Open Source Best Practices (OSBP)

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

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

What is the goal?
This course introduces best practices for GitHub-based software development, which may eventually help students gain 10,000 stars in 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 in when they graduate from their bachelor's or master's 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 working with proprietary codebases. Introducing students to the so-called "soft skills" may be beneficial to the projects 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 a summary of best practices as seen by Kaicode, an open source festival organized and sponsored by Yegor since 2015. In laboratory classes, students either write research papers or submit pull requests to GitHub repositories suggested by the teacher. Groups that write 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?
- How to earn money via open source?

Lectures & Labs

The following 80-minute 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 (Bash, Python), yegor256/rultor (Java, XML), yegor256/qulice (Java), cqfn/jpeek (Java, XML), and objectionary/eo (Java, XSLT).

Grading

To pass the course, students must attend lectures and earn points by contributing to any of these open-source projects:

- `objectionary/lints` (Java + XSLT)
- `cqfn/aibolit` (Python)
- `objectionary/eoc` (JavaScript)
- `yegor256/0rsk` (Ruby)
- `objectionary/reo` (Rust)

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

Result	Points	Limit
Attended a lecture	+1	6
Submitted a ticket (that was accepted)	+2	8
Submitted a pull request (that was merged)	+4	32

Final grades are based on accumulated points:

- 25+ earns an “A,”
- 17+ a “B,” and
- 9+ a “C.”

If eight or fewer points are accumulated during the course, the student may still earn them before the retake.

An online lecture is counted as “attended” only if a student was personally present in Zoom for more than 75% of the lecture’s time. Watching the lecture from a friend’s computer doesn’t count.