ISSN 2278-3091
Volume 8, No.2, March - April 2019
International Journal of Advanced Trends in Computer Science and Engineering
Available Online at http://www.warse.org/IJATCSE/static/pdf/file/ijatcse20822019.pdf
https://doi.org/10.30534/ijatcse/2019/20822019

Software Testing: The Generation Tools

Dr Leelavathi Rajamanickam[1], Nurul Azlia Binti Mat Saat[2], Siti Norbaya Binti Daud
[1]SEGi University, Malaysia, leelavathiraj@segi.edu.my
[2]SEGi University, Malaysia, azliamatsaat@segi.edu.my
[3]SEGi University, Malaysia, norbayadaud@segi.edu.my

ABSTRACT

Software testing is an area in software development life cycle in terms of manpower and cost. Much research has been done in order to reduce the cost and manpower to fix the errors and bugs. Generating test cases from different phases automatically improves the quality. Generating test cases at an early stage is efficient than having it after development phase. The effort and time spend on finding and fixing errors is reduced than having it after development. At the future stage fixing the errors results in enormous code correction and consumes more time and effort. In this paper, the different paradigms of testing techniques for test cases are generated and can investigate their coverage and associated capabilities.

Key words: generation-testing tools, software-testing, security, testing-types.

1. INTRODUCTION

Software testing is to perform various tests on a data to obtain a result, by comparing the original value with tested value. It is not only fining errors or bugs in the software but refining to get a good quality of the software [2]. The phases of system development life cycle require time, effort and cost. To get a bug free or error free program software testing is very important phase for development. It also finds faults in the design [1]. To ensure high quality software, software testing is performed. This is done to detect defects in the System under test that can cause software failures. The process is time-consuming and complex. It can consume about 50% of the total cost. It can also be defined as the process of validating and validating System under test to meet technical and business expectations. The software testing generation has been continuing from many years. The growing maturity in programming and the technologies forces to test the programs and to optimize. Early programmers were responsible for testing, and the program should be high reliable. Testing teams are organized to produce a key checklist for the whole project. As on change of period, it was detected that testing teams can identify the errors or bugs and give a clear structure. In current perspective of business, it needs more relaxed scenarios and identify the cut cost methods for getting a good quality. Testing began as manual testing in the 19th century and evolved in the 21st century with automated testing of hybrid/keyword frameworks. In this day and age, testing software projects typically costs about 30-50% of the project. As the complexity in the software rises automatically the process of testing also increases in cost and this leads to high level of risk. If the software testing process is of low quality then it leads to system shutdown, reframe of work, high maintenance cost etc. Software validation is used to check that the system under test is compliant and similar to the structure test. Validation is done by running system under test, which is similar to functional testing. Generally speaking, there are four types of software testing (a) black box testing (b) white box testing and (c) greybox testing, and functional testing. Software tests can be performed manually or automatically using test tools. Automated software testing has been found to be better than manual testing. Recently, more and more test case generation tools have been available. These tools use a variety of methods to perform their tasks.

Software testing also works on the product's security parameters, which are the most recent priority to avoid any type of vulnerability and backdoor hacking that could lead to development

2. REVIEW

To identify the defects, errors or bugs in the process of programming or development, software testing is required. This is required to make sure that customer is satisfied with the application and it is reliable. And also, it is required to ensure the quality of the product. By providing good quality products to the customers, their trust can be achieved. To provide a good quality product or an application to the customers testing is important. To have accurate result with low maintenance and reliable cost, testing is to be performed so that the application does not have any errors. If the application has errors then in further stages of development, many consequences are to be faced.

2.1 Quality

First of all, your products are of high quality, which is very important to customers. Customers will definitely pay more for quality. What's more, by giving good products to the customer, the customer is satisfied and there will be a good relationship between both the parties and can have brand for the products which will have higher priority.

2.2 Trust

Software testing provides confidence in product development and quality assurance. Quality assurance performed during software testing provides business confidence in the realization of business and user requirements.

2.3 Increase Success

To have increase in success such as profit good products are to be delivered and it does not require much advertising because it get publicity very fast. The most important and best active directory tool to use is through the word-of-mouth. Providing products that have been rigorously tested and quality tested means respecting customers. This will help retain existing customers and gain new ones. The testing phase will not only bring profits, but also reduce existing costs. In future this concept saves money, the software that is going to sold doesn't have to be constantly fixed. It is of compromise on quality that ends up costing more than the plan. The other benefit is that the bugs are eliminated before the product is given to customer. This prevents in having unsatisfied customers and spending unnecessarily leads to customer support. The other benefit, using an automated software test solution reduces service costs.

2.4 Reliability

The probability of the software operations will operate without any errors, bugs or failures for a specific period of time is software Reliability [16].

During software testing, product quality is measured by a variety of parameters, such as functionality, performance, security, availability, and acceptability. All parameters of product results are verified to make the product more reliable.

2.5 Security

Software testing also works on the product's security parameters, which are the most recent priority to avoid any type of vulnerability and backdoor hacking that could lead to development of source code.

3. TYPES OF SOFTWARE TESTING GENERATION TOOLS.

3.1 Combinatorial Testing

Failures are detected by combinatorial testing where the parameter interactions are triggered by software under test, an array test suite is generated by the sampling mechanisms. Combinatorial test is used to select a set of input cases with high failure indication capability. It can reveal 90% of programming errors. This is done by specifying the scope of each input. Then analyze the system under test to find out the degree of interaction between these inputs (T). It then generates a test set where every possible combination of T variable is in the set. The minimum number of test cases generated should be a multiplication of the possible values of the maximum T input variable. Combinatorial Testing research can include the following (a) combinatorial testing for modeling, (b) enhancing the existing test suite generation algorithm, (c) improving analysis of testing result, (d) exploring the application of combinatorial testing to different levels of testing and additional types of systems, (e) understand limitations and strengths of combinatorial testing, and (f) combining other testing techniques with combinatorial testing.

3.2 Acts

Acts is a combinatorial test case generation tool. It supports 1 to 6 ways of interaction, using 3 different algorithms:

Base case: Where each possible value of a parameter has to appear once in the test set.

IPOG: In-parameter-order-generation is used by utilizing the greedy search technique to calculate maximum interaction element combinations coverage.

IPOG-D: In-parameter-order-generation is a combination of IPOG with a recursive called as IPOG-D; it is to reduce the search space during generation of the test suites.

3.3 Genetic Algorithm

Genetic Algorithm also known as EvoSuite, it uses an evolutionary approach to generate test suites. The tool targets code coverage standards, such as branch coverage. In addition, it reduces unit tests to improve readability. It can be executed using command line prompt and it also has plugin to integrate it in Maven, IntelliJ and Eclipse. The tool was the result of a 2010 study by Gordon Fraser and Andrea Arcuri. Many researchers see EvoSuite as the primary reference tool in the search-based software testing literature.

3.4 Randoop

Random Search also known as Randoop, it creates unit tests for java classes in JUnit format. It can be used for two purposes (a) to find bugs in your program (b) to create regression tests that will alert when the program's behavior changes in future. Randoop combines random test generation with test execution to produce an efficient method. It showed errors that were unknown and are widely used libraries. The use of randoop continues to be used in industries such as ABB Corporation. This random search generates a feed-back directed from random test generation. This technique is random but clever, generating a sequence of methods or constructor calls for the class under test. Randoop increasingly creates a continuous method by selecting randomly selecting the call method that is to be applied and

parameters are selected based on the previously constructed sequence. In the new method a set of new instruction are executed and it is checked based on the contracts. The sequence that results in a breach of contract is output to the user as a test for breach of contract. Sequences that show normal behavior are stated as regression tests. The set of sequences that produce illegal behavior are ignored. The normal sequences generate new sequences and to extend a sequence that already is in a distorted state.
A test case can be generated from the feedback-directed aspect by randoop test generation; it can perform both systematic and unsystematic random test generation [8] and covers error detection. The four small data structures that are non-trivial, randoop achieved a higher or equal block and predicate coverage than model checking and unsystematic random generation. Randoop found many unknown errors are not found by either model checking or unsystematic random generation. You can always re-run the randoop to check for new errors, generate tests for newly written code, or regenerate tests after code changes, resulting in ideal behavior changes.

3.5 JUnit Generator

It is a unit testing framework and it is important for the development of test-driven development, and is the family of unit testing frameworks which are collectively known as xUnit that originated with SUnit.

Although ACTS generated a bigger number of test cases, it is the fastest in test case generation and the most appropriate in statement coverage, branch coverage, and mutation scoring. The number of test cases consumed by EvoSuite is much smaller, but this is due to the relationship between mutation scores and generation time. But it achieves reasonable representation and branch coverage. Randoop spent the most time generating test cases, but achieved higher variance scores than EvoSuite, and fewer than ACTS. But that comes at the expense of reporting and branch coverage. JUnit Generator was ranked the lowest in statement coverage, and branch coverage.

4. MATERIALS AND METHODOLOGY OF SOFTWARE TESTING GENERATION TOOLS

Software testing generation tools is testing the whole application in terms of behavior, front-end and back-end functionalities along with load testing. A set of activities are performed through manual testing, automation testing or both, it aims to show errors in the software application. It covers the entire end to end functional testing of a software application. This testing helps the team to evaluate and enhance the software quality and at the same time, it the entire software application is being tested with functional testing, which also evaluates and enhances the quality of software. And reduces the testing cost. The whole bundle of tests can be performed by using application that covers frontend or GUI testing, backend testing or database testing, load testing, etc.

The above main types of testing methodologies can be used to assure the required level of testing depending on the type of application.

4.1 Functional Testing and Blackbox Testing

For any software application using functional testing methodology for the given set of inputs, the original result is matched with the expected result. The program tester is not aware of the code; therefore, it is known as blackbox testing. Blackbox testing is mostly used for three types of testing, i.e. functional testing, non-functional testing, and regression testing. The following testing strategies are followed by blackbox testing. They are equivalence class approach, boundary value approach, decision table approach, and state transition tables approach.

4.2 Unit Testing and Whitebox Testing

In whitebox testing, the testing is done by the programmers after the coding part is completed for any application module. Whitebox testing follows the coverage-analysis, path -coverage, dead-code-analysis, code-duplication-analysis, infinite-loop-analysis.

4.3 Greybox Testing

Greybox testing is a combination blackbox testing and whitebox testing. The program tester applies the mixed strategies that include both blackbox as well as whitebox testing strategies.

5. CONCLUSION

Software testing is a very important component method which takes a huge amount of the project's time, resource, and budget. Testing's main purpose is to force the system to come out with any kind of system failure that could be found and eventually fixed. In addition to failure detection, it ensures that bugs in specific functionalities have already been fixed which increases the confidence and quality of the project, though it doesn't mean that the function is bug-free.

REFERENCES

1. Leelavathi Rajamanickam. **Software Testing, Analysis and Objectives**, International *Journal of Advanced Trends in Computer Science and Engineering (IJATCSE)*, Vol. 3, pp. 01-04, 10 Oct. 2014.
2. Leelavathi Rajamanickam. **Tools for Object Oriented Software**, *International Journal of scientific research and management (IJSRM)*, Vol. 2, pp. 01-04, pp. 1205-1208, Aug. 2014.
3. Ghahrai. A, M. **Seven Principles of Software Testing — Testing Excellence.** Dec. 2018.

4. Awan. k, Shah. J, Akram. A, Gupta. N, Kim. H, Adke. R and R. Unbehaven. **Types of Software Testing: Different Testing**, 2018.
5. D. Graham, E. V. Veenendaal, I. Evans and R. Black. **Foundations of Software Testing**: ISTQB Certification Thomson Learning, 2007.
6. Bertolino. A. **Software testing research and practice** *ASM'03 Proceedings of the abstract state machines 10th international conference on Advances in theory and practice,* In Proc, pp. 01-21, Mar. 2003. https://doi.org/10.1007/3-540-36498-6_1
7. Myers, Glenford J. **The art of software testing**, *The Psychology and Economics of Program Testing,* 2nd ed. NewYork: Wiley, c1979. Ch. 2, pp. 10-21.
   Peter Sestoft, **Systematic Software Testing**, Vol 2, 2008-02-25, pp.1-17.
8. Programming Research Ltd. **Static and Dynamic Testing Compared**. *PR:QA White Paper Series: WP1.1*, pp. 1-4.
9. U. Rueda, R. Just, J. P. Galeotti, and T. E. J. Vos. **Unit Testing Tool Competition**. *in 2016 IEEE/ACM 9th International Workshop on Search-Based Software Testing (SBST).* May 2016. pp. 19–28 https://doi.org/10.1145/2897010.2897018
10. A. Arcuri, J. Campos, and G. Fraser. **Unit test generation during software development: Evosuite plugins for Maven, IntelliJ and Jenkins**, *in IEEE International Conference on Software Testing, Verification and Validation (ICST). IEEE Computer Society*, 2016, pp. 401–408. https://doi.org/10.1109/ICST.2016.44
11. BYRCE, R. C. AND MEMON. **Test suite prioritization by interaction coverage**. *In Workshop on Domain Specific Approaches to Software Test Automation (DOSTA'07).* ACM, New York, 1–7.
12. COHEN, M. B., SNYDER, J., AND ROTHERMEL, G. **Testing across configurations: Implications for combinatorial testing**. *SIGSOFT Software Engineering*. Notes 31, 6, pp. 1–9. https://doi.org/10.1145/1218776.1218785
13. Kuhn, D.R.; Kacker, R.; Lei, Y. **Practical Combinatorial Testing**. *National Institute of Standards and Technology,* October 6, 2010. https://doi.org/10.6028/NIST.SP.800-142
14. Zamli, K.Z., Othman, R.R., Zabil, M.H.M. **On Sequence Based Interaction Testing**, *IEEE Symp. On Computers and Informatics, IEEE*, 20-23 Mar. 2011, pp. 662-667. https://doi.org/10.1109/ISCI.2011.5958995

16.D. HemaLatha, P. PremChand. **Estimating Software Reliability Using Ant Colony Optimization Technique with Salesman Problem for Software Process**. *International Journal of Advanced Trends in Computer Science and Engineering (IJATCSE),* Vol 7, No.2, pp. 20-29. https://doi.org/10.30534/ijatcse/2018/04722018