SENG3320 – Notes

Week Two

Bryce Tuppurainen

White Box Testing Coverage Metrics – Provided Source

**Statement coverage**: **all statements** in the programs should be executed at least once

To measure the effectiveness of statement coverage, follow a branch down the CFG for a given test case, or combination of test cases. Statement coverage is the number of statements that are followed against the number of statements exist.

This is useful in confirming that each of the statements is necessary and will not cause any bugs (at least for this particular test case, this is NOT extensive)

**Branch coverage**: **all branches** in the program should be executed at least once

Similarly, instead of statements instead count the edges in the CFG instead of the nodes. You’re essentially confirming that each of the branches will run to completion and provide the correct result.

**Path coverage**: **all execution paths** in the program should be executed at least once

Instead of using the possible branches working down the CFG as the denominator, instead use the number of possible unique paths that can be followed, i.e. they can deviate even in one branch on the way down and it is ‘unique’.

Note: A combination of the statement, branch, and path coverage provides. Note that statement coverage is a subset of branch coverage, which itself is essentially a subset of path coverage. They all strictly subsume each other.

CFG and Testing Notation

e1 OR e2 OR e3 OR e4 – Edges, running down the program execution path

e1e2e3e4 – Branch/Path, A combination of edges running the whole program provides a path, a branch is when one edge can be taken by opposed to another and doesn’t have to be the whole execution path

t1: {x=3, y=true, s=”myString”} t2: {x=0, y=false, s=”anotherString”} – Test cases

Situations where 100% SBP Coverage Doesn’t Detect Errors

function maxValue(int x, int y) {

if (x <= y) return x;

return x;

}

t1: {x=1, y=1} = 1 PASS

t2: {x=1, y=0} = 1 PASS

100% statement, branch, and path coverage

function sum(int x, int y) {

return x-y;

}

t1: {x=1, x=0} = 1 PASS

100% statement, branch, and path coverage

Software Testing Tools – JUnit and xUnit

Frameworks for various languages such as C++ and Java that allow for simple libraries to import and run more efficient testing code. Such as assertEquals in JUnit, that literally just checks that the method being called is equal to the expected output.

See the lecture notes from week two for information about the methods that are available in JUnit

Testing Terms – Part Two

**Test Case**: A set of inputs to test a particular objective.

**Test Oracle**: The predicted outputs for a given case, it is a part of the test case. Specifically the answer

**Test Driver**: A software framework that can load a collection of test cases or a test suite.

**Test Suite**: A collection or set of test cases