Ensuring Quality in Software Projects

Series of lectures by Yegor Bugayenko

Abstract:
The course is a series of loosely coupled pieces of advice related to quality of software development. Pragmatic programmers may listen to them if they don't want to tolerate chaos in their projects. The course is not only about coding practices, but also about static analysis, test coverage, bug tracking, dependency and artifact management, build automation, DevOps, and many other things. If we don't do them right, they may severely jeopardize the quality of the entire project.

Who is the teacher? I'm developing software for more than 30 years, being a hands-on programmer (see my GitHub account: @yegor256) and a manager of other programmers. At the moment I'm a director of an R&D laboratory in Huawei. Our primary research focus is software quality problems. You may find some lectures I've presented at some software conferences on my YouTube channel. I also published a few books and wrote a blog about software engineering and object-oriented programming.

Why this course? In one of my videos a few years ago I explained what I believe is killing most software projects: it's the chaos they can't control. Most of us programmers start projects full of enthusiasm and best intentions. We are confident that this time the design will be solid, the code will be clean, and our customers will be happy because there will be no bugs. Eventually, sooner rather than later, the reality appears to be as bad as it was in the previous project: the code is messy, the design resembles spaghetti, and the bugs are unpredictable and hard to fix. We learn some lessons, abandon the project, and start a new one, again with the best intentions. But in a new project nothing changes. Most programmers that I know run in this cycle for decades. I believe, this course may help you not become one of them.

What's the methodology? The course is a collection of individual cases not closely connected to each other. Each lecture discusses a single open-source GitHub repository. Each discussion highlights the mistakes made in the repository and suggests improvements. Each lecture ends with a conclusion and a formulated recomendation. The recomendations may help students prevent and control chaos in their own future projects.

Course Aims

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

- How to use Git
- How to code
- How to design software
- How to write automated tests
- How to deploy

After the course a student *hopefully* will know:

- How to prevent dependency hell
- How to use static analysis
- How to organize automated tests
- How to design integration tests
- How to maintain documentation
- How to avoid code smells
- How to convince a manager that quality is important
- How to organize bug tracking
- How to do code reviews
- How to configure pre-flight builds
- How to deal with negligence of other programmers
- How to practice DevOps being a programmer
- How to hire a programmer
- How to measure test coverage
- How to argue with customers
- How to discipline fellow programmers
- How to protect yourself from chaos