FIT2107 SOFTWARE QUALITY AND TESTING
ASSIGNMENT 3 (PART B)

Nikhita Peswani

31361552

The purpose of this assignment is to reflect up on the work completed in Part A. Part A of this assignment was a code review process in which each team was assigned another team's code to review for potential bugs. Each team reviewed the other team's code and held meetings to discuss any concerns or errors discovered. All issues were raised on the git issue tracker, where the coding team could view the code and respond to feedback. Even though we were not required to edit the code to resolve those issues, I believe that learning the entire process of raising issues helped each of us become more familiar with how it works in the real world.

I believe the thing that could have been done better in our code review process was during the Preparation and Examination step.

In the Preparation step, reviewers are to carefully examines the code before the meeting. They are obligated to read the code, understand its structure and operation, and are expected to list all of their questions/doubts, as well as potential change requests (CR) and suggestions for improving the code. We partially exploited the preparation step by preventing ourselves from understanding the organisation and operation of the entire code by dividing the code into parts based on specifications, with each of us reviewing the portion that we implemented for our own code for that specification. As a result, each of us only knew a portion of the code and had no real opinions on the issues raised by our colleagues. Whatever issues that were raised by any one of us, were actually presented to the authors of the codes. This may not be the case in the real world because an issue/suggestion will not necessarily be an issue to another reviewer, or the other reviewers may have better suggestions to the ones we have proposed.

During the examination process, the author presents the procedural logic used within the code, the paths indicating significant computations, and the interdependence of the unit under review on the other units. The presenter is supposed to read the code line by line and the reviewers are expected to ask any questions they may have prepared about that particular section of the code. There was no presentation by the author and no code reading by the presenter at our code review meeting. We skipped right to the part where we asked questions and provided our suggestions. There were times when the author was unaware of which parts we were discussing, or when certain questions were raised, they were already addressed in the code but we as reviewers were unaware of what they were for. This type of disruption could have been avoided if we had followed the process correctly.

The peer review process allowed authors to get feedback on their code from reviewers, allowing them to identify any potential flaws in their code that they had missed. If I was a coder on an individual group basis, the other team members acted as reviewers and assisted in identifying any missing parts. Yes, in future software development projects, I would adopt/conduct reviews for other software artefacts (such as class diagrams, for example). This is because it aids in the verification of artefacts that I believe are correct but may lack some functionalities.

Proposing fixes for identified defects may have various advantages and disadvantages based on my experience in this assignment as well as the readings provided to us. However, whether they are advantages or disadvantages depends on who we are proposing it to. This means that some people do not respond well to constructive criticism.

Everyone implements the code in a unique manner based on their knowledge and experience. However, there may be other coders on the project with superior skills. When they propose fixes for identified defects, the current programmer may benefit from it. Because the reviewer reviews the code, they can provide a thorough review of the code and ensure the project's efficiency.

Proposing fixes, on the other hand, has drawbacks. The programmer may dislike other people's opinions and believe that because his code is working properly, there is no need to change it based on the suggestions of other coders. These people are unconcerned about the code's complexity or duplication. The process of identifying and proposing fixes may be very time consuming for a reviewer and no output may come out if the coder is not open to new ideas. In addition, there are no metrics or process measurement tools to ensure that a particular fix/suggestion will actually contribute to the code's improvement. A reviewer may mark almost everything as a defect, resulting in low self esteem of the coder and they need to rewrite the majority of the code.

Yes, absolutely, I would propose fixes for identified flaws in my own projects at my university or office. This technique allows an individual to hone their skills while also ensuring that the project's end output is efficient.

Testing a software to access human aspects such as different ages, gender, sex, disabilities, nationalities, and so on is critical in software development because it determines the quality of the software / applications developed. It assists us in identifying and correcting usability flaws in the product and ensuring that the software is usable not only by developers but also by those for whom we are developing it.

However, there are some difficulties when testing for human factors. This procedure is time-consuming and may be costly as we will need to gather relevant people to test the program. Each human aspect has so many variations that testing for all of them may be impossible.

User-oriented testing, usability testing, and regression testing are examples of software testing approaches that a developer can use to assess non-functional correctness.

The process of testing the interface and functions of a website, app, product, or service by real users performing specific tasks in realistic conditions is known as user-oriented testing. The goal of this process is to assess the usability of that website or app and determine whether the product is ready for real-world use. For relevant results, testers should not be overly directed and should be allowed to interact with the website or app naturally, to determine whether the system is intuitive and comfortable enough for people who are unfamiliar with it.

Usability testing is the practice of testing the usability of a design with a group of representative users. It usually entails watching users as they attempt to complete tasks and can be done for a variety of designs.

Regression testing is a type of software testing used to ensure that a recent program or code change has not affected existing features negatively. Regression testing is simply a full or partial re-execution of previously executed test cases to ensure that existing functionalities continue to function properly.