CS4218 SOFTWARE TESTING Spring 2016
National University of Singapore
Offered by Prof. Abhik Roychoudhury during 2014-16

[Ack: Abhijeet Banerjee, Dr. Konstantin Rubinov, Dr. Marcel Boehme, Sergey Mechtaev for their contributions, specifically A. Banerjee in 2014]

Software Quality Assurance (QA) Report

Topics: Comparison of testing strategies, measures of confidence, Orthogonal Defect Classification (ODC).

Analysis across project artifacts and milestones.

[Project assessment and artifacts]:

A. Unit test cases – 10 marks
B. TDD test cases – 3 marks
C. Integration test cases - 5 marks
D. Code quality – 2 marks
E. Hackathon bug reports – 5 marks
F. Hackathon bug fixes – 5 marks
G. QA report – 10 marks

TOTAL: 40 marks

Please answer the following questions. Please provide numbers, plots and explanations with few lines of text (say 1-2 paragraphs) for each question.

Report

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

---

2. Analyze distribution of fault types versus project activities:

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

   Types of faults: unit fault (algorithmic fault), integration fault (interface mismatch), missing functionality.

Activity: requirements review, unit testing, integration testing, hackathon, coverage analysis.

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.

2.2. 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.

Discuss whether the distribution of fault types matches your expectations.

2.3. Analyze bugs found in your project according to their type.

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.

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').

Is it true that faults tend to accumulate in a few modules?

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

---

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

- requirements analysis %
- coding %
- test development %
- test execution %
-

---

4. Test-driven Development (TDD) vs. Requirements-driven Development

What are advantages and disadvantages of both based on your project experience?

---

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

In particular, 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?

---

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

---

7. Automated test case generation: did automatically generated test cases (using Randoop) 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).

---

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

---

9. 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 CS4218 project?

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

---

10. 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?

---

11. How would you check the quality of your project without test cases?

---

12. What gives you with the most confidence in the quality of your project?

---

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

---

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

---

15. 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. Can you try to suggest an improvement to the project structure which can help us relate the project to industrial practices more tightly?