Test Methodology

Justin Pearson

2014-04-24 Polacksbacken 08-13

Cover Sheet

Problem no.	Solution provided	Max	Your points
1		18	
2		13	
3		10	
4		6	
5		10	
	Total:	57	

Anonymous Exam Code . : . . . . . .

Exam Rubric

All answers to be written in English or Swedish. A mark of 50% is required for a 3, the marks 4 and 5 are evenly distributed in the interval from 50% to 100%. I will not be able to come to the exam to answer questions, if you have trouble understanding a question then please state clearly any assumptions that you have to make to answer the question.

Hjälpmedel:

Pen,pencil, ruler,rubber, dictionary, calculator.

1. General Questions on Testing.

   (a) Describe the distinction between a *Software fault* and a *software failure*. Give examples to illustrate the distinction. (**points 2**).

   (b) Explain in detail the RIP (Reachability, Infection and Propagation) model and its relation to software faults and failures. You should use an example to illustrate your explanation. (**points 4**)

   (c) When testing software there are two important concepts: *software observability* and *software controllability*. Define both concepts (**points 2**) and explain how these concepts are related to the testing of an embedded system. (**points 2**)

   (d) A test case is composed of the following: *test case values*, *expected results*, *prefix values* and *postfix values*. Explain each component of the test case with examples of each. (**points 8**)

2. Consider the following fragment of Java(like) code:

```
public int foobar(int x) {
  int z = 0;
  while(z<15) {
    for(int i=0; i<x; i++) {
       if( i % 3 == 0) {
         z = z + i;
       }
    }
  }
  return(z);
}
```

   Note that `%` the remainder after division operator.

   (a) Draw a Structural Graph that represents the above piece of source code. You must indicate how each node and edge corresponds to the code.(**points 3**).

   (b) For you graph, enumerate a complete set of test paths that guarantee path coverage. (**points 2**).

   (c) Construct a set of test cases that covers all feasible test paths that guarantee path coverage. If the path is not feasible, then say so. (**points 2**)

(d) Enumerate all the prime paths of the graph you have constructed. You must show how your prime paths are constructed. (**points 2**)

(e) Construct a complete set of test paths that cover all the prime paths. (**points 2**)

(f) For each of the test paths above give a test case for the path. If the path is not feasible, then say so. (**points 2**)

3. Logic Coverage. Given that the variables `a,b` are integers, and that `P,Q` are boolean variables, consider the following expression:

```
(P || (a > 5) ) && (  (a < b) ||  ( b<5 && Q) )
```

Note that `&&` is logical conjunction (and) and `||` is logical disjunction (or).

(a) Define *clause coverage* (**points 2**) and for the above expression derive a set of test cases that give clause coverage. (**points 2**)

(b) Define what it means for a clause to determine a predicate. (**points 2**)

(c) There are many variations on how to define *active clause coverage*. Give two definitions and explain the differences. (**points 4**).

4. Input Space Partitioning.

(a) For an input domain define what a partition of a domain is. (**points 2**)

(b) When deriving test cases, why should the blocks in a partition be disjoint? (**points 2**)

(c) The test designer has many possible partitions to pick. What criteria does the test designer use to design partitions? (**points 2**)

5. General Practical Questions on Testing.

(a) Discuss why it is often useful to have as a test engineer somebody who has not actually developed the component under test. (**points 2**)

(b) What is a test plan? Why is one necessary or useful? What sort of things go into a test plan? (**points 4**) (I don't expect you to give a complete description of test plans, but I do expect you to include some of the important items that should go into a test plan.)

(c) Describe test driven development; in particular, describe the meaning of the mantra: "Red, Green, Refactor". (**points 4**)