Code Review Basics

Hui Chen [a]

[a]CUNY Brooklyn College, Brooklyn, NY, USA

May 13, 2025

Outline

Outline

Code Review

After completing a milestone (e.g., complete a user story) and assumes after all tests pass,

- Developers review the code written by the others
- Sometimes the implementor and the reviewer or the reviewers discuss some aspects of the code

Purpose of Code Review

Main purposes are

- Improving code quality (by fixing bugs, by improving readability, ...)
- Improving people's understanding of the code
- Improving people's implementation skills

A common practice in software development organizations.

Benefits of Code Review

Some studies show that code reviews are more effective in catching errors than testing

- Code reviews are able to find different kinds of errors (e.g. unclear error messages, inadequate comments, hard-coded variable values, repeated code patters that should be consolidated)
- Provide a place to enforce coding conventions

Also, when developers know that their code will be reviewed, they scrutinize it more carefully themselves.

Code Review Approaches: Formal vs. Informal

- Formal code review
- Informal code review

Outline

Formal Code Review

- Assign distinct roles, e.g., Moderator, Author, Reviewer(s) and Scribe
- Have formal meetings for code review
- Reviewers look through the code before the meeting, and arrive to the meeting with a list of comments
- Focused effort looks for the most frequent kinds of errors
- Detailed comments to author ensure that author learns from mistakes
- Usually involves a post-meeting to ensure that all the requested changes were made by the author
- A well functioning company separates code reviews from employee performance reviews.

Example. See IBM

Outline

Informal Code Reviews

Somewhat less effective than formal reviews, but also less time consuming

- Preparation for the review is also important (code should be read beforehand)
- Author usually moderates
- No follow up to ensure bugs were properly fixed

There are a number of approaches, e.g.,

- Pair programming
- Over-the-shoulder reviews
- E-mail pass-around reviews
- Tool-assisted reviews

Pair Programming

Two people program on the same screen

- Driver does the typing
- Observer checks for errors

Can be exceptionally useful for:

- complex parts of the code, where two sets of eyes would produce a better result
- for learning (Novice with Expert pairings)

Key part of extreme programming agile methodology

Disadvantage: cost, lost productivity.

Tool-Assisted Reviews

Tools are developed to aid code reviews, e.g.,

Example: Gerrit

Projects using Gerrit

- Open Stack

Practice at Example Companies

- Google
- Facebook

Outline

Summary and Questions

- Code reviews can improve product quality, and programmer skill (and code knowledge)
- Formal vs. Informal Code Reviews
  - Tools are available
- Preparation for the review is essential

Outline

Zeller, Andreas. Why programs fail: a guide to systematic debugging. Elsevier, 2009.
"Engineering Software as a Service" by Armando Fox and David Patterson (2nd Edition)
"Introduction to Software Design with Java" by Martin P. Robillard
"Essentials of Software Engineering" by Frank Tsui, Orlando Karam, and Barbara Bernal(4th Edition)