Automated Test Case Generation in Software Testing

Anik Acharjee[*], Dr. Amar Singh[*]

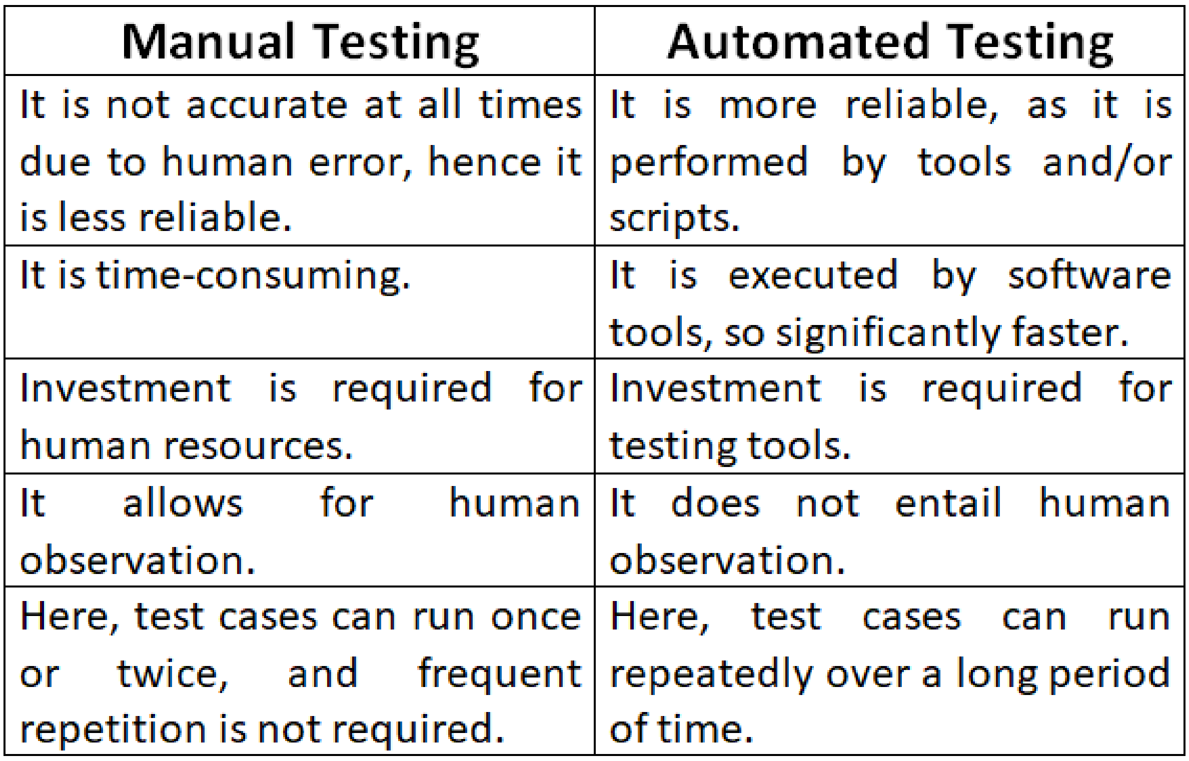
[*]School of Computer Applications, Lovely Professional University, Phagwara, Jalandhar, India.

***Abstract*— Testing is the crucial part in Software Development Mechanism. When software is developed by the developer, it needs to be tested appropriately at the commencing stage. So, test case generation needs to be proficient. But, this test case generation consumes more time gradually when the software expands with time. In the literature, it is found that Manual Test Case Generation acquires infinite time. Such time cannot be provided to accomplish the task of developing the software by the developer. Not only this, it increases the cost of the software directly or in-directly. Due to which, the whole process needs to be computerized eventually. Here, in this paper, several automated test case generation has been studied properly for further research work.**

***Index Terms*— Test Case Generation, testing, automation, test cases,**

I. INTRODUCTION

Testing is the essential and mandatory task. It should be achieved at the commencing stage of the Software Development Mechanism. In early 1980s, the software developer used to develop the software. It is then manually tested by the tester. The tester utilizes the Manual Testing Mechanism. It has been analyzed gradually that such mechanism consumes more time and costs. This effects the overall expansion of the software, directly or in-directly. Software Testing Automation has different rewards – 1) a way of storing domain/project/task Knowledge, 2) proof of completion of testing and 3) execution speed. Computerized Software Testing safeguards reasonable time and worthy money. It improves reliability. It also shoot-up Test Coverage. Not only these, it also improves Team Morale. It removes human error. It expands testing speed without sacrificing quality. It does not require human interference. Manual Testing can become boring and hence error prone. These two types of testing mechanism can be explained further with a suitable image -



Manual Testing	Automated Testing
It is not accurate at all times due to human error, hence it is less reliable.	It is more reliable, as it is performed by tools and/or scripts.
It is time-consuming.	It is executed by software tools, so significantly faster.
Investment is required for human resources.	Investment is required for testing tools.
It allows for human observation.	It does not entail human observation.
Here, test cases can run once or twice, and frequent repetition is not required.	Here, test cases can run repeatedly over a long period of time.

Figure – 1 – Manual Testing v/s Automated Testing

II. LITERATURE SURVEY

The author in [1] discussed about Dynamic Many-Objective Sorting Algorithm (DynaMOSA). This algorithm can be used in context to coverage testing. It has been proved that this approach is far better than MOSA (Many-Objective Sorting Algorithm). DynaMOSA has shown 28 percentage of branch coverage as compare to MOSA. It has also shown 27 percentage of mutation coverage as compare to MOSA.

Shahbazi A et. al. in [2] focused on black-box string test case generation methods. Two identical functions are effectively used. Enormous string distance functions are incorporated as one of the objective. String length distribution is the other objective. When both the objectives are used simultaneously with a multi-objective optimization algorithm, string test cases are generated. These string test cases outperform the traditional methods used earlier.

Shan L et. al. in [3] presented an approach called data mutation. In the research, it is found that it can manipulate large number of test data. This approach is cost effective and can predict large percentage of errors and faults.

Nebut C et. al in [4] utilized use case mechanism for automated test case generation. It has been implemented on different object-oriented embedded software [4]. Here, statement coverage has been implemented for the generated tests.

Huang H et. al. in [5] evaluated test case generation in fog computing programs. The author proposed a mathematical model. This is based upon path coverage. Test-case-path relationship matrix is also proposed for generated test cases.

Matinnejad R et. al. in [6] researched with Simulink Models. Here, the author proposed black-box test generation approach. This approach is applied on Simulink models for Test Generation and Test Prioritization [6]. It is basically applied on meta-heuristic search.

Bures M et. al. in [7] invented a framework for automating test cases. The system that needs to be tested is used by screen-flow-based model. In this re-search, it is found that this model inherence the whole process for the tester. The results show that this proposed framework has improved the time efficiency.

Yousaf N et. al. in [8] inaugurated model-based testing approach for Interaction Flow Modeling Language (IFML) Models. It is basically evaluating test case generation for the Web User Interface (WUI) of IFML Models [8]. Not only this, a generator tool called Model-Based User Interface Test Case (MBUITC) has been incorporated as a part of the research. This approach can be more useful in the commencing stage.

Arrieta A et. al. in [9] presented multi-objective test generation and prioritization approach. This can be used for testing and debugging industrial Cyber-Physical Systems (CPSs). Here, a fitness function is used with four objectives. It is designed with various cross-over and mutation operators.

Singh R et. al. in [10] focused on object-oriented (OO) systems. It has been identified that generating Test Cases for such systems is one of the tedious task. The proposed technique has been implemented after analyzing two unique and adequate case studies. One is the sample of Sequence Diagrams (SDs) and another is the SD of ATM (Automated Teller Machine).

III. Conclusion –

In this paper, various automated test case generation have been reviewed. It is found in the literature that manual testing degrades the productivity of the software as compared to automated process. Automated mechanism of evaluating test cases and prioritization has come the first priority for the testers in the Software Testing domain. Automated Test case generation has many more benefits and comfort while testing large and complex software in today's robotic world.

REFERENCES

[1] Panichella A, Kifetew FM, Tonella P. Automated test case generation as a many-objective optimisation problem with dynamic selection of the targets. IEEE Transactions on Software Engineering. 2017 Feb 2;44(2):122-58.

[2] Shahbazi A, Miller J. Black-box string test case generation through a multi-objective optimization. IEEE Transactions on Software Engineering. 2015 Oct 7;42(4):361-78.

[3] Shan L, Zhu H. Generating structurally complex test cases by data mutation: A case study of testing an automated modelling tool. The Computer Journal. 2009 Aug;52(5):571-88.

[4] Nebut C, Fleurey F, Le Traon Y, Jezequel JM. Automatic test generation: A use case driven approach. IEEE Transactions on Software Engineering. 2006 Mar 27;32(3):140-55. [5] Huang H, Liu F, Yang Z, Hao

Z. Automated test case generation based on differential evolution with relationship matrix for IFOGSIM toolkit. IEEE Transactions on Industrial Informatics. 2018 Jul 18;14(11):5005-16.

[5] Matinnejad R, Nejati S, Briand LC, Bruckmann T. Test generation and test prioritization for simulink models with dynamic behavior. IEEE Transactions on Software Engineering. 2018 Mar 1;45(9):919-44.

[6] Bures M, Frajtak K, Ahmed BS. Tapir: Automation support of exploratory testing using model reconstruction of the system under test. IEEE Transactions on Reliability. 2018 Mar 2;67(2):557-80.

[7] Arrieta A, Wang S, Markiegi U, Sagardui G, Etxeberria L. Employing multi-objective search to enhance reactive test case generation and prioritization for testing industrial cyber-physical systems. IEEE Transactions on Industrial Informatics. 2017 Dec 29;14(3):1055-66.