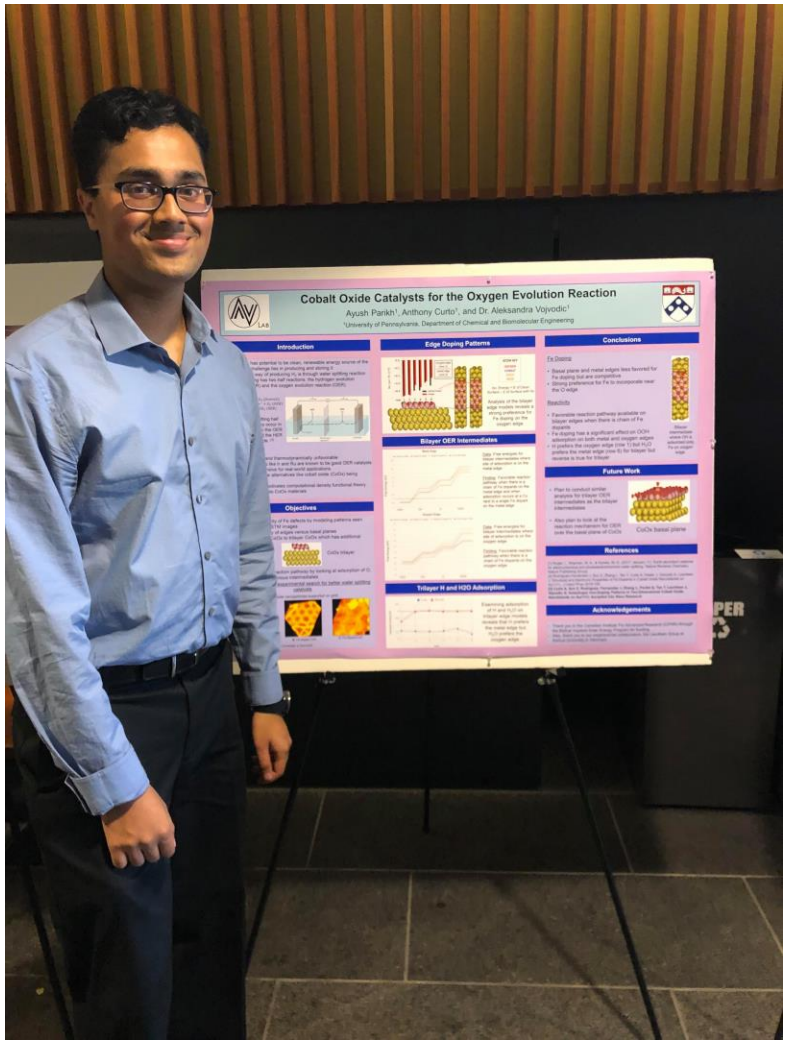
How Microsoft Ensures Code Quality

Ayush Parikh
Georgia Tech
CS 8903 Seminar
May 22, 2024

About Me

- Software engineer at Microsoft since Sept 2021
- Interned summer 2020
- Backend engineer for Microsoft Teams
- BSE in CS from Penn in May 2021
- Previous undergrad research experience studying catalysts for hydrogen fuel cells
- New member of HAAG on Higher Ed team



Guiding Principles

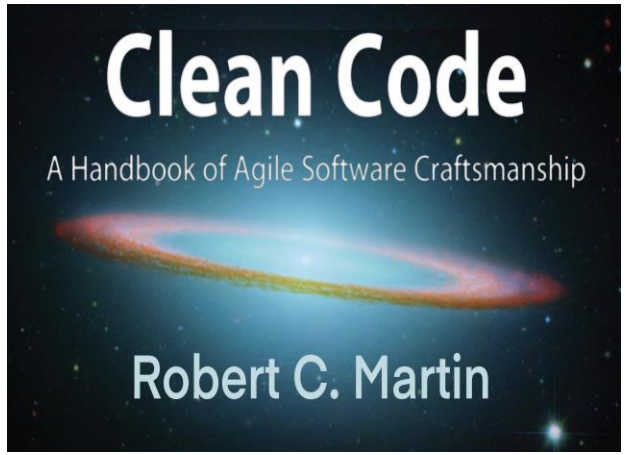
1. Write clean code
2. Document and share knowledge
3. Test, test, and test!
4. Get feedback and incorporate it
5. Leverage AI tools to your advantage

1. Write Clean Code

- Keep it simple stupid! Avoid unnecessary complexity
- DRY = Don't repeat yourself. Move repeated code to a function.
- Functions should be single purpose with fewer arguments
- Ensure variable and function names are descriptive

Instead of ***bool flag*** use ***bool shouldSetImmersiveModeCapability***

- Declare variables close to usage
- REMOVE commented code blocks



1. Write Clean Code

- Use comments for explanation of intent, clarification, or as a warning of consequences

```java
public static SimpleDateFormat makeStandardHttpDateFormat()
{
  //SimpleDateFormat is not thread safe,
  //so we need to create each instance independently.
  SimpleDateFormat df = new SimpleDateFormat("EEE, dd MMM yyyy HH:mm:ss z");
  df.setTimeZone(TimeZone.getTimeZone("GMT"));
  return df;
}
```

- Preferred to explain yourself in code

```java
// Check to see if the employee is eligible for full benefits
if ((employee.flags & HOURLY_FLAG) && (employee.age > 65)){...}
```

```java
if (employee.isEligibleForFullBenefits()){...}
```


Source = https://medium.com/codex/clean-code-comments-833e11a706dc

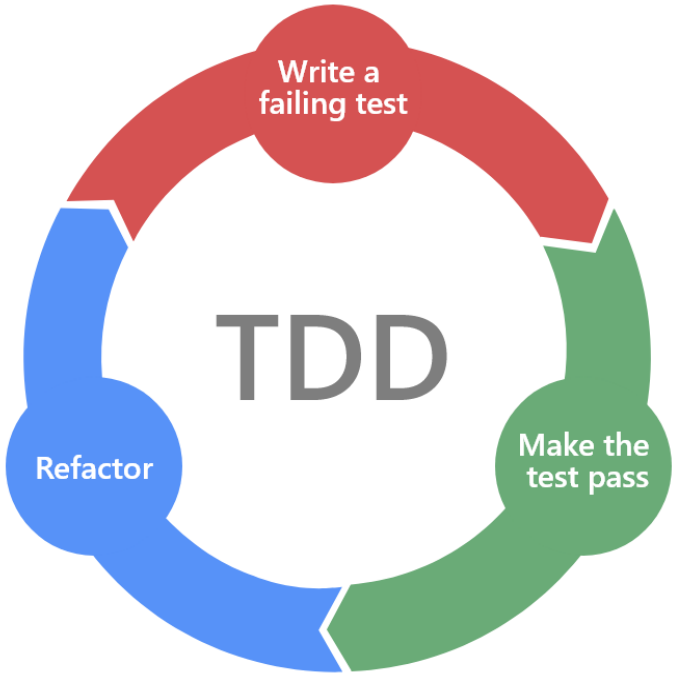
2. Document and Share Knowledge

- Document all interfaces with summaries of functions, parameters, and return values
- Include types for all variables whenever possible
- All new APIs go through a design review and are documented via Swagger
- Any new service goes through an architecture review and security review
- Team discusses best practices in weekly "Dev Discussions" meetings on Fridays

```
namespace TestConsole
{
    2 references
    public interface ICommentExample
    {
        /// <summary>
        /// Checks if user with given email already exist on the system
        /// </summary>
        /// <param name="email"></param>
        /// <returns>Returns true if user is already registered in the system else false</returns>
        2 references
        Task<bool> IsUserExist(string email);
    }
    1 reference
```

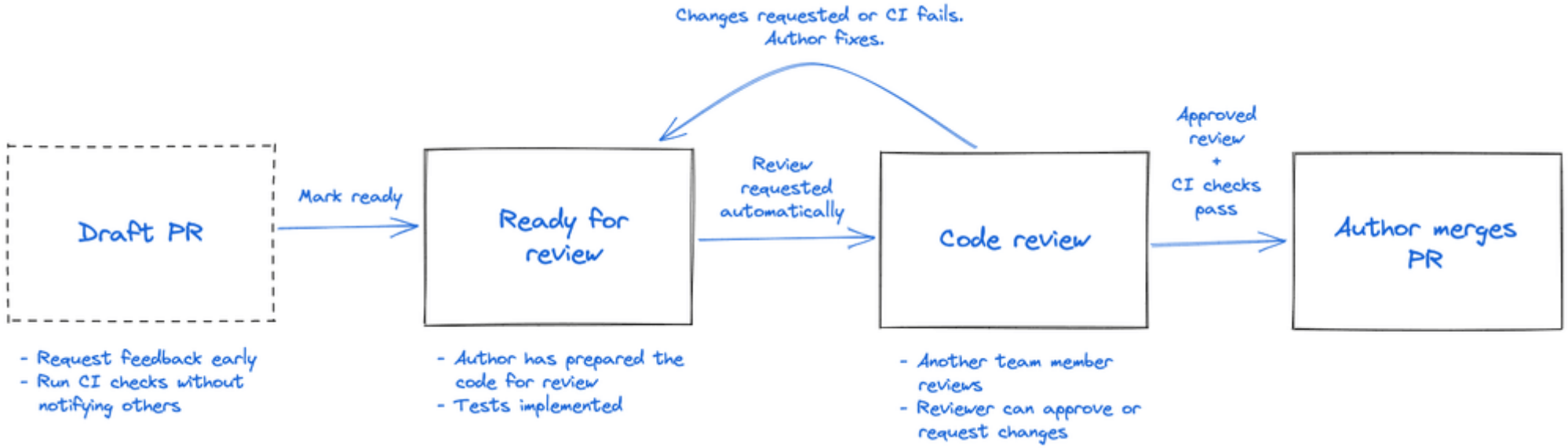
3. Test, test, and test!

- Test driven development is emphasized
- Unit tests
- End to end (E2E) tests
- Manual tests in dev environments
- Integration tests that make calls to all partner services
- Any pull requests require test coverage and validation links with green test pipeline runs



4. Get Feedback and Incorporate It

- For any commit to master, 2 engineers must review and approve and highly encouraged to have > 2 reviewers
- Pull request (PR) comments must be addressed before making a commit



Source = https://www.swarmia.com/blog/a-complete-guide-to-code-reviews/

5. Leverage AI tools to your advantage

- Microsoft teams encourage use of Github Copilot which is especially useful for quick scripting and writing tests systematically
- All pull requests are automatically annotated with an AI generated description of changes
- Pull requests now also get automatically commented with AI suggestions for refactoring
- This is just the start – more advanced AI tooling is yet to come!

Source = https://www.swarmia.com/blog/a-complete-guide-to-code-reviews/

Guiding Principles Summary

1. Write clean code
2. Document and share knowledge
3. Test, test, and test!
4. Get feedback and incorporate it
5. Leverage AI tools to your advantage

Thanks for listening!

aparikh49@gatech.edu

Questions?

Let's also open the floor for discussion. Would anyone like to share how their organization maintains code quality?