Proposal

Xiang Li

932-514-926

Testing is currently a widely used way to find software application bugs which can provide a robust and high quality code in the software industry. There is an opinion in the academic field that labor based testing costs manual intensity. This kind of testing is inefficient and slow. Alternatively, automatically generate a bunch of test case is a more accurate, efficient and fast method to satisfy demanding of testing. In this case, most scientists aim to reduce the manual test. So auto-test generation is considered an important part of software testing. This method requires tester construct an algorithm for code to generate testing cases to detect bugs in a software application. According to the article found on the Internet, test generation techniques belong to the black-box testing category. These techniques are useful during functional testing where the output of the code should correctly match the given requirements.

Nowadays, there are many kinds of test generations. The most useful generation algorithm is random testing which we have used last term in TSTL test. Random testing is a black box software testing technique. This testing can generates test cases by randomly and independently. The only thing that a tester should do for this test is check the output is pass or fail. This is an efficient and easy testing method.

To compare with the random test, I am interesting right now is a test generation algorithm called anti-random testing. Now the anti-random testing has been treated as a very beneficial method for creating effective test cases. The basic premise of anti-random testing is, in order to achieve higher coverage, that the algorithm can choose new test vectors that are as far away from existing test inputs as possible. In the paper “ Fast Antirandom (FAR) Test Generation ”, the author Andre Bai, Tom Chen et. present and empirically evaluate a technique to compute the distance far away from the existing test inputs. In author’s idea, fast anti-random test generation has a more efficient method to

generate test patterns, which are suitable for large input vectors and long sequences of tests.

The paper “ Fast Antirandom (FAR) Test Generation ” presents a step to show how does anti-random test generation calculate the next test state position. The authors also illustrate with an example how to generate a six bit anti-random vector when given five input test vectors. However, this example is a six bit anti-random vector. I am not quit sure how to apply this algorithm to python code testing. So the next step of the project is discuss this algorithm with professor and try to implement this algorithm with python.

As the authors explain that what a good perspective is, comparing with pure random testing after 10 iterations, the branch coverage of anti-random test generation can reach to 84.43%. However, pure random testing is just 76.48%. I am have confident that anti-random test generation will give me a good branch coverage.

Next step of the project is reading the paper again and evaluate the distance calculation algorithm. Now, I am not familiar with the TSTL SUT API. I will read the paper “The Template Scripting Testing Language” which gives me a list of SUT API. Finally, I wish I could implement the anti-random test generation successfully.

Reference:

[1] A. von Mayrhause, T. Chen, A. Hajjar, A. Bai, and C. Anderson, "Fast antirandom (FAR) test generation," IEEE, 2014, pp. 262–269. [Online]. Available: http://ieeexplore.ieee.org/xpl/login.jsp?tp=&arnumber=731625&url=http%3A%2F%2Fieeexplore.ieee.org%2Fxpls%2Fabs_all.jsp%3Farnumber%3D731625. Accessed: Apr. 20, 2016.