Software Quality Assurance (QA) Report

Analysis across project artefacts and milestones

Write a report covering the following topics and questions. Please provide numbers, plots and explanations with few lines of text (say 1-2 paragraphs) for each question. Note that you should not simply answer each question, but rather create a well-rounded report about your testing efforts during the project. Feel free to add any other information you consider useful regarding your testing efforts.

Hint: Read the questions you need to cover in the QA report and plan what information you need to **keep track of and document while working on Milestones 1 & 2, Hackathon, and Rebuttal**. We suggest the use of an issue tracker and additional files to document your testing efforts throughout the semester. You will need such information when preparing your QA report for Milestone 3.

Topics and questions to be covered in the report

1. How much source and test code have you written?
Test code (LOC) vs. Source code (LOC).

2. Give an overview of the testing plans (i.e. timeline), methods and activities you have conducted during the project. What was the most useful method or activity that you employed?

3. Analyze the distribution of fault types versus project activities:

3.1. Plot diagrams with the distribution of faults over project activities.

Types of faults: unit fault (algorithmic fault), integration fault (interface mismatch), missing functionality. Add any other types of faults you might have encountered.

Activity: requirements review, unit testing, integration testing, hackathon, coverage analysis. Add any other activities you have conducted.

Each diagram will have a number of faults for a given fault type vs. different activities.

Discuss what activities discovered the most faults. Discuss whether the distribution of fault types matches your expectations.

3.2. Plot a diagram for distribution of faults found in basic functionality (old code) during activities on adding extended functionality (new code):

Activity: integration testing, hackathon, coverage analysis. Add any other activities you have conducted.

Discuss whether the distribution of fault types matches your expectations.

3.3. Analyze bugs found in your project according to their type (in all milestones).

Analyze and plot a distribution of *causes* for the faults discovered by Hackathon activity.

Analyze and plot a distribution of *causes* for the faults discovered by Randoop, or other tools.

Causes: Error in constants; Error in identifiers; Error in arithmetic (+,-), or relational operators (<, >); Error in logical operators; Localized error in control flow (for instance, mixing up the logic of if-then-else nesting); Major errors (for instance, 'unhandled exceptions that cause application to stop'). Add any other causes you might have identified.

Is it true that faults tend to accumulate in a few modules? Explain your answer.

Is it true that some classes of faults predominate? Which ones?

4. Provide estimates on the time that you spent on different activities (percentage of total project time):

- requirements analysis and documentation %
- coding %
- test development %
- test execution %
- others %

5. Test-driven Development (TDD) vs. Requirements-driven Development. What are advantages and disadvantages of both based on your project experience?

6. Do coverage metrics correspond to your estimations of code quality?

For example, what 10% of classes achieved the most branch coverage? How do they compare to the 10% least covered classes?

Provide your opinion on whether the most covered classes are of the highest quality. If not, why?

7. What testing activities triggered you to change the design of your code? Did integration testing help you to discover design problems?

8. Automated test case generation: did automatically generated test cases (using Randoop or other tools) help you to find new bugs?

Compare manual effort for writing a single unit test case vs. generating and analyzing results of an automatically generated one(s).

9. Hackathon experience: did test cases generated by the other team for your project helped you to improve its quality?

10. Debugging experience: What kind of automation would be most useful over and above the Eclipse debugger you used – specifically for the bugs/debugging you encountered in the project?

Would you change any coding or testing practices based on the bugs/debugging you encountered in the CS4218 project?

Did you use any tools to help in debugging?

11. Did you find static analysis tool (PMD) useful to support the quality of your project?

Did it help you to avoid errors or did it distract you from coding well?

12. Propose and explain a few criteria to evaluate the quality of your project, except for using test cases to assess the correctness of the execution.

13. What gives you the most confidence in the quality of your project? Justify your answer.

14. Which of the above answers are counter-intuitive to you?

15. Describe one important reflection on software testing or software quality that you have learnt through CS4218 project in particular, and CS4218 in general.

16. We have designed the CS4218 project so that you are exposed to industrial practices such as personnel leaving a company, taking ownership of other's code, geographically distributed software development, and so on. Please try to suggest an improvement to the project structure which would improve your testing experience in the project?