Erik Sierra

3/11/2024

CS 499 Module 2 Journal

**Part 1: Code Review and Its Importance**

What is Code Review?

A code review is a systematic examination of source code by one or more developers to find defects, improve quality, and ensure compliance with coding standards before the code is merged or released. It can be conducted manually (peer review) or with automated tools.

Why Is Code Review Important for Computer Science Professionals?

Code reviews are a important practice in software development for many reasons:

* Improves Code Quality – Detects bugs, vulnerabilities, and inefficiencies early in development.
* Enhances Maintainability – Ensures code is readable, structured, and documented properly, making it easier to modify in the future.
* Ensures Consistency – Adheres to coding standards and best practices across the codebase.
* Encourages Knowledge Sharing – Helps junior developers learn from senior team members; this promote a collaborative learning environment.
* Reduces Technical Debt – Identifies inefficient or redundant code, preventing unecessary delays in the future.

Best Practices for Code Reviews

From the resources, some crucial best practices in code review include:

1. Limit the Number of Lines of Code (LOC) Per Review – In their article SmartBear explains that reviewing more than 400 LOC at a time reduces the effectiveness of defect detection. Keeping reviews between 200-400 LOC over 60-90 minutes results in 70-90% defect discovery (2025).
2. Review at a Slower Pace – Code should be reviewed at a rate of under 500 LOC per hour to maintain thoroughness.
3. Timebox Reviews – A code review should not exceed 60 minutes at a time to avoid fatigue and maintain efficiency.
4. Use Checklists – Helps ensure common mistakes are consistently caught, especially issues that are difficult to spot (Team, 2023).
5. Annotate Code Before Review – Developers should provide explanations for complex logic to guide reviewers through changes.
6. Establish a Defect Fixing Process – There should be a structured approach to logging and resolving issues found during review.
7. Maintain a Positive Review Culture – Feedback should be constructive and aimed at improving the code rather than criticizing the developer.
8. Automate Where Possible – Tools like linters and static analysis tools can catch syntax issues and enforce style guidelines, allowing reviewers to focus on logic and architecture.

When Should Code Reviews Occur in the Development Process?

Code reviews should occur before merging code into the main branch but after unit testing has been completed. The rationale for this is:

* Reviewing before merging ensures that only high-quality, bug-free code integrated into the main codebase.
* Conducting reviews after unit testing ensures that basic functionality is verified before peers spend time on a detailed review.
* Reviewing too early (before testing) may lead to wasted effort on debugging issues that could have been caught automatically.
* Regular code reviews (e.g., after each sprint or feature implementation) maintain consistent code quality throughout the development cycle.

**Part 2:**

**What software have you chosen to use to record your code review?**

Software I’ve chosen to record my code review is called ScreenRec. This tool is a simply downloadable software that can screen record a monitor, while also processing internal and external sound. This is helpful so that I am able to speak while reviewing my code on screen.

**Describe your approach to creating an outline or writing a script for your code review for each of the three categories that you will be reviewing based on the rubric as well as the code review checklist.**

My approach to getting setup for this code was preparing previous assignments and resources that explain in sum what is going with my code, as well as the different aspects related to each category. I had sections that related to each section so that I was able to speak about each one and how they all work with each other to make a unified project at the end.

References

SmartBear. (2025). *Best practices for peer code review*. smartbear.com. <https://smartbear.com/learn/code-review/best-practices-for-peer-code-review/>

Team, F. T. (2023, April 28). *Code review checklist: 7 steps to level up your review process*. Pluralsight. <https://www.pluralsight.com/resources/blog/software-development/code-review-checklist>