DE GRUYTER OPEN

Management Systems in Production Engineering

2016, No 2 (22), pp 106-109

Date of submission of the article to the Editor: 02/2016
Date of acceptance of the article by the Editor: 04/2016

DOI 10.12914/MSPE-06-02-2016

BENEFITS OF AN APPLICATION OF EVOLUTIONARY STRATEGY IN THE PROCESS OF TEST DATA GENERATION

Marek ŻUKOWICZ
Rzeszow University of Technology

Abstract:
The aim of the article is to highlight the advantages that can be obtained through the use of evolutionary strategy in software testing, specifically in the process of test data generation. The first chapter introduces the reader to the topic of the article. Presents information of the problem of software quality, test data fitness and quality criteria. The second chapter provides an overview of the publication in which is described the test data generation problem by using evolutionary strategies. In this chapter there are presented, different approaches to address the optimization problem of test data selection. The third chapter sets out the advantages which in the opinion of the author result from the application of evolutionary strategy in the process of test data generation. In this section have been drawn conclusions from the article, from books listed in the bibliography. The author of the article presents advantages of evolutionary strategy too as a person, which tests a software in practise. The last chapter in addition to summaries and conclusions, proposes the author to suggest in which issues related to testing could be used evolutionary strategies.

***Key words**: evolutionary strategies, test data, test data quality criteria, optimization test data*

INTRODUCTION

The test data are the values that is inserted into the tested system as an input for a certain functionality, in order to verify or validate its correctness. The input is inextricably linked with the test case [18]. These are the information, which values should be substitute for the function implement the requirements in the system. A test case is a single operation of checking certain properties of the software or device.

In the literature there it is a breakdown of the test data to the data required and not required. The authors of a position [18] say that the test data required are those which a declaration is necessary in the process of use of the application. In practice it is easy to see that computer applications differently respond to incorrectly entered data or simply contain different types of errors witch prevent the proper functioning of the application right after entering data into the form, regardless of whether these data are required or not. For each application there are such data sets, which likely will be introduced during use the application or test data, that users probably do not enter through the entire life cycle of the software [17]. Regardless of what the test data are entered as input, the application must react positively of them and that data will be stored, processed, transferred or the application should return a message about incorrect data, which in addition will inform the user which field contains illegal data or what function received improper input. There are a lot techniques that allow testers to make the choice of data so that they are consistent with the actual data which will be put into application by potential users. Under the heading [19] test data acquisition techniques are given as:

- testing the function areas,
- distribution on equivalence class,
- boundary value analysis (migth be analized input and output boundary values),
- code coverage testing
- the application of the decision-making tables.

Other methods of obtaining test data are:

- applying the pair-wise testing method [22],
- applying the method of n-wise testing,
- the use of general or own mathematical models in tested systems [21, 23, 24].

In the software testing it is exists such concept as a test strategy, that is a certain sequence of the activities to obtain information about the test cases that is also on the test data. In the literature are favored formal test strategies (consistent with the international rules of qualities) and the informal strategies the test (whitening of data on the basis of the experience of testers, the guesswork of connected mistakes with introduced the data). Test strategies can be mixed with the technicians acquisitions of input data. According to Farley , Humble and Roman [6, 20] test data they ought in particular to be marked off such features how:

- performance (can be understood as a time of the test),
- dependence on the system (quality depends on technological aspects),
- completeness (the degree to which these data have values for all attributes in the system),
- timeliness,
- reliability,

- consistency (eg. The consistency between the test data)
- accuracy (for tests on numbers, in applications that require high accuracy, eg. convert measurements or units).

The correct test data can also be obtained from the Internet or through the using data generators, which are free applications.

In recent years with the development of evolutionary strategies there appear ideas to their using in the test data generation process. The origins of these works date back to the second half of the 90s, eg. the position [10]. Evolutionary strategies are divided into evolutionary programming, genetic algorithms and evolutionary algorithms. Differences between genetic algorithms and evolutionary algorithms are not large. Very often these terms are used differently in the literature any arrangement of the algorithm. Initially, genetic algorithms were based on binary strings, and evolutionary algorithms for writing floating-point [19]. The purpose of writing this article is to review the application of evolutionary strategies in the process of test data generation and an indication of the values to be gained from application evolutionary strategies in the software testing process. The author assumes that the reader knows the basic concepts such as graph algorithm, optimization, extreme features, evolutionary strategy, and knows the basics of computer science.

REVIEW OF SELECTED EVOLUTIONARY STRATEGIES IN THE TEST DATA GENERATION PROCESS

As already mentioned in the previous section, the beginnings of attempts to apply evolutionary strategy date back to the second half of the 1990s. Position [10] presents strategy in which by a code of a program are generated control flow graph (CFG) and gn the control dependence graph (CDG). The genetic algorithm works in such a way that it generates a set of test cases and verifies that the generated cases satisfy stop conditions. If the conditions are not satisfy, then the algorithm draws a new set of individuals, from which will be generate new individuals (test data). The constraint for the algorithm is also time. Quality measurement is done by covering the paths in the CDG graph by generated test data. The strategy was compared to an algorithm that generates random test data. It turned out that the genetic algorithm in a fraction of the time can solve the problem of covering CDG graph paths than a random algorithm. An analogous approach is presented in [14].

Authors of positions [1, 5, 7, 13] present the evolutionary strategies that generate the test data through the use of graphs. Author of position [1] proposes the using of a genetic algorithm that generates sequence of test paths based on data flow graph. In the article it has been introduced definitions the independent path, the base path of and the basic set of paths. Independent path is one a path in which each edge is not present in other paths in the data flow graph. The basic set of paths is one which satisfies the three conditions: each track is independent, all edges in the graph should be covered by basis seth of paths, each path not included in the basic set of paths can be composed of linear combination of paths from the basis set. An interesting feature of the algorithm is a variable length of generated elements (chromosomes) that arise as a result of the application of crossover operator. Under the heading [5] is described an algorithm that works this way, that his goal is not to find a single solution, while several solutions in the same time. These solutions is a collection of nodes in the data flow graph that are defined as objectives that should be achieved by the algorithm. The provided input population (test data) is divided into subsets, of which one the best in terms of fitness element is chosen. The set of the best elements of this population is a set for which the algorithm checks the availability of targets. Test data (generated items) that can be used to get to the target are added to a collection of test cases. As a result of successive iterations of the algorithm is adding to the collection of test cases new test data such that which has not been generated in the previous iteration. The authors of position [7] to describe a strategy of test data generation similar to the strategy described in [5]. The differences appear in the application of the fitness function and in the selecting individuals from subsets, which are formed from the initial input data. Evolutionary algorithm described in [7] works in parallel by searching each subset and this approach increases the speed of finding a solution. The position [13] presents a strategy in which the UML diagram is joined with SD diagram (Sequence diagram) that is a sequence of instructions executed to allow access to the target (in this case the node in the graph). The problem concerns the indication of such path (actions which shall be carried out during the tests), which should be tested in the first. The article presents three algorithms: the first proposes the input data generation from a diagram named, the second proposes assigning weights to nodes, the last turns activity diagram into state diagram in order to allow the process of prioritization of test paths (test data). The fitness function in the genetic algorithm uses the weight assigned to the paths in the graph in the process of evolution. The path is converted into binary sequence from which the evolutionary algorithm calculates the fitness function, apply on this sequence the crossing and mutation. If the algorithm finds a sequence that reaches the best match value for input, then this string is indicated as the most optimal. In this case, the optimization is to minimize the square of the calculated cost of the path. Under the heading [2] author present a strategy that combines K-means clustering algorithm with the genetic algorithm. Provided inputs are converted to numbers in order to allow for the using K-means grouping methods. After dividing data into subsets by the K-means algorithm, a genetic algorithm starts a work. Each class has a center of gravity. The task of the genetic algorithm is to generate such a subset of input data, that doesn't have two points about similar coordinates. The authors of position [3] presented a proposal of the algorithms based on evolutionary strategies in which the optimization process of test data generation starts since providing some of the initial population to algorithm (test data), followed by the measuring of statements coverage in the code by the fitness function. Based on the value of the fitness function an algorithm selects elements of the population that are involved in reproduction. The authors compared the using of genetic algorithm, an evolutionary algorithm and second evolutionary algorithm implemented in parallel. The main difference between these algorithms in the article is a mutation process and crosover. Item [4] in bibliography is so interesting, that is a combination of the algorithm used in Kalman filter known from automation with evolutionary algorithm, which solves the problem of optimizing the test data generation. The short name of described in the article algorithm is KFGA. The authors accept the assumption that ge-

nerated input data should cover as as much as possible lines of code by calling the appropriate instructions in the tested software, but with such a limitation, so the algorithm generated the least test cases (input data). KFGA algorithm is an adaptive algorithm, so the results of this algorithm are generally acceptable, however, are not always the best, as can be seen in tables, in which the authors compared the other previously known evolutionary strategies of the KFGA algorithm. Articles listed in [8] and [9] show very similar strategies that rely on finding errors in the tested code. They differ mainly in that the evolutionary algorithm in position [9] try to find for several paths at the same time. Errors are divided into categories, finding errors is evaluated for these categories. In position [11] the authors present the interesting approach of applying evolutionary algorithm, in which the stop conditions are dynamic, in this case are dependent on the quality of the generated test data. The algorithm decides when to end the process of test data generation. Here is a strategy of fall risk failure model. A position [12] is mostly a review of SBST (Self Based Testing Model) methods that generate the test data according to certain three different strategies based on genetic algorithms. The article was created as a result of the grant. The contents are mainly conected with functional testing related to code coverage tests. The authors of position [15] show how it can be used evolutionary algorithm to test data prioritizing (test cases prioritizing), taking into account the time to run the test and number of defects that can find a test case. Author of position [16] presents the evolutionary algorithm, which uses a strategy called Harmony Search optimization process for test data generation. An evolutionary algorithm starts from the harmonic memory initialization proces. It is a step in which there are generated test data vectors randomly (vectors of n-dimensional). New data vector is generated by three rules: pulling the data from the harmonic memory, randomly and in a controlled strategy by the creator of an algotithm. The remaining steps of evolutionary strategy described in the article are analogous to the traditional evolutionary algorithms. It turns out that such a strategy, in some cases of a structural testing gives a quick and good solution.

ADVANTAGES OF TEST DATA GENERATION USING EVOLUTIONARY STRATEGIES

The analysis of the two previous chapters lets the reader draw some conclusions which result from the application of evolutionary strategy in the test data generation process:

1. Randomness but one whose direction is controlled by using the adjustment function and evolutionary processes (crossover and mutation). Test data generators offen generate input data, but in a totally uncontrolled manner. Most of the items contained in the publication references, it appears that the application of evolutionary operators and matching allows you to generate test data in such a way that they are in accordance with the intended use, which are also reliable, depending on the tested system.
2. Introduction of a mutation, which can be seen as a factor which eliminate the phenomenon of repetition of some of the schema when you generate test data, and from mathematical point of view the mutation can protect from being stuck under the local maxima and minima.
3. Evolutionary algorithm method is quite versatile, so it can be used to generate test data based on different models of functionality in informatic systems. For each mathematical model or use case diagram it can be used the evolutionary algorithm in the test data generation proces.
4. Method is relatively fast: finding a solution is often possible after a few iterations or searching a small part of the states as it can be seen in the publications contained in the bibliography [2, 3, 4].
5. Resistance to the paradox of the pesticides. The paradox of the pesticides in testing says that the same testing repeated without any change are becoming less effective in finding faults in the tested program [20]. By using the application of evolutionary strategies to generate input data, the risk of this phenomenon greatly decreases or does not occur.
6. The possibility of defining the fitness function in such a way that generated the test data which are highly associated with the context of the using the tested software. Testing is context-sensitive, so the test data must also be dependent on the context of using the application.
7. Multi-criteria optimization. Fitness function, which sets the direction of the new generation (test data), may be the sum of several functions that measure the quality and can be multiplied by weight. This approach gives the ability to generate tests that are correct for several data-related attributes.
8. Possibility of adaptation, i.e. dynamic stop criteria of an algorithm. If the algorithm finds no satisfactory solution to the problem raised in relation to the specific function and purpose, it is possible to change the parameters that control the process of generating a new generation (in this case, the test data), which may improve the quality of the solution to raised the problem [20].
9. The possibility of combining with other algorithms (K-means, the Kalman Filter), which increases the quality of the optimization problem solution [4, 7].

SUMMARY

The main purpose of writing the article was to gather information about the commonly known techniques for test data generation, to present ways to test data generation process using evolutionary strategies (little known by the person who test in practice) and to present the benefits of this approach benefits. During a deeper analysis of the article, the reader will realize that testing and the test data adjusting is not a trivial activity. Publications generally contain a simple application examples and a small reference to the practice because they are not described research on the systems used, eg. by private companies or public institutions. Described in the second chapter publications, however, present strategies that solve a very important problem in testing. Analyzing the articles included in the bibliography the reader colud be notice that evolutionary strategies coping well with the optimization problem of the test data generation. It is therefore to experiment with such strategies into practice, conducting tests used or developing informatic systems. In the future, there will be the part of the author's proposals for the application of evolutionary strategies in testing systems, which are used by enterprises and evolve continuously, adapting to the changing environment.

"The bar quality" of informatic systems is increasing over the years. Evolutionary algorithm by applying the crossing, mutation and matching functions as may generate test data that were characterized by such features quality, which are very important in testing.

LITERATURE

[1] S.G. Ahmed. *Automatic generation of basis test paths using variable length genetic algorithm,* Journal Information Processing Letters, Volume 114 Issue 6, June, 2014, pp. 304-316.
[2] R. Alavi, S. Lofti. *The New Approach for Software Testing Using a Genetic Algorithm Based on Clustering Initial Test Instances*, International Conference on Computer and Software Modeling 2011, IPCSIT vol.14 (2011).
[3] E. Alba, F. Chicano. *Observations in using parallel and sequential evolutionary algorithms for automatic software testing*, Computers & Operations Research, Vol. 35 Issue 10, October 2008, pp. 3163-3183.
[4] A. Aleti, L. Grunske. *Test data generation with a Kalman filter-based adaptive genetic algorithm,* The Journal of Systems and Software Vol. 103, May 2015 pp. 343-352.
[5] M. Alshraideh, B.A. Mahafzah, S. Al-Sharaeh. *A multiple-population genetic algorithm for branch coverage test data generation*, Software Quality Journal, Vol. 19, Volume 19, Issue 3 September 2011, pp. 489-513.
[6] D. Farley, J. Humble. *Ciągłe dostarczanie oprogramowania*, Helion 2015.
[7] D. Gong, T. Tian, X. Yao. *Grouping target paths for evolutionary generation of test data in paralel*, The Journal of Systems and Software, Vol. 85, Issue 11, November 2012, pp. 2531–2540.
[8] D. Gong, Y. Zhang. *Generating test data for both path coverage and fault detection using genetic algorithms*, Frontiers of Computer Science, December 2013, Vol. 7, Issue 6, pp. 822-837.
[9] D. Gong, Y. Zhang. *Generating test data for both path coverage and fault detection using genetic algorithms: multi-path case*, Frontiers of Computer Science, October 2014, Vol. 8, Issue 5, pp. 726-740.
[10] M.J. Harrold, R. Pargas, R. R. Peck. *Test-Data generation using genetics algorithms,* Journal of Software Testing, Verification and Reliability 1999.
[11] I. Hermadi, C. Lokan, R. Sarker. *Dynamic stopping criteria for search-based test data generation for path testing* , Information and Software Technology, April 2014 Vol. 56, pp. 395-407.
[12] J. Hudec, E. Gramatova. *An Efficient functional test generation method for processors using genetic algorithms*, Journal of Electrical Engineering, July 2015, Vol. 66, Issue. 4, pp. 186-193.
[13] N. Khurana, R.S. Chillar. *Test Case Generation and Optimization using UML Models and Genetic Algorithm*, Procedia Computer Science, August 2015, Vol. 57, pp. 966-1004.
[14] H. Kim, P.R. Srivastava. *Application of Genetic Algorithm in Software Testing*, International Journal of Software Engineering and Its Applications, October 2009, Vol. 3, Issue 4, pp. 87-96.
[15] R. Krishnamoorthi, A. Sahaaya, S.A. Mary. *Regression Test Suite Prioritization using Genetic Algorithms,* International Journal of Hybrid Information Technology, July 2009 Vol.2, Issue .3, July.
[16] C. Mao. *Harmony search-based test data generation for branch coverage in software structural testing*, Neural Computing and Applications, September 2013, Vol. 25, Issue 1, pp.199-216. Springer-Verlag London 2013.
[17] M. Mirzaaghaei, F. Pastore, M. Pezze. *Automatic test case evolution,* Software testing, Verification and Reliability, April 2014, Vol. 24, Issue 5, pp.386-411.
[18] A. Piaskowy, R. Smilgin. *Dane Testowe, teoria i praktyka*, Helion 2011.
[19] A. Roman, *Testowanie i jakość oprogramowania*, PWN 2015.
[20] D. Rutkowska, M. Piliński, L. Rutkowski. Sieci neuronowe, algorytmy genetyczne i systemy rozmyte, PWN W-wa 1997.
[21] D. Warchoł, M. Żukowicz. *Testing education: test case prioritization using matrices,* General and Professional Education, May 2015, Vol. 1, Issue 8, pp. 57-62.
[22] M. Żukowicz. *Edukacja testowania: Narzędzie All-pairs Testing w procesie optymalizacji testów konfiguracji – zastosowanie narzędzia w systemie B2B OPTIbud,* General and Professional Education, Dezember 2015, Vol. 4, Issue 12, pp. 99-106.
[23] M. Żukowicz. *Edukacyjne i ekonomiczne aspekty zastosowania cyklu Hamiltona w projektowaniu i testowaniu oprogramowania*, General and Professional Education, numer Dezember 2014, Vol. 4, Issue 13, pp. 95-102.
[24] M. Żukowicz, O pewnych problemach analizy wartości brzegowych, Internet:, http://testerzy.pl/materialy/index.php?file=analiza-wartosci-brzegowych.pdf, access date 03.03.2016.

mgr Marek Żukowicz
Rzeszow University of Technology,
Faculty of Electrical Engineering and Computer Science
Department of Automation and Computer Science
ul.Wincentego Pola 2, 35-959 Rzeszów, POLAND
e-mail: bobmarek@o2.pl

Artykuł w polskiej wersji językowej dostępny na stronie internetowej czasopisma.

The article in Polish language version available on the website of the journal