Software Testing, Verification and Validation

December 16, 2022

Week #14 — Recitation #9

U.PORTO
FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

Mock exams

- MESW, https://moodle.up.pt/mod/folder/view.php?id=93973
- MEI.C, https://moodle.up.pt/mod/folder/view.php?id=93974

Mock questions

(2009 v1)

Grupo 4. [4 valores]

```java
/** ****************************************************
 * Finds and prints n prime integers
 * Jeff Offutt, Spring 2003
 ******************************************************* */
private static void printPrimes (int n)
{
  int curPrime; // Value currently considered for primeness
  int numPrimes; // Number of primes found so far.
  boolean isPrime; // Is curPrime prime?
  int [] primes = new int [MAXPRIMES];

  // Initialize 2 into the list of primes.
  primes [0] = 2;
  numPrimes = 1;
  curPrime = 2;
  while (numPrimes < n)
  {
    curPrime++; // next number to consider ...
    isPrime = true;
    for (int i = 0; i <= numPrimes-1; i++)
    { // for each previous prime.
      if (isDivisible (primes[i], curPrime))
      { // Found a divisor, curPrime is not prime.
        isPrime = false;
        break; // out of loop through primes.
      }
    }
    if (isPrime)
    { // save it!
      primes[numPrimes] = curPrime;
      numPrimes++;
    }
  } // End while

  // Print all the primes out.
  for (int i = 0; i <= numPrimes-1; i++)
  {
    System.out.println ("Prime: " + primes[i]);
  }
} // end printPrimes
```

a) Prepare a test suite that ensures 100% branch coverage. For each test list in tabular form the branches that it covers.

b) Classify each occurrence of each parameter and local variable as **def** or **use**. Construct tables for each variable identifying they **def-use** paths. From the tables generate tests covering as many **def-use** pairs as possible.

Mock questions

(2016 v1)

Group 3. [Graph Coverage. 3.5 points]

Consider the following method that gives the difference between the largest and the smallest integer in an array.

```java
public static int range (int[] v) {
  if (v == null)
    throw new NullPointerException ();
  if (v.length == 0)
    return 0;
  int min = v[0];
  int max = v[0];
  for (int i = 1; i < v.length; i++)
    if (v[i] > max)
      max = v[i];
    else if (v[i] < min)
      min = v[i];
  return max - min;
}
```

a) Draw the control flow graph for the method.

b) For each node and edge, identify all definitions and all uses.

c) Exhibit a path from a definition of variable max to a use of the same variable that is *not* def-clear with respect to max.

d) Describe all du-paths.

e) For variables max and i only, identify a set of du-paths that satisfy All-Defs Coverage (ADC) but not All-Uses Coverage (AUC). Justify your choice.

f) Characterise tests that cover all du-paths identified in **e)**. You do not have to code the tests.

Group 4. [Program Mutation Testing. 3 points]

```
 1  public static int range (int[] v) {
 2    if (v == null)
 3      throw new NullPointerException ();
 4    if (v.length == 0)
 5      return 0;
 6    int min = v[0];
 7    int max = v[0];
 8    for (int i = 1; i < v.length; i++)
 9      if (v[i] > max)
10        max = v[i];
11      else if (v[i] < min)
12        min = v[i];
13    return max – min;
14  }
```

#	Line	Was	Becomes
1	4	v.length == 0	v.length != 0
2	8	i < v.length	i <= v.length
3	9	v[i] > max	v[i] >= max
4	13	max – min	max + min

a) For each mutant induced by the mutations in the table, identify a test that kills the mutant, if possible. Justify your choices.

b) Are there functionally equivalent mutants? Which? Justify your answer.

Recitations’ exercises

- Recitation #2, *category-partition*, https://paginas.fe.up.pt/~jcmc/tvvs/2022-2023/recitations/recitation-2.html
- Recitation #3, *boundary value analysis*, https://paginas.fe.up.pt/~jcmc/tvvs/2022-2023/recitations/recitation-3.html
- Recitation #7, *dataflow testing*, https://paginas.fe.up.pt/~jcmc/tvvs/2022-2023/recitations/recitation-7.html