On The Application Of Artificial Life Based Test Data Generation

Harsh Bhasin
Delhi technological University
i_harsh_bhasin@yahoo.com

Shewani
M.Tech Scholar
A.I.T.M, Palwal

Deepika Goyal
Assistant Professor
A.I.T.M, Palwal

Abstract— **Manual test data generation is not feasible in case of large Software. Moreover, the approach does not work for Component Based Modules. Automated Test Data Generation is the process of automatically generating the test cases on the basis of particular criteria. Automated Test Data Generators are intensely important owing to the fact that a good Test Case Generator would lead to better testing. The proposed work uses Artificial Life as the base of Test Data Generator. The technique produces test cases that can be used for Black Box Testing. It is a continuation of our earlier attempt to use Langton's loop to accomplish the task. The technique has been explained and exemplified in the work and is currently being applied on a larger enterprise resource planning system. The work uses module state diagrams to craft the test cases. The technique opens the door of Artificial Life to Test Case generation.**

Keywords: Automatic Test Case Generators, Artificial Life, Langton's Loop, Black Box Testing.

I. INTRODUCTION

The growth of software engineering can justifiably be attributed to the advancement in Software Testing. The quality of the test cases to be used in Software Testing determines the quality of software testing. However, generating test cases is an intense, complex and time consuming task. In order to save both time and resources, automated test data generators are used. During the Literature review it was found that the field has been explored by many researchers. However, the techniques proposed to accomplish the task are inefficient and inapt as far as coverage, execution time, and fault detecting capabilities are concerned. It may be noted that since, the importance of automated test data generators is growing day by day. Therefore, there is a need to develop a test data generator which is good in terms of the above parameter.

Test Data Generators have been developed using many mathematical and nature inspired techniques. The present work explores our earlier attempt of applying a Artificial Life (AL) based technique in order to generate test cases. It may be noted that AL has never been used in the Test Data generation. The work exemplifies the use of technique. The initial results obtained are encouraging and paves way for the use of technique in professional softwares. Currently an Enterprise Resource Planning (ERP) system is being verified. A set of 20 programs has also been chosen for the future verification. Rest of the paper has been organized as follows. The work is a part of our endeavor to examine testing techniques and apply the system being developed on the techniques [14-19].

II. LITERATURE REVIEW

An extensive Literature Review has been carried out in order to understand the work that has been done so far and to find the gaps in the existing work. The papers have been selected by a group of lecturers and Assistant Professors, keeping in mind the importance and applicability of the papers.

During the review it was found that, the period before 2006 saw publishing of very less number of articles in this domain. It has been observed by Shahid[] that owing to the fact that it is a specialized area, therefore very less number of people showed interest in that. It is also possible that the period mentioned did not require Automated Test Data Generation. In order to generate Automated Test, it was also essential to understand the crafting of Test Cases which was being done in that period. It was observed by many researchers that only 5% of the articles in the journals were related to Automated Test Data Generation. It may also be noted that papers were published each year after 1997 till 2013 which proves that the topic is continuously being explored. It was also noted that, both White Box Testing and Black Box Testing technique were used in Automated Test Data Generation.

The three main approaches of Automated Test Data Generation have got almost equal consideration. These approaches are Random, Path oriented and Goal oriented approaches. Many researchers have also tried numerical approaches for Automated Test Data Generator. These included simplex, branch and bound etc.

The stress on heuristic search method, an evolutionary technique, for example Genetic Algorithm and Simulated Annealing was not that much but some of the researchers did tried to explore the possibility of using such methods in test data generation.

The focus on the feasibility of the path selected by the technique has also been examined in various papers. The review has been summarized in Table 1.

Sr. No.	Author	Technique Used	Ref. No.
1	C. V. Ramamoorthy, Siu-Bun F. Ho, W. T. Chen	Symbolic Execution	1
2	Webb Miller, David L. Spooner	Numerical Maximization Method	2
3	Lori A. Clarke	Symbolic Execution	3
4	Praveen Ranjan Srivastava, Tai-hoon Kim	Variable length Genetic Algorithms, Critical Path Coverage	4
5	James Miller, Marek Reformat, Howard Zhang	Program Dependence Analysis Techniques and Genetic Algorithms.	5
6	Harmen and Hinrich Sthamer	Genetic Algorithms	6
7	Roy P. Pargas, Mary Jean Harrold, Robert R. Peck	Genetic Algorithms, Statement Coverage, Branch Coverage.	7
8	Joachim Wegener, Andre Baresel, Harmen Sthamer	Evolutionary Testing.	8

9	Moheb R. Girgis	Genetic Algorithms, Def-Use Association Coverage.	9
10	Dunwei Gong, Wanqui Zang, Xiangjuan Yao	Genetic Algorihtms, Many Paths Coverage based on Grouping	10
11	Anastasis A. Sofokleous, Andreas S. Andreou	Dynamic Test Data Generation, Genetic Algorithms,	11
12	Jin-Cherng Lin, Pu-Lin Yeh	Genetic Algorithms	12

Table 1: Test data Generation Techniques

III. BACKGROUND

In the previous task, Artificial Life method was used to produce test cases [13]. The technique uses modules as the basic unit. The technique is novel as it serves two purposes i.e. generation and minimization of the test cases. In the review it was found that, some of the works depict time as a major factor compared to the coverage of test cases but in the work coverage was given more importance. In the technique the stress was on the function coverage as testing is done to instill confidence in the software, therefore maximization of function coverage factor will lead to lesser time in the maintenance phase.

The technique generates the test cases for all the modules of the program. The technique presented a new state machine called the Function Flow Graph (FFG). The FFG will cover all the modules and help in generating the test suit.

In the technique, for all the modules, the calling sequence is identified, and it starts from the entry point. The input and the output values are identified for each node. Test data is generated for each module and they are fed to the Langton’s Loop.

IV. Verification of the technique

So as to exemplify the technique a tool developed in C# has been taken. The purpose of the tool is to generate Genetic population and perform crossover and mutation. The various modules in the tools are given in Table 2.

S.NO.	**NAME**	**INPUT**	**OUTPUT**	**TASK**
1	generateButton_Click	Number of chromosomes and number of cells in a chromosome.	A 2D array which consists of 0’s and 1’s.	Produces a 2D array.
2	getValuesButton_Click	Population, 2D array.	A 1D array which consists of values corresponding to those chromosomes.	Produces a 1D array.
3	applyButton_Click	The 1D and 2D array generated above and also the threshold value.	Those chromosomes which have threshold greater than requisite value.	Produces chromosomes.
4	Button10_Click	Final population and mutation rate.	A mutated population which is a 2D array.	Produces an array containing mutated population.

5	Button_Click	A 2D array which depicts the population and crossover rate.	A 2D array which consists of original and crossover population.	Produces an array and crossover population.
6	ShowButton_Click	A 2D array which consists of original and crossover population.	A 2D array shown in text box.	Produces a 2D array.

Table 2: Modules in Genetic Tool, which is to be tested using the technique

Figure 1 shows the module state diagram of the tool.

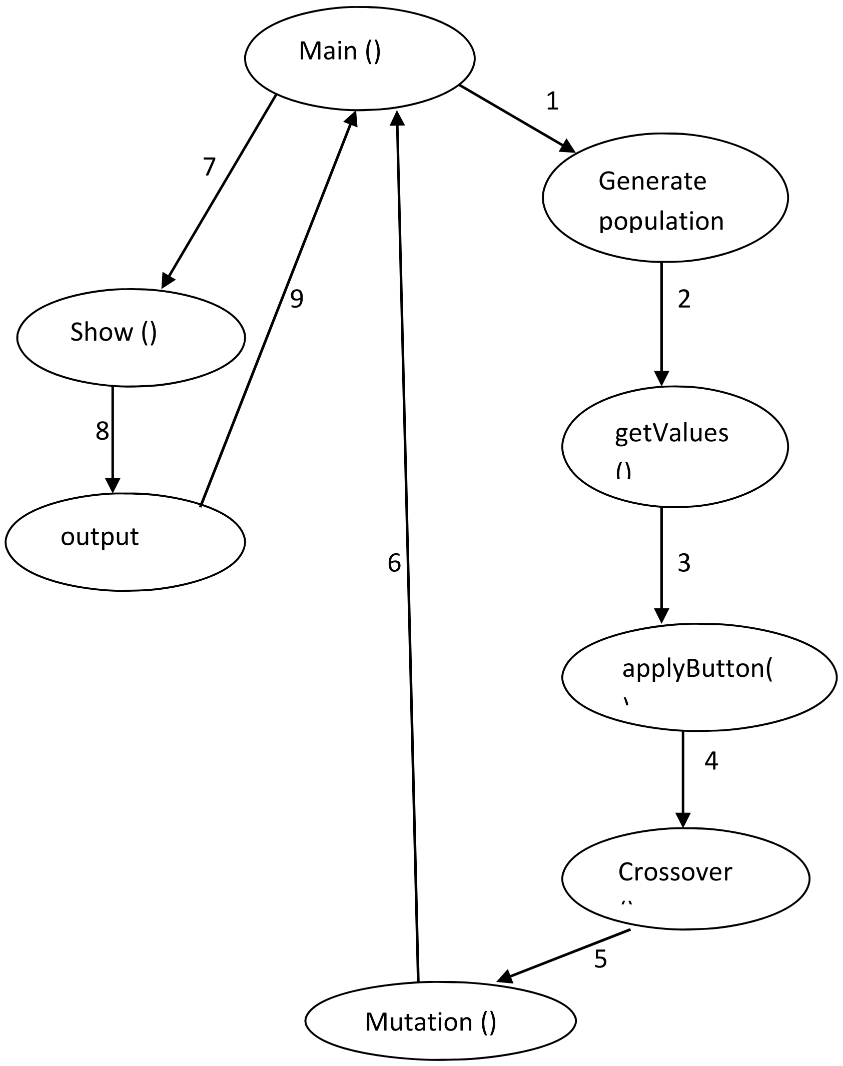

Figure 1: Module state diagram of the tool.

Langton's loops are a type of artificial life in a cellular automaton. They consist of a loop of cells having genetic information. The genes tutor it to make three left turns, finishing the loop, which then detaches from its parent. The intersection point of Lagtons loop generates 0 or 1, n number of times thus crafting the test cases. For example in the above case let the population Size be 30 and the number of cells in chromosomes be 25, then 750 values are needed to craft the test population. Figure 2 depicts Langton’s loop.

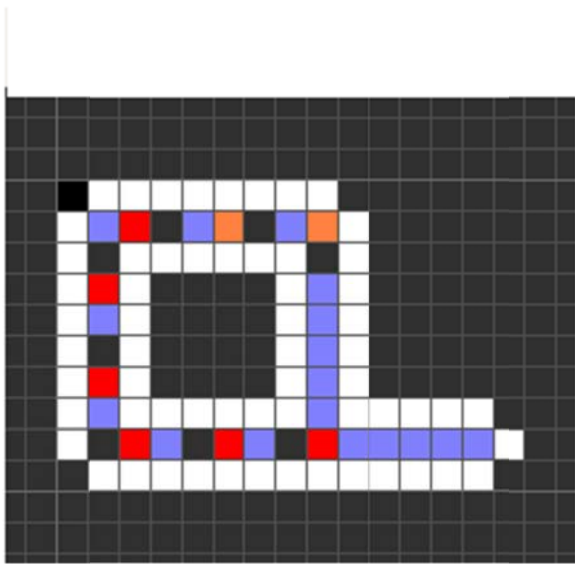

Figure 2: Langton’s loop

V. CONCLUSION

The above work is a continuation of our effort to develop a framework of Automatic Test Case generation, which would be used both for Black Box and White Box Testing [13, 20]. The technique that can be used for Black Box Testing has been explained and exemplified in the work. The technique that would be used for White Box Testing has been designed and is been verified in parallel using cellular automata [20]. The above work successfully amalgamates the two disciplines. It also paves the way of application of Artificial Life to Software Testing.

VI. REFERENCES

1. C. V. Ramamoorthy, S. B. F. Ho, W. T. Chen, On the Automated Generation of Program Test Data, Trans. Softw. Eng., Vol. 2, pp. 293-300, 1976.
2. Webb Miller, David L. Spooner, Automatic Generation of Floating-Point Test Data, IEEE Trans. Software Eng 1976, Vol. 2,. pp.223-226, 1976
3. Lori A. Clarke, A System to Generate Test Data and Symbolically Execute Programs. IEEE Transactions on Software Engineering, Vol. 2, 1973.
4. Praveen Ranjan Srivastava, Tai-hoon Kim, Application of Genetic Algorithm in Software Testing, International Journal of Software Engineering and Its Applications, Vol. 3, No.4, October 2009
5. James Miller, Marek Reformat, Howard Zhang, Automatic test data generation using genetic algorithm and program dependence graphs, Information & Software Technology 01/2006; 48:586-605. pp.586-605.
6. Harmen and Hinrich Sthamer, The Automatic Generation of Software Test Data Using Genetic Algorithms, University of Glamorgan, 1995
7. Roy P. Pargas et. Al., Test Data Generation using Genetic Algorithms, Journal of Software Testing, Verification and Reliability. 1999.
8. Joachim Wegener et. Al., Evolutionary Test Environment for automatic Structural Testing, Information and Software Technology
9. Moheb R. Girgis, Automatic Test Data Generation for Data Flow Testing Using a Genetic Algorithm, Journal of Universal Computer Science, Vol. 11, No. 6, 2005.
10. Dunwei Gonga et. Al., Evolutionary generation of test data for many paths coverage based on grouping, The Journal of Systems and Software, Vol. 84, pp. 2222– 2233, 2011.
11. Anastasis A. Sofokleous, Andreas S. Andreou. Automatic, evolutionary test data generation for dynamic software testing. The Journal of Systems and Software, Vol. 81, pp. 1883–1898, 2008
12. J. C. Lin, P. L. Yeh, Automatic test data generation for path testing using GAs, Information Sciences, 131 (1-4), pp. 47-64, 2001.
13. Harsh Bhasin, Shewani and Deepika Goyal, Test Data Generation using Artificial Life, International Journal of Computer Applications, Vol. 67, No. 12, pp. 34-39, 2013.
14. Harsh Bhasin et. al., A Novel Approach to Cost Cognizant Regression Testing, International Journal of Computer Science and Management Research, Vol. 2, Issue 5, pp. 2595–2600, 2013.
15. Harsh Bhasin et.al, Software Architecture Based Regression Testing, International Journal on Computer Science and Engineering, Vol. 5, No. 04, pp. 226-229, 2013.

16. Harsh Bhasin et.al, Software Architecture Based Regression Testing Implementation, International Journal of Computer Science and Engineering, ISSN (Print):2278-9960, Vol. 2, issue 3, pp. 1-4, 2013.
17. Harsh Bhasin et. al., Regression testing using Fuzzy Logic, International journal of Computer Science and information technology, Vol. 4, issue 2, 2013.
18. Harsh Bhasin et. al., Implemenattion of Regression testing Using Fuzzy Logic, International Journal of Application or Innovation in Engineering and Management, Vol. 2, Issue 4, 2013.
19. Harsh Bhasin et. al., Orthogonal Testing Using genetic Algorithms, International Journal of Computer Science and Information Technology, Vol. 4, Issue 2, 2013,.
20. Harsh Bhasin and Neha singla, Cellular Automata Based Test Data Generation, ACM SIGSOFT Software Engineering Notes, Accepted.