (2023) (all videos are available).

Why this course?

Writing code is the so-called "hard skill" that most students are very advanced at, when they graduate their bachelor or master programs. However, they usually lack experience in open source development, where teams are distributed, team members are not closely related to each other, and quality expectations are higher than in co-located teams that work with proprietary code bases. Introducing students to the so-called "soft skills" may be beneficial to the projects that they will join after graduation: students will understand the dynamics and mechanics of software engineering much better and will contribute more fluently and effectively.

What's the methodology?

There are eight lectures, each of which is a summary of best practices, as Kaicode, an open source festival organized and sponsored by Yegor since 2015, sees them. In laboratory classes, students either write student research papers or submit pull requests to GitHub repositories suggested by the teacher. Groups that wrote the best papers will be encouraged to submit them to ICSE, ESEC/FSE, or a similar A* conference (student or NIER tracks); the teacher will help them prepare the papers accordingly.

Course Structure

Prerequisites to the course (it is expected that a student knows this):

- How to write code
- How to design software
- How to use Git

After the course, students *hopefully* will understand:

- How to make their bug reports appreciated?
- How to make their pull requests merged?
- How to reject a pull request politely?
- How to become an active contributor of a large repository?
- How to keep up with GitHub etiquette?
- How to invite and motivate contributors?
- How to deal with frustration during code reviews?
- How to avoid stale pull requests (never merged)?
- How to use GitHub Actions effectively?
- How to format the `README.md` file?
- How to control quality and avoid chaos in a public repository?
- How to use GitHub account in lieu of a C.V.?
- How to get 100 stars?
- How to release in one click?
- How to employ ChatGPT as a coding companion?
- How to get 10K stars?

Lectures & Labs

The following lectures constitute the course:

1. Debating
2. Reporting Bugs
3. Making Changes
4. Reviewing Changes
5. Setting Guidelines
6. Integrating
7. Releasing
8. Gaining Popularity

At laboratory classes, organized by a Teaching Assistant (TA), students either make pull requests to GitHub repositories suggested by the teacher, or write sections for their research papers.

Most probably, one of the following repositories will be suggested by the teacher for contribution during the course: yegor256/cam, yegor256/rultor, yegor256/qulice, cqfn/jpeek, and objectionary/eo.

Grading

At the first lecture, students form groups of **2–4** people in each one (no exceptions!). Each group picks a research topic from the list suggested by the teacher.

Each group writes a research paper in LaTeX, according to the guideline. The length of the pager may not exceed four pages in acmart/sigplan 10pt format (including references and appendices). The paper must be presented to the teacher incrementally, section by section: 1) Method, 2) Related Work, 3) Results, 4) Discussion, 5) Conclusion, 6) Introduction, and 7) Abstract. Once a section is approved by the teacher, the next section may be presented for review.

After a presentation of a section, the teacher may ask the group to *stop* working with the paper. In this case, no sections may be presented for review any more: they all will be rejected. This decision is *subjectively* made by the teacher and will *not* be explained to the students, however the following may contribute to such a negative decision: a) ChatGPT, b) plagiarism, c) negligence, and d) laziness.

When the Abstract is accepted by the teacher, a group may ask a student from another group to review their paper, according to this guideline. The review must be accepted by the TA.

There is no exam at the end of the course. Instead, each student earns points for the following results:

Result	Points	Limit
Attended a lecture	+2	8
Attended a lab	+2	8
Merged a pull request to a suggested repo	+4	32
Reviewed a paper of another group	+3	9
"Method" section (guideline)	+8	
"Related Work" section (guideline)	+12	
"Results" and "Discussion" sections (guideline)	+12	
"Conclusion"	+6	
"Introduction" section	+6	
"Abstract" and Title	+4	

Then, 56+ points mean "A," 48+ mean "B," and 24+ mean "C."