SE463: Software Testing and Quality Assurance

PROJECT DESCRIPTION

For this project, you will consider an existing software application and apply systematic testing to make it robust. More specifically, you will assess its overall quality and will report on ways to improve it. To this end, you will be using some of the testing techniques we learn in this course in order to detect and report various faults from the selected application.

Your final deliverable for this project will be a complete and professional testing report. The original developer(s) will use this report to replicate and remove the defects you have discovered and address any other recommendations you may have on how to improve the tested application. *Note: it is important that you describe carefully in your report ways to replicate all defects found and justify why they need to be fixed.*

Logistics

1. Find at least one partner from our class to work together on this project. If you prefer to work alone, please discuss it with your instructor.
2. Review all possible options of candidate software applications to be tested (see some posted on our course website).
3. Together with your partner, select an application of your choice to test from the list posted on Moodle. If you have any other ideas of your own for this project, you can discuss them with your instructor.
4. Send an email to your instructor with the name of your team, partner(s) and the selected project.
5. Link to your own team's Wiki on Moodle and start posting related information about your project.
6. Prepare and post regular Weekly Status Reports (WSR) on Moodle. Use the template posted under the Project module.
7. Test drive the selected application and start by doing some exploratory testing (as described below). Once you get a good understanding of the application, decide with your partner on a more in-depth testing strategy which will include some black-box and white-box unit testing techniques we learned in class.
8. By the end of the semester, complete and submit a final evaluation report that includes clear replication steps of all faults detected and other recommendations for improving the overall quality of the application (see a recommended format of your final report in the Appendix below)

About Exploratory Testing (Chapter 18)

James Bach defines Exploratory Testing as the simultaneous *learning*, test case *design*, and test *execution* (www.Satisfice.com)

Observation Tours for Learning *(Mike Kelly http://michaeldkelly.com/blog)*

- **Feature tour**: Move through the application and get familiar with all the controls and features you come across.

- **Complexity tour**: Find the five most complex things about the application.
- **Claims tour**: Find all the information in the product that tells you what the product does.
- **Configuration tour**: Attempt to find all the ways you can change settings in the product in a way that the application retains those settings.
- **User tour**: Imagine five users for the product and the information they would want from the product or the major features they would be interested in.
- **Scenario tour**: Imagine five realistic scenarios for how the users identified in the user tour would use this product.
- **Variability tour**: Look for things you can change in the application - and then you try to change them.
- **Data tour**: Identify the major data elements of the application.
- **Structure tour**: Find everything you can about what comprises the physical product (code, interfaces, hardware, files, documentation, etc.)

Test Case Design & Execution

- You should consider the following when designing your test cases:
    - *Mission*: What am I trying to accomplish?
    - *Coverage*: To what extend am I planning to test?
    - *Risk*: What can go wrong?
    - *Techniques*: How am I going to test?
    - *Environment*: Where, when, for how long, with whom and with what, am I testing?
    - *Status*: Where am I? What have I found so far? How much more do I have to do?
    - *Obstacles*: Do I have any issues I need help with? Is there anything I cannot work around?
    - *Audience*: Whom is this report for?

Justification

When you find an issue, you need to explain and justify why you think it is a problem. If there are no requirements documents around, you need a vocabulary for why you think something is a problem. Here are some recommendations:

Inconsistent with History
If a product is inconsistent with its history then you might have found a problem. A product's history can include previous versions, patches, claims, etc. If something has changed (and no one told you it was supposed to change) then you might have found a problem.

Inconsistent with Image
Most companies want to have a good image in the marketplace. That means their software needs to look professional and consistent with accepted standards. If a product is inconsistent with image, then what you are saying is "we will look silly if we release this software to the market."

Inconsistent with Comparable Product
If something is inconsistent with a comparable product, you are letting another product serve as your oracle for that test. As long as the comparable product really is comparable, and you want your product to be an alternative to that product, or you desire the same users that product has, then this oracle can be very compelling.

Inconsistent with Claims
If something is inconsistent with claims, then it is inconsistent with requirements, help, marketing material, or something a project stakeholder said in the hallway. A claim can be anything that someone in your company says about the product.

Inconsistent with User Expectation
Inconsistent with user expectation says that this product does not do something that a reasonable user of this product would expect it to do or does not perform a task in a way that they would expect. Using this oracle means you have some idea of who the user is and you have some indication of what they expect.

Inconsistent within the Product
Inconsistent within the product means that something behaves one way in one part of the product, but behaves a different way in another part of the product. This could be terminology, look and feel, functionality, or feature set. All you are doing is pointing out where the product is inconsistent with itself. These are often compelling defects.

Inconsistent with Purpose
The inconsistent with purpose oracle says that the behavior you found is contradictory to what a user would want to do with this software. This is often used in conjunction with inconsistent with claims or inconsistent with user expectations since those tend to indicate the purpose of the software, but it does not have to be. You might be talking about the purpose of a feature.

Final Report

Your project's final report will be comprehensive and it will include all the artifacts and related documentation that you have created and maintained throughout the semester. A recommended organization of your report is as follows (Note: you have some freedom to adjust the format of this report, as you see appropriate)

1. Cover page (course name, team name, team members, semester, project name, etc.)
2. Table of contents
3. Executive Summary
4. Chapter 0: Introduction
    a. Project selection and description
    b. Team structure and member roles
    c. Organization of your report
5. Chapter 1: Testing Strategy
    a. Explain which overall testing approach you have selected (e.g. spec-based, code-based or a combination) and why

    b. Describe various functionality/feature tours you conducted in order to better understand and assess the behavior of the application
    c. Discuss your initial observations of the application including any complex and vulnerable areas identified that needed more testing
6. Chapter 2: Test Case Design
    a. Discuss the design technique(s) you followed in order to generate your test cases such as BVT, ECT, Decision Tables, Exploratory, Data Flow, Path/Branch, etc.
    b. Describe all the coverage metrics you used to establish a sense of testing completeness
    c. Include a table and describe the actual test cases you designed and executed
    d. In case you conducted any code-based testing, please describe any code instrumentation, program graphs and any other related artifacts created
7. Chapter 3: Test Execution and Reporting
    a. Include a list of specific issues or defects found after running all your test cases
    b. Classification and prioritization of these defects (e.g. critical, fatal, minor, design issue, etc.)
    c. Detailed description of all replication steps recorded for each of the above defects for the developers
8. Chapter 4: Conclusions
    a. Briefly describe what you have accomplished during this project and your personal experiences with learning about how software should be tested
    b. Explain how you would be able to effectively use what you have learned from this project as a professional software engineer
    c. Discuss in your own words, how you see the current status and future of software testing
9. Chapter 5: Team organization and roles
    a. Explain the role and detailed contribution of each member in the team
    b. Clearly describe how the overall work was divided and carried out by each member
    c. Provide a brief assessment of how teamwork was effective (or not)