**ERIC WEBB**

**Nova Southeastern University**

**College of Engineering and Computing**

**Fall 2019 - Master Level Course**

**CISC 680 - Software Engineering - CRN – 21741**

**Term Code: Fall 2019 (202020) Course**

**Assignment 4: (Question Set 2) - See Syllabus for assignment % and Due**

**1. What are the attributes of a good software test?**

**Software tests should be able to communicate clearly their objectives and expected results. There should not be any argument or doubt on whether it is an effective test or what the results should be. The intent of a software test should be direct and easy to understand in such a way that a non-technical person can comprehend what the objective and tasks at hand are. A test should be significant that one can understand and grasp the importance of the test and its results. A test could be clear on the results, but one may not understand why these results are important. A good test should be isolated in the fact that it should not be dependent on other tests. If it depends on other tests then results could be skewed depending on the results of each individual test. Test should be automated so that excessive time is not spent creating and running a test. These automated test should also be quick to write and quick to run. These test should be unique in such a way that the results provide confidence that is not offered from other tests. A good software test typically only invokes a portion of code and does not use the code in its entirety. The test will usually deal with a certain portion of code like a single method or a particular piece of business logic. A good test should be order independent so it can be ran regardless of order of other tests, giving the tester a choice on which test to choose. It is no secret that a good test should be re-creatable and be able to be mimicked. It is said that a good test should be thought of in a manner similar to an end user. Thinking outside the box and asking questions are good attributes of a software test. Asking questions like why and why not something occurred or how something occurred can lead to better understanding of the applications functionality. Test results should be easily displayed and readable to the human eye.**

**2. Describe three control structure testing strategies.**

Three examples of control structure testing are Branch testing, Condition Testing, and Data Flow testing. To begin, branch testing is sometimes synonymous with the term “decision testing.” Branch testing is based off of obvious decision statements such as for, while, if, etc. These branch tests can also include more subtle decision statements such as a Boolean, try-catch, and ternary expressions. Essentially they use decisions statement in the program. For these branch tests each condition needs to be executed at least once. An example of this could be explained in pseudo code as follows, if dog is an instance of animal then test bark, else leave dog house. This is an example of a branch test. Next we have loop testing, loops are an integral part of most programs and need to be tested accordingly. There are four types of loops that need to be tested; simple, nested, concatenated, and unstructured. A loop can cause many instances to occur so one needs to make sure that each instance created is properly tested. One popular testing method I use when testing a program is an on change condition. For example, if a loop creates fifteen instances of a dog and the last instance changes to be a cat it alerts me as a false. It is good to test simple loops by testing a few scenarios such as skipping the loop, the n-1 equivalent, and the 1+n equivalent. You should test nested loops by starting in the most inner loop and setting all other passes to their minimum values and then working from the inner loop outwards increasing passes till all loops are tested. If you are testing concatenated loops test them as simple loops if they are independent of each other, if they are dependent of each other test them as you would a nested loop. Lastly when it comes to unstructured loops it is best to redesign the looping structure or the algorithm. When it comes to basis path testing you will create a test case scenario for every method , statement , or function in your design at least once. This can be done in a flow graph notation such as basically a top down approach to testing, Next you can use a flow graph notion that breaks the testing down into regions where nodes and edges meet. An edge being the control flow itself, a node being the procedural function. A node and edge have to bind and meet, this becomes a region. This way you can have different parts of you code tested that are similar to other parts of your code deriving away from the top down fashion of a flowchart. When testing independent program paths it is important to calculate the cyclomatic complexity of how many paths in your program to test for. One more useful tool for basis path testing is a graph matricie. These are a good rendition of nodes line weight along with the likely hood of how often they might occur.

**3. Why is regression testing an important part of any integration testing**

**procedure?**

**When testing code it is imperative to implement a form of regression testing. This is a part of any integration testing for good reason. Integration testing aims to test individual parts of code but as a whole. While testing components of code individually you might not get the same results as if you tested them all together. This can test from the main program down toward the more atomic pieces of code called top down integration, also it can be tested from the smallest units up towards the main program called bottoms-up integration. When it comes to Objected Oriented Programing this becomes more difficult since code is not ran in hierarchal fashion but can be resolved using methodologies such as thread and use-,based testing. This helps verify the program works as a whole instead of verifying small individual parts to make up as the whole. Every time a new piece of code is added, new input output paths are created establishing new data flows. You must go back in your code and verify that your new changes not only work, but they do not break any existing logic or functionality. This is why regression testing is so crucial to integration testing because it verifies that new code has not broken the old code. This can be tested in a variety of ways within the regression test suite. For example, a test suite can use a representative test to verify all the programs functionality. They can perform additional tests on code that is more likely to be changed in a more specific format. From that they can also perform tests on components that have indeed been changed. If it were not for regression testing new functional code might be added but the now broken older functionality would not be discovered till it was too late. Imagine adding a nice new user interface to a banking system, but breaking the older functionality of allowing users to withdraw funds. This would cost the banking company a lot of grief with complaints and maybe even law suits. From this example you could see how regression testing is not something to be overlooked and sometimes you must go back and test very basic functionality of code no matter how simple or small. This is why regression testing is crucial to any integration testing.**

**4. What are the key differences between validation testing goals and**

**acceptance testing goals?**

**5. Describe the how test cases are derived from behavior models to**

**facilitate interclass testing?**

**6. List the components of a formal specification language and describe their**

**roles.**

**7. Describe the process of writing a formal specification for some system**

**function.**

**8. Technical testing metrics fall into two major categories. What are they?**

**9. Describe the five activities associated with the software measurement**

**process.**

**10.Describe the role of class-oriented metrics in assessing the quality of an**

**OO system.**

**11.Why is it important for software developers to make use of measurement**

**to guide their work?**

**12.Why is the "make-buy" decision and deciding whether to outsource**

**software development an important part of the software planning**

**process?**

**13.Describe the process of building a risk table.**

**14.What is forward engineering?**

**15.What characteristics need to be exhibited by organization to improve its**

**software process?**