Department of Computing

Curtin University

Software Engineering Testing (SET)

Week 3 Laboratory/Tutorial

The following exercises are intended to be done in a laboratory/tutorial session with a teaching assistant or instructor present. The exercises have been designed to reinforce concepts taught in SET.

1. How are faults and failures related to testing and debugging?

2. For what do testers use automation? What are the limitations of automation?

3. Suppose you were testing a program that does calculations. Consider four development contexts:

- Computer game
- Early development of a commercial product, at the request of the project manager, to help her identify product risks and help her programmers understand the reliability implications of their work
- Late development of a commercial product, to help the project manager decide whether the product is finished
- Flight control software

For each context:

- Why are you testing?
- How should you organize your testing to help you achieve the mission?
- How aggressively should you hunt for bugs? Why?
- How extensively will you document your work? Why?
- Suppose the program has a numeric input field. The spec says it must accept single digits, but not how it should respond to letters. Should you test with letters? What if you're time pressed?

4. The following exercise is intended to encourage you to think of testing in a more rigorous way than you may be used to. The exercise also hints at the strong relationship between specification clarity, faults, and test cases.

   a) Write a Java method with the signature
   `public static Vector union (Vector a, Vector b)`

The method should return a Vector of objects that are in either of the two argument Vectors.

b) Upon reflection, you may discover a variety of defects and ambiguities in the given assignment. In other words, ample opportunities for faults exist. Identify as many possible faults as you can. (*Note:* Vector *is a Java* Collection *class. If you are using another language, interpret* Vector *as a list*).

c) Create a set of test cases that you think would have a reasonable chance of revealing the faults you identified above. Document a rationale for each test in your test set. If possible, characterize all of your rationales in some concise summary. Run your tests against your implementation.

d) Rewrite the method signature to be precise enough to clarify the defects and ambiguities identified earlier. You might wish to illustrate your specification with examples drawn from your test cases.

5. What are some of the factors that would help a development organization move from Beizer's testing level 2 (testing is to show errors) to testing level 4 (a mental discipline that increases quality)?