On the applicability of Artificial Intelligence in Black Box Testing

Esha Khanna
Assistant Professor, Information Technology, DAV Institute of Management, Faridabad, India
eshakhanna30@gmail.com

***Abstract*—Efficient and thorough testing is essential to create quality software. In some cases, software code may not be available in testing phase. In such scenarios black box testing is used. The work presents various black box testing techniques. A systematic literature review has been carried out according to the guidelines proposed by kitchenham. The work discusses the applicability of artificial intelligence techniques in black box testing.**

Keywords- Software Testing; Black Box Testing; Artificial Intelligence; Test Cases.

I. INTRODUCTION

Testing is a process of executing software with the intent of finding errors [1]. According to IEEE (1986, 1990), "Software testing is the process of analyzing a software to detect the differences between existing and required conditions (that is, bugs) and to evaluate the features of the software system [2]." Efficient and thorough testing improves the software quality and reduces the maintenance cost. Testing is an umbrella activity and is carried out in all the phases of software development life cycle. Testing is classified as static or dynamic [3]. In static technique, testing is performed without actual execution of program. As execution environment is not required therefore the technique requires fewer resources. Static testing techniques include reviews, inspection and walkthroughs. Dynamic testing techniques run the software code under test to find out the difference between required and existing conditions.

Testing is generally carried out by a group of people which are not the part of development team. This is due to the fact that humans are unwilling to find errors in their own work. In some scenarios software code might not be available to testing team. In such cases black box testing technique can be used. Black box testing is testing that ignores the internal mechanism of a system or component and focuses solely on the outputs generated in response to the selected inputs and execution conditions [4]. The work reviews various black box testing techniques. An extensive literature review has been carried out to find the gap in the existing literature. The review has been carried out using the guidelines of Kitchenham [5]. The work also examines the role of artificial intelligence techniques in black box testing

The goals of the paper are as follows.

- To review black box testing techniques.
- Classify black box testing techniques based on Artificial intelligence.

The paper has been organized as follows. Section two of the work discusses black box testing. Section three reviews various black box testing techniques. Section four presents artificial intelligence techniques in black box testing and section five concludes.

II. BLACK BOX TESTING

Black box testing technique intents to find errors in a module without taking into account internal working of software. Black box testing rely only on the input/output behavior, without any assumption about what happens in between the "pins" (precisely, the entry/exit points) of the system [6]. According to IEEE 1990, "black box testing ignores the internal mechanism of a system or component and focuses solely on the outputs generated in response to selected inputs and execution conditions [4]". Black box testing relies on the functional requirements of module under test. It can be applied to all the software testing levels i. e. unit, integration, system, acceptance and regression. Objective of black box testing is to find that how well a system confirms to specified requirements. Black box testing techniques are used to find missing or incorrect functionalities of module under test.

Black box testing process is described as follows. Test cases are created and are fed to system under test. Actual results are then compared to the predicted results in order to find functional errors. Figure 1 depicts Black box testing procedure.

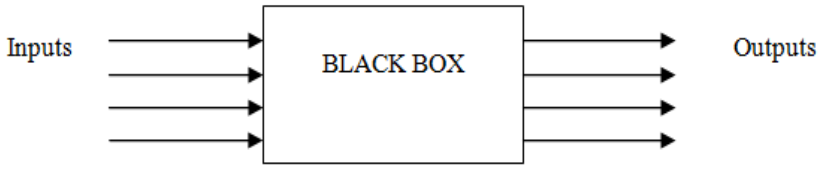

Figure 1. Black Box Testing

Crafting of good test cases is one of the most important tasks in black box testing. Test case is defined as a set of test inputs, execution conditions, and expected results developed for a particular objective, such as to exercise a particular program path or to verify compliance with a specific requirement [7]. According to IEEE std 829-1983 test case is a documentation specifying inputs, predicted results, and a set of execution conditions for a test item [8].A good test case is a one which has a high probability of finding maximum number of errors. Test cases must be comprehensive and creative in nature. A good test case has a high potential to improve software quality and reduce post maintenance costs. Test cases in a test suite must be designed in such a way that execution of test suite covers functionality of software. Both individual units and interaction among the units must be thoroughly tested. Figure 2 shows structure of a test case.

Test Case ID	Module ID	Description	Inputs	Expected Outputs

Figure 2. Test case structure

III. BLACK BOX TESTING TECHNIQUES

A. *Conventional Black Box Testing Techniques*

Black box testing techniques are classified as Boundary Value Analysis, Robustness, Equivalence Partitioning method, Decision Table based, State Table Based and Error Guessing [3,9].

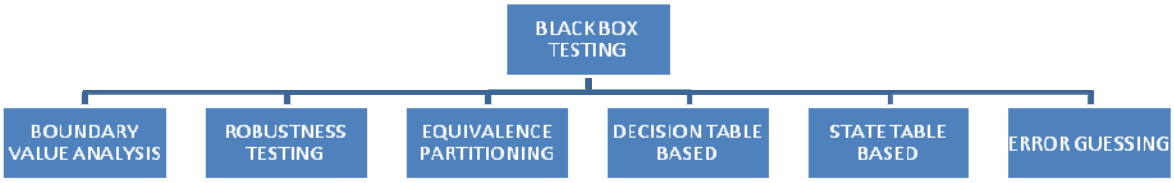

Figure 3. Conventional Black Box Testing Techniques

- *BOUNDARY VALUE ANALYSIS:* In this technique, test cases are selected on and near the boundary of the input domain [10,11]. The technique is based on the fact that boundary values of input domain have higher tendency to detect an error. For n input variable, boundary value analysis technique generates 4n+1 test case [9].
- *ROBUSTNESS TESTING:* It is an extension of Boundary Value Analysis technique. The technique takes into account outside values from input domain along with the values on and near the boundary [10, 11]. Software responses are therefore tested for both valid as well as invalid inputs. Robustness testing technique generates 6n+1 test case for n input variable [9].
- *EQUIVALENCE PARTITIONING:* In this technique, input domain is partitioned into equivalence classes in such a way that test cases of same class generate same output results. The technique strives to touch the completeness of the testing domain without executing the redundant test cases. Equivalent classes are created for both valid and invalid test cases [10, 11].
- *DECISION TABLE:* Decision tables are used to represent the logical relationship between conditions (inputs) and actions (outputs). Test cases are then derived by taking every possible combination of condition and action [10, 12].
- *STATE TABLE BASED:* The technique selects test cases covering all the possible states and transactions on them. This technique is suitable for transaction processing, embedded and real time systems [10, 12].
- *ERROR GUESSING:* The technique aims to guess the hidden bugs and errors which do not fit into any of earlier defined situations. Error guessing is based on tester's expertise along with the history of errors discovered in earlier project [11].

B. Other Techniques

In the work by Blanco et. al. [13] scatter search metaheuristic technique was proposed in order to generate test cases of Black box suite. The work used branch coverage criteria to generate test cases. Proposed technique was evaluated on 13 benchmark programs. The work also compared the technique with GA, TSGen and EDA test case generators.

Frezza et. al. [14] proposed graph data model based technique in order to generate automated test cases for black box testing. Test cases were crafted by capturing the relationships between design and requirement phases. The proposed technique was evaluated using floating point arithmetic and logical unit examples. Hu and Lin [15] used unification mechanism along with constraint solving mechanism in order to generate test cases for black box suite. The proposed structure was evaluated on UML defined java methods.

Murnane and Reed [16] proposed a mutation based technique in order to generate test cases for black box testing. The work reported a case study on two different programs comparing mutation based technique, equivalence class testing and boundary value analysis technique. The reports resulted that test cases generated using mutation based technique were efficient and effective in black box testing. In the work by Mumtaz and Sadiq [17] , a tool was developed in order to automate black box test case generation process. The work validated the tool using line equation problem and concluded that robustness testing techniques are superior to boundary value analysis.

Chen et. al. [18] combined mirroring and ART technique for black box testing. Mirror ART technique was applied on square's input domain and results were compared with ART technique. Results presented that proposed technique is more cost effective than ART. In the work by Chan and Yu [19] partial dynamic analysis was used in order to limit the number of test cases in black box test suite. The technique was examined on path based methods.

Kanatamnehi et. al. [20] proposed a dynamic measure "potential of a branch" to improve on the coverage and efficiency of BBT. Potential of a branch was calculated by merging structural and coverage information. In order to increase the branch coverage magnifying branches approach was used. Four different sized programs triangle, calendar, roots and max were evaluated using the proposed technique and results were effective.

Verma and Karambir [21] finite automata based technique were proposed in order to perform black box testing for component based software. The work proposed bbt by using DFA and NFA techniques. The work was evaluated using five UML diagrams of online shopping catalog.

The number of test cases of a black box test suite may become colossal. Due to limited time and resources all the test cases may not be executed. Test cases are therefore prioritized so that important test cases that have a high probability of detecting errors are executed first. In the work by Noguchi et. al. [22], black box test cases were prioritized based on test execution history of similar products. The work took into account ant colony optimization technique in order to prioritize black box test cases. The proposed framework was simulated using two products and results in effective testing.

In another work requirement analysis and design specification were used in order to prioritize black box test cases [23]. The work presented an algorithm for test case prioritization that creates module description document for each input and output module. Specifications gathered in module description document were then used to extract the range of each input and output, which were further used generate test cases.

IV. ARTIFICIAL INTELLIGENCE BASED BLACK BOX TESTING TECHNIQUES

Artificial intelligent techniques are classified as Genetic Algorithms, Neural Networks and Fuzzy Logic.

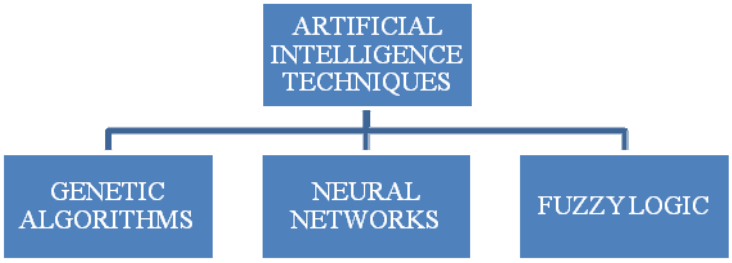

Figure 4. Artificial Intelligence Techniques

A. Genetic Algorithms

Genetic Algorithms are adaptive heuristic search algorithms which mimics the process observed in natural evolution [24]. Genetic Algorithms are used to solve optimization problems. They follow "survival for the fittest" principle laid by Charles Darwin. Genetic algorithms are used in black box testing in order to enhance the quality of test cases. Selection, crossover and mutation are the basic operations of genetic algorithms [24].

Last et. al. [25] introduced Fuzzy-based age extension of genetic algorithm (FAexGA) approach in order to automate test case generation for black box testing. FAexGA included test cases that has higher probability to find errors and eliminated the test cases that have no probability to expose an error. The proposed approach was demonstrated on complex Boolean Expression. Results depicted better performance using proposed approach.

In the work by Fisher and Tonjes [26] a Genetic algorithm based automated test data generator for BBT was proposed. The work used micro genetic algorithms as filters to enhance the test data quality. The work examined GA components in order to generate qualitative test data for Black Box testing environment.

B. Neural Networks

Neural networks are physical cellular systems which can acquire, store and process the experiential knowledge [27]. They mimic the human brains in order to carry out learning tasks. Some of the applications of NN include robotics, securities, medical science, aerospace, recognition and defense. Neural networks are also being used to carry out the task of software testing. Neural networks learning techniques are used in automatic generation of test cases for black box testing. In the work by Mariani et. al. [28], a tool AutoBlackTest was presented in order to generate test cases for GUI application. AutoBlackTest used reinforcement learning techniques, Q-learning agent and test case selector to generate test cases for interactive application. The work also compared the proposed technique to GUITAR and resulted that AutoBlackTest generated test cases detects more failures than GUITAR.

In the work by Lilan, wu et. al.[29], back propogation neural networks based technique was used to generate test cases covering the functionality of the software under test. The proposed methodology was implemented on an automatic teller machine (ATM).

In one of the works by Saraph et. al., [30], neural networks were used as classifiers to form equivalence classes of input domain. The proposed technique was used to reduce the number of test cases of black box test suite. The technique was based on three phase algorithm on network pruning.

Data mining algorithms were used by Last et. al. [31], to induce functional requirements for test data. The induced functional requirements were then used to minimize the set of test cases. Info fuzzy networks were used in order to accomplish the task of data mining. The technique was validated using partial differential equations.

In the work by Vanmali et. al. [32], artificial neural networks were presented as automated oracle to determine the result of test case. Back propagation algorithm was used to train the neural network. Training of neural networks was done using black box testing.

In black box testing, number of test cases generated for a system may be colossal. Execution of all these test cases is not possible due to resource and time constraints. Bhasin et. al. [33] presented a framework which prioritizes the black box test cases with help of neural networks. The work also presented the guidelines to prioritize test cases in black box test suite. The proposed technique was implemented on financial management system.

In another work, neural networks were trained to find out the priority of black box test cases [34]. In training phase, neural networks were fed with test cases which are manually prioritized using manual test case prioritizer, design specifications and SRS documents. Trained neural networks were then used to prioritize other test cases on the scale of 1 to 10; 1 being the highest priority. The proposed work was implemented on enterprise resource planning system.

In the work by Bhasin et. al. [35], backpropogation neural network model was proposed to prioritize the test cases. The work was implemented on 200 test cases. 2, 5, 10, 15 and 20 layers neural networks were used to carry out the experiment. The result concluded the efficiency of neural networks as test case prioritizers.

C. Fuzzy Logic

Marcos et. al. [36] proposed a hybrid intelligent approach based on neuro-fuzzy classifiers and multi-agent system architecture in order to improve the quality of black box testing. The work compared the efficiency of traditional black box testing techniques with the proposed hybrid intelligent approach.

In the work by Sujata et. al. [37], fuzzy logic was implemented in order to generate test cases. The work proposed search based testing approach using fully logic and natural language processing.

In another work, fuzzy theory was used in black box testing phase to identify acceptability of test cases by their degree of existence [38]. The work implemented fuzzy theory on graph based testing, boundary value analysis, equivalence class partitioning and orthogonal array testing.

V. CONCLUSIONS

Black box testing is essential at times, when software code is not available during testing phase. Crafting good test cases for black box testing is one of the most important tasks to create quality software. The work reviews various techniques for creating and prioritizing black box test cases. The work presents both conventional as well as artificial intelligence techniques used to increase efficiency of black box testing. The review has been carried out in accordance with the guidelines proposed by kitchenham [5]. The work also presents the role of artificial intelligence techniques (Genetic Algorithms, Neural Networks and Fuzzy Logic) in black box testing.

REFERENCES

[1] J. Myers, "The Art of Software Testing, second edition", John Wiley and Sons, 2004.
[2] IEEE Standard Computer Dictionary: A Compilation of IEEE Standard Computer Glossaries; IEEE; New York, NY.; Software testing, 1990.
[3] A. Bertolino, "A brief essay on Software Testing", in Thayer, R.H., Christensen, M.J (eds.) Software Engineering, 3rd edition. Vol 1, pp. 393-441, Wiley-IEEE, 2005.
[4] IEEE Standard Computer Dictionary: A Compilation of IEEE Standard Computer Glossaries; IEEE; New York, NY.; Black, Rex; (2002), Managing the Testing Process (2nd ed.), Wiley Publishing, 1990.
[5] B. A. Kitchenham, . et. al. "Systematic literature reviews in software engineering .A tertiary study", Information & Software Technology .INFSOF , vol. 52, no. 8, pp. 792-805, 2010
[6] IEEE 1990 definition of black box testing.[online]
[7] IEEE Standard 610 (1990) definition of test cases [online].
[8] IEEE std 829-1983 definition of test cases.
[9] N. Chauhan, "Software Testing: Principles and Practices", oxford publications, 2010.
[10] P. C. Jorgensen, "Software Testing A Craftsman's Approach", CRC Press, 1995.
[11] C. Kaner, J. Falk and H. Q. Nguyen "Testing Computer Software", 2nd Edition, Wiley, 1999.
[12] B. Beizer, " Software Testing Techniques", 2nd Edition, Van Nostrand Reinhold, 1990.
[13] R. Blanco, J. Tuya and B. Adenso-Diaz, "Automated test data generation using scatter search approach", Information and Software Technology, 51,4, pp. 708-720, 2009.
[14] S. T. Frezza, S. P. Levitan and P. K. Chrysanthis,"Linking requirements and design data for automated functional evaluation", Elsevier, 1996.
[15] Y.T. Hu and N.W. Lin, "Automatic black-box method-level test case generation based on constraint logic programming", IEEE, Computer Symposium (ICS), 2010 International, pp: 977 – 982 , 2010.
[16] T. Murnane and K. Reed, "On the effectiveness of mutation analysis as a black box testing technique", conference on Software engineering, IEEE, 2004.
[17] M.A. Khan and Mohd. Sadiq, "Analysis of black box software testing techniques: A case study", Current Trends in Information Technology (CTIT), IEEE, 2011.
[18] T.Y. Chen, F.C. Kuo, R.G. Merkel and S.P. Ng, "Mirror adaptive random testing", Elsevier, 2004.
[19] E.Y.K. Chan and Y. T. Yu, "Evaluating several path-based partial dynamic analysis methods for selecting black-box generated test cases", IEEE, 2004.
[20] H. V. Kantamneni, S. R. Pillai and Y. K. Malaiya,. "Structurally Guided Black Box Testing", online-http://www.cs.colostate.edu/~malaiya/structbbox2.pdf
[21] D. Verma and Karambir, "Component Testing Using Finite Automata", Indian Journal of Computer Science and Engineering (IJCSE), 2012.
[22] T. Noguchi , H. Washizaki, Y. Fukazawa, A. Sato and K. Ota, " History-Based Test Case Prioritization for Black Box Testing Using Ant Colony Optimization" Software Testing, Verification and Validation (ICST), IEEE, ISSN-2159-4848, 2015.
[23] H. Bhasin, E. Khanna and Sudha, "Black Box testing based on Requirement Analysis and Design Specification" International Journal of Computer Application, volume 87, No. 18, 2014.
[24] S. N. Shivanandam and Deepa, "Principles of Soft Computing", Second edition, Wiley India, 2012.
[25] M. Last, S. Eyal and A. Kandel, "Effective Black-Box Testing with Genetic Algorithms", Hardware and Software, Verification and Testing, Vol 3875, Springer, 2006.
[26] M. Fisher and R. Tonjes, "Generating test data for black-box testing using genetic algorithms" Conference on Emerging Technologies & Factory Automation, IEEE, 2012.
[27] J.M Zurada, "Introduction to Artificial Neural Systems", Jaico Publishing House.
[28] L. Mariani, M. Pezz, O. Riganelli and M. Santoro , "AutoBlackTest: Automatic Black-Box Testing of Interactive Applications", IEEE, 2012.
[29] L. Wu, B. Liu, Y. Jin and X. Xie, "Using Back-propagation Neural Networks for Functional Software Testing", Anti-counterfeiting, Security and Identification, ASID 2008, 2nd International Conference on 20-23 Aug. 2008.
[30] P. Saraph, M. Last and A. Kandel, "Test Set Generation and Reduction with Artificial Neural Networks" in M. Last, A. Kandel, and H. Bunke (Editors),.Artificial Intelligence Methods in Software Testing, World Scientific, 2004.
[31] M. Last, M. Friedman and A. Kandel, "The Data Mining Approach to Automated Software Testing" Conference on Knowledge Discovery in Data, 2003.
[32] M. Vanmali, M. Last and A. Kandel, " Using a Neural Network in the Software Testing Process", International Journal of Intelligent Systems, 2002, pp. 45-62.
[33] H. Bhasin and E. Khanna," Neural Network based Black Box Testing", ACM Sigsoft Software Engineering Notes, volume 40, 2014.
[34] H. Bhasin, E. Khanna and Sudha, "On the Applicability of Neural Networks in Black Box testing" International Journal of Computer Application, R S Publications, issue 4, volume 2, 2014.
[35] H. Bhasin, E. Khanna and K. Sharma, "Neural Networks based Automated Priority Assigner", Series Advances in Intelligent Systems and Computing, Springer, Volume 381, 2015.
[36] M.A.B. Junior, F.B. de Lima Neto and J. C. S. Fort," Improving black box testing by using neuro-fuzzy classifiers and multi- agent systems", International conference on Hybrid Intelligent Systems, IEEE, October, 2010.
[37] V. Sujatha, K. Sriraman, K. Ganapathi Babu and B.V.R.R. Nagrajuna, "Testing and Test case generation by using Fuzzy Logic and N. L. P Techniques", International Journal of Computer Engineering and Technology, volume 4, Issue 3, 2013.
[38] V. Chandra, "Fuzzy Theory in Black Box Testing", International Journal of Advanced Research in Computer Science and Technology, Volume 2, Issue 2, 2014.