SENG8040

Testing Methodologies

Assignment #3: Functional / Mutation Testing – Due October 30th, 2019 @ 8:00AM - In Class

Question 1 [30 marks]

Background

You work in the Quality Assurance area for a company that develops mobile phone software. One of the new applications that your company is developing is voice recognition software that will accept a variety of voiced in commands and places the telephone call. The project charter and business requirements are in the process of being developed for this project. In anticipation of this, your Manager has asked you to do some preliminary work.

Develop Finite State Machines for the following;

1. Handle outgoing calls using number keypad/verbal/speed dialing methods (20 marks)
2. Handle incoming calls (10 marks)

Upon completion of the Finite State Machines, you will build out simple test cases in the following format.

Test Id	Test Step(s)	Expected Result(s)

The following transition conditions/states must be considered in your answer:
Idle, ringing (incoming/outgoing), hanging up, busy, dialing (verbal, digits, speed), answering, connecting, disconnecting, dial tone, talking, timing out, incomplete dial, invalid number, long distance, dropped call, redial and going to voicemail.

As the most experienced QA person in your group, you have been given the task of figuring out how to develop these.

MARK RUBRIC:
You will get full marks if your Finite State Machine can flow from one state to the next and that the test cases derived match the state machine. One mark will be deducted for each state machine transition that is not correct or not represented by a test case.

Question 2 [70 marks]

1. Using Visual Studio, create a new C# Console Application (or use the same one from Lab #2 earlier this term) for a simple calculator. You may use the AwesomeCalculator that is included in eConestoga with this Assignment.

2. Ensure there are four methods included in this application: addition, subtraction, multiplication, and division.

3. Create three unit tests using NUnit for each method (for a total of twelve unit tests) and ensure that each unit test is passing – attach the results as a screenshot.

4. Create and compile four separate mutants - one for each of the above-mentioned methods.

5. On each Mutant, re-run your unit tests again with the same arguments as in step 3. from above and attach the results as a screenshot.

6. What were the results of your original unit tests with the Mutants?

7. Now, add at least four more unit tests for each mutant (in addition to the original division test cases) and attach the results as a screenshot. You should have 28 (12 original plus 16 for the mutants) unit tests now.

8. What were the results of your unit tests in this case? What does this tell you about the quality of your unit testing strategy based on using Mutants?

Remember that your original program and each mutant must be compiled separately – don't just update your original program to make it a mutant or back to the original program.

For this assignment you must use a form of version control. You must use Git through Git Bash as in Assignment 01 and show three distinct, unique commits.

1. **Demonstrate your program in class.**
2. **MS Word document (in order)**:
    a. Assignment Cover sheet with your name, student ID, "Assignment #3" in the title and date
    b. Assignment Rubric left blank (found on eConestoga)
    c. Print out of your Finite State Machines and the associated test cases.
    d. Copy of your C# source code
    e. Copy of your NUnit Test class source code
    f. Print out of Doc showing the results of your NUnit tests being run in the NUnit UI, along with explanations / answers to the above questions. Finally, a screenshot/output of your git repository log or any output from your version control showing commits.
3. **Source Code (zipped)**: A single compressed (.zip format) archive file containing the solution folder of your source code (so I can run it). You may have five compressed files here - one for the original program plus four for each mutant.

Do not include the MS Word document in any of the .zip files - keep them separate (-10% penalty if otherwise)