Software Testing 2022

S1833951 li fang

January 26 2022

5.1. Identify and apply review criteria to selected parts of the code and identify issues in the code.

R1:

- Usability: The readability of the code is not too good, as I use method, which does return a Boolean value by taking in the 8 coordinates of the two set of points. So, it is very compact, user will need to check on this method in order to understand this function
- Functionality: the function is full, but it is not friendly for any alternation, as the core method linesIntersect can only be replace by rewriting the whole function, so if there is any update requirement for this function, I might need to rewrite this function.
- Maintainability: the function is easily maintained, as it does not contain too many methods. It is easily tested and debugged, as it is simply structured.

R2:

- Reliability: The function is not reliable enough, as when I am doing mutation test, any minor mistake will result total failure of the function. That is expected though, because this feature has not been taking consider when construct this function
- Usability: the readability is good, as both of the function used for r2 is easily structure and with comment given, and the method been used are common except for Math.pow, which is not a regular one to be used.
- Maintainability: The function's maintainability is medium, it is easily understood and easily structured, so can be updated easily, but because it is easily structured, any update on the requirement which result in a modify in code, may need to rewrite the "near" function, the "dis" function will work well though.

5.2. Description of a CI pipeline

A CI pipeline is a set of automated processes that are used to build, test, and deploy software. The pipeline typically includes several stages, such as integration, building and packaging, testing, and deployment. Each stage is typically triggered automatically when code is committed to a version control system.
The goal of a CI pipeline is to keep the software in a state that is ready for release at all times by identifying and addressing problems early in the development process. This could involve detecting and correcting bugs, maintaining code quality and consistency, and confirming that the software fulfills the requirements and specifications.

There are some feature CI pipelines have:

- Source code management system: Use of software such as Git, Subversion, or any other source control system, managing source code online for back-up, or re-roll when the new version is broken.
- Set up automated build: Use a build tool such as Gradle, Maven (the one I had use for this project) to automate the process of compiling and packaging the code into a deployable artifact, saving time and resources magnificently.
- Configure automated testing: The use of Integrate testing frameworks such as the one been use for this project, the Junit test to run unit tests, integration tests, and other types of tests on the code changes.
- Implement code analysis: The use of Integrate code analysis tools such as SonarQube, PMD, ESLint, to ensure code quality and compliance with coding standards.
- Establish a deployment process: The code changes can be deployed to a staging or production environment using Jenkins, Terraform, or other deployment tools.
- Monitor the pipeline: Set up monitoring and logging systems to track the pipeline and detect any issues that may arise.
- Implement Continuous delivery/Deployment: If the pipeline passes all stages, it can automatically deploy the software to a production environment
- Feedback Loop: Continuously gather feedback from the users and incorporate them into the development process

5.3. Description of what aspects you would automate is

- Unit Testing: Unit tests are typically automated and should be run as part of the build process. Automated Unit test can ensure the system works correct as individual function, make it easier to find bugs. Automated unit test is also useful to find bug during developing phase of the software, and unit test is the test can be easily automated since they are always small, atomic test.
- Integration testing: Integration tests are used to test the integration of different units of code, such as modules or components. Integration tests can be run after unit test, to ensure the performance of some sub-branch of the system, integration test can be automated using testing framework or tools, so should be easily automated.
- Performance testing: Performance tests are used to test how the software behaves under different loads and conditions. It can include load testing, stress testing, and scalability testing. Since performance tests can be automated by using tools such as Apache JMeter, Gatling, so it should be easily automated without much effort been put.
- Security testing: Security tests are used to test the software's security features and identify potential vulnerabilities. It always includes test like penetration testing. There are plenty of tools can automate security tests like OpenVAS and Burp Suite in real life. Automated test remove human intervention, and security test is regularly needed, so automation suit very good for security test.

5.4. Description of how the pipeline would operate and what issues you think it would identify

A CI pipeline does operate by automating a series of tasks and tests that whenever the code is updated or changed. The process starts with a developer committing code changes to a version control system, such as Git. This will trigger the CI pipeline to automatically pull the latest code

and run a series of tasks, which include running unit tests and other tests, and performing static code analysis all sort of stuff. If any of these tasks fail, the pipeline will alert the developer at the moment to fix the issue.

Once the pipeline has completed all of the tasks successfully, it will deploy the code to a staging environment for further testing. If the staging environment passes, the pipeline will deploy the code to the production environment. This process ensures that issues are caught and addressed as early as possible in the development process, helping to ensure that the software is always in a releasable state.

Issue that pipeline should identify:

- Bugs and errors: These can be syntax errors, logical bugs, or any other kind of issue that makes the code not work as intended. For example, if the code is supposed to calculate the distance from point A to B, if the calculate it wrong, or using wrong output, the pipeline should be able to identify this.
- Code quality issues: These can include poor code structure such as too many loops inside loop, lack of commenting as readability problem. Pipeline can identify various states of the code and if it does not perform efficient enough (for example, trap in a single function for 10 minutes), the pipeline should be able to identify it.
  This can also include the consistency among code, like using none-standard library, using multiple types of polygon inside the same function or system(geojson polygon and other type of polygon lets say).
- Poor test coverage: This can include missing unit tests or integration tests, or not having enough test coverage in general. If a function is not being tested at all, or if the tests are not covering all possible scenarios, the pipeline would identify this issue.
- Security vulnerabilities: this could be caused by lack of encryption on sensitive data, if a developer is using an unencrypted connection to a database to store sensitive data, the pipeline would identify this issue, or any other unauthorized approach to the source code and database should be notice by pipeline.
- Incompatibility with other system: This can include issues related to integration with other systems, or issues related to the dependencies used by the software. If the langue is updated, some part of the software is using new method, and old one does not, there would be compatible issue with the latest version and old version of code, the pipeline would identify this issue.