# Oncological Inspired Techniques for Intelligent Software Testing

**Lipsa Sadath[1], Reshmi Nair[2]**

*[1]Faculty, Information Technology, Amity University, Dubai, UAE*
*[1]lsadath@amityuniversity.ae, [2]rnair@amityuniversity.ae*

***Abstract: Biologically inspired computing techniques are highly on demand due to the inter-connection of various specialized fields and their requirement of adaptable methods in solving software engineering problems. Mutation testing has acquired much importance in this scenario. The concept of mutation testing has originated from the bio immune system. This paper is a novel study on mutation testing that proposes a framework OSWM (oncological software mutation) from oncological tests performed in human pancreatic cancer cells. The framework connects the relationship between cancer cell tests and software tests ensuring mutation tests owe its origin from human cells tested. The study argues an intelligent software with an efficient immune system can therefore aim at testing, diagnosing and healing on its own.***

***Keywords: Mutation testing, Oncology, Mutants, Clinical trials, Parameters, Mutant score, Operators.***

## I. INTRODUCTION

The significance of software testing is attributed in achieving and evaluating the excellence of software specifications as per the standard requirements. The essentiality of testing a software is huge so as to ensure you deliver the right product to the market.

Software tests can be conducted in different patterns. Typically this depends on the set of data you choose to test your software with. That is the data that can hold more probability of finding errors or that can have lesser probability of finding errors. Therefore any software product is expected to have undergone rigorous testing before it reaches the market especially if has to be dealing with human safety [1]. Thus the understanding of true software testing can make a profound difference in the success of a project [2].

The concept of unethical and dangerous laboratory experiments being eliminated using executable biological prediction software [3] has gained a lot of importance in the bio-inspired conceptual algorithms.

A system has an ability to be itself dependable in a persistent way even when there are failures in the system [4]. This concept is many a time matched with the fault tolerance system in a software to prove its resilience when matched with the bio-immune system.

Clinical trials are experimental trials performed at the research level in medical institutions. These are involved with human participants with biomedical and behavioral research study strategies designed to obtain specific treatment conditions. These include drug absorption, novel vaccines, dietary supplements, medical devices and its responses. These information generate data on safety, bioavailability and efficacy of the test samples [28, 29].

Pancreatic cancer is one among the top types of the cancers with worst diagnosis according to World Cancer Statistics [30]. At the later stage of diagnosis, it invades further organs across pancreas available in the vicinity such as stomach, duodenum, colon, kidney and spleen is observed [31, 32, 33].

This paper discusses basics, types and importance of software testing for reliability with mutation testing on focus which is oncology test inspired. Concepts picked in the discussions for generating biological test cases are from oncological test point of view.

Section II are software error terminologies, section III explains the myths in software testing, section IV describes the types of test levels, both structural and functional tests, further more section V explains software mutation with mutant order, execution, testing and suite explanation. Section VI describes the related works on mutation tests, section VII is oncological mutation describing oncological mutants and their types in comparison with software mutations at each level, section VIII is the description of the framework OSWM and section IX with future works and scope as test model.

## II. TERMINOLOGIES

*Errors* are mistakes that could be syntactical, or logical. These mistakes in codes are called *bugs.* These bugs may create other problems in the program that may be termed *faults*. Such faulty programs may cause programs to crash or not deliver the functionality which could be termed as a *failure* of the project. Testing approaches with an intension to find the defect while debugging targets to remove the bugs [6] and make the software fault free.

## III. MYTHS IN SOFTWARE TESTING

Software engineers believe themselves to have all confidence in coding. They may not have the feeling of having errors in their software until they are tested by the software testers. It is often believed that if the engineering tools used are perfect,

**978-1-5386-9346-9/19/$31.00 ©2019 IEEE**

any software can go error free. Typically anyone who develops a software finds it difficult to find an error in their own product as long as the functionality of the customer requirements are met well.

## IV. TYPES OF TESTING

*Alpha tests* are performed at the developer's site creating an environment equivalent to that of the original one while a *beta test* occurs at the customers' site which is the original site where the software performance is checked. Acceptance tests are performed to validate all requirements for the customers in a well-planned manner.

### *A. Test Levels*

*Unit testing* calls to check individual modules or classes, methods etc., *integration testing* checks the capability of the software when the modules are integrated with one another while the *system testing* covers the whole system function for a complete error free system including security and robustness of the system, Figure:1[1, 2, 7].

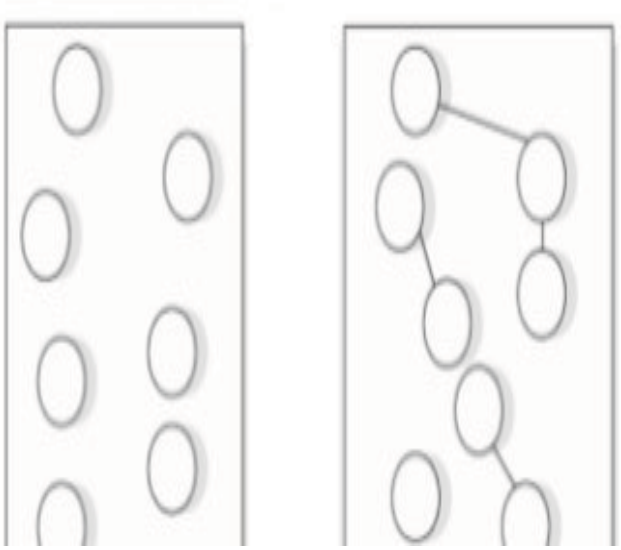

**Fig. 1. Test Levels**

### *B. Structural and Functional Tests*

These tests can be both *structural and functional*. A functional test typically checks the functionality of the software keeping in view the user requirements, whereas a structural test catches code errors that a functional test may skip at times. This is because there are situations where some codes do not disturb the functionality of a software. Structural tests looks at the logic examination generating test cases which is the prime activity of *white box testing*.

The functional and structural tests have different types of testing in themselves. While functional tests may include different test patterns such as *analysis of boundary value test data, test by creating equivalence classes so that the classes catch all similar types of errors, testing by decision tables, cause effect graphing technique, special value testing* etc. to confirm the functionality of a program, the structural tests include *data flow, path testing, cyclomatic complexity tests, graph matrices, mutation tests* etc. [1]. Structural tests are impressive as they are expected to go through the codes of the program in different patterns to check the structure of a software.

## V. SOFTWARE MUTATION

The test is based on fault simulation techniques making copies of the original program. These simulated copies are called mutants. Basically mutants are deviation obtained from the original software program. If mutants are developed and left without testing those using test cases, they are dead codes. Test cases are documentations of testing steps.

Therefore effective test cases are generated to check these mutants. A mutant which is checked by these test cases when detected with errors are then killed [1]. Otherwise they are called equivalent mutants which are similar to the original program with no errors in the mutants created. Therefore the process of mutation aims at finding test cases to effectively kill huge number of mutants. That is, these mutants are injected with the test cases to verify that the test cases catch the errors there and typically check the codes meet the requirements they are intended for.

Mutants are chosen very carefully. When operators are used to generate them, we call mutant operators.

For Example;
For an original expression, p+1 in the program, it's better to use a mutant p+2 than using a p*50 which has no relation.

### *A. Mutant Order*

Whenever a code is mutated there needs to be a way by which the mutation levels are measured, as to how many times the same codes have undergone mutations. Thus a single change made to an expression is considered first order mutant and a mutant from the first order is then considered a second order mutant.

### *B. Mutant Execution*

While executing mutants, it becomes very important that the input values are chosen carefully. Otherwise many mutants may go unexecuted and could show a functional success than a structural success for the time being.

Consider the following example below where the Sum is got only when the value of $l$ is less than $m$ and $m$ is equal to $n$. In such cases mutants are generated only when the two conditions are met at the same time and the loop is entered.

```
Read (l, m, n);
If (l<m) and (m=n) then
{
Sum: =l+ m+ n; {Make mutants, m1, m2, m3….}
}
```

Therefore the killing of these mutants depend largely on their location in a program. This is inspired from the concept of the cancer cells located in human body, detected, killed and treated for the disease.

Rankings may be used while rating the score of a program as the number of killed mutants against the total number created and the equivalent mutants.

{#killed / (#total-#equivalent)}*100…… (1)

### *C. Mutation Testing*

Mutation testing is known to be a fault based testing method providing a preferable test measure with the aid of mutant system [14]. The fundamentals of Mutation testing is achieved by representing the errors made by the programme test suite. The detailed analysis of them can be obtained by choosing the location, parameters in the prevailing system before and after incorporating to the test suite [1]. The test adequacy criteria can also be simulated by these above mentioned parameters [14]. The mutants in the testing unit are executed against the test suite in order to weigh the quality or score of a generated test suite. If there is a difference between the results of running a mutant and the original program, the error denoted by the mutant is detected by the test suite. Mutation score is the expected outcome of the mutation testing process. This is achieved by the input test data quality. Therefore mutation score is defined as the ratio of the number of detected errors or bugs against the total number of the implanted faults.

### *D. Test Suite*

A test suite consists of one or more test cases that can be executed individually against mutants. Test Cases and suites [1], Figure: 2 are developed mostly manually by the testers. Therefore smart ideas are supposedly sometimes outperformed by some random testing [5].

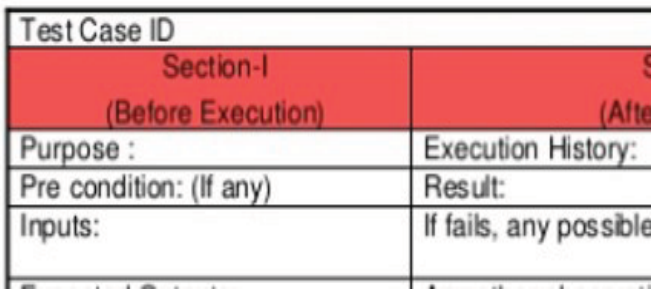


**Fig. 2. Test Case**

## VI. RELATED WORKS

Automated test generation approaches were used for generating test cases in mutation analysis by Enoiu et.al, [8]. The model used mutants and the original program to develop the test cases. The test activity costs in a software were much reduced with effective automated test cases generated by My et.al, [9].

Automated unit tests were generated by Fraser et.al, [10] to check object oriented classes' mutations. Fraser et.al, opinioned that defective mutants should be generated to deliberately find faults in a software. The test suites built by Fraser et.al [11] where two aspects were integrated. The suite cut down on infectious conditions by avoiding redundancy on mutants and then killing maximum number of mutants.

An ant colony optimization is used to generate cost effective strategy in mutation test by Ayari, et al. [12]. The concept is compared with other revolutionary algorithms. Higher mutation scores were generated by May Peter [13] which were much inspired from the immune system algorithms. These mutation systems took very less execution time.

## VII. ONCOLOGICAL MUTATION

### *A. Oncological Mutants*

Mutation is the process of sudden alteration within the genetic system of an organism [15]. This can arise due to the error in the central dogma of biological system which causes DNA damage [16] thereby influencing the DNA replication and translation mechanisms. Mutations can result in normal and abnormal biological mechanism thereby leading to cellular evolution, development of immune system and carcinogenic cells. According to a medical study conducted in 2017, more than 66% of the mutations resulting in cancer are random, while 29% as a result of environmental effect and less than 5% through inheritance [17][18]. Carcinogenesis or oncogenesis is the formation of abnormal cell growth wherein normal cells are transformed into cancerous cells. There are various check points in the cellular, genetic, epigenetic and abnormal cell division steps. Figure: 3 depicts a detailed understanding of the cellular instability leading to cancer cells. When there is an alteration in the normal metabolic cell growth and differentiation of a cell, this results in cancer cells [19]. In the process of protein synthesis, initiated from transcription to translation, there are chances of occurrence of genetic and epigenetic changes due to chromosomal defects. This condition can furthermore result in the DNA nucleotide damage or silencing of various pro-active genes throughout the genetic module of a cell [20] [21]. Oncogenes can also be defined as normal genes undergoing uncontrolled division of cells that express at a higher level in comparison with normal cells. Moreover, it is expressed with the property of altered genes that have unique features depicted during cell division.

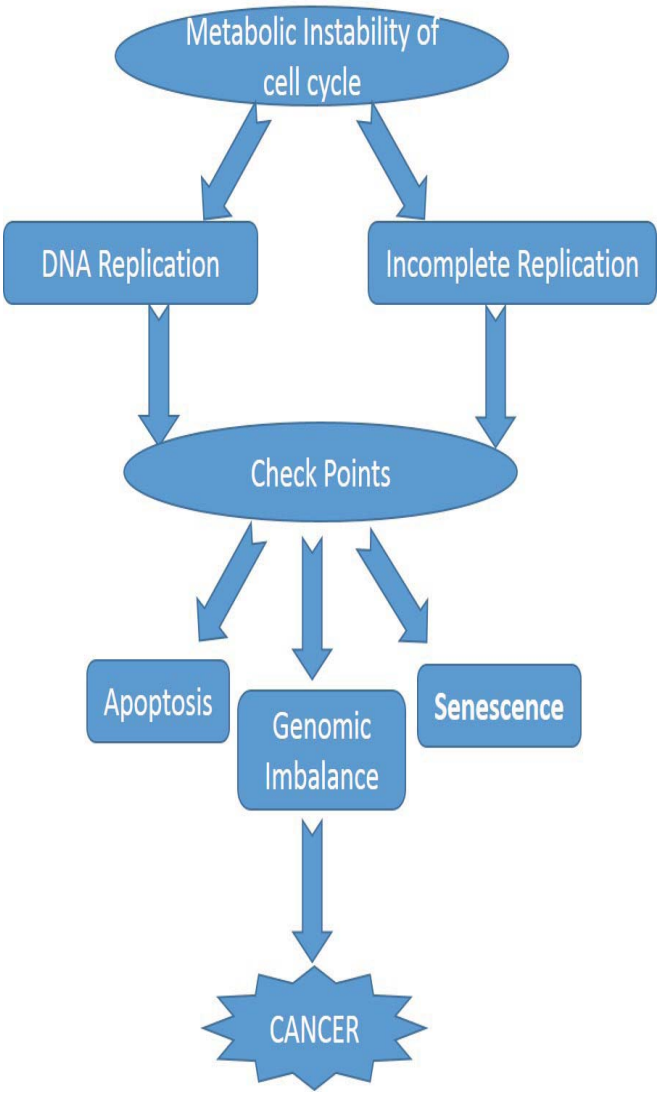


**Fig. 3. Cell division mechanism leading to cancer cells**

### *B. Types of mutation*

(i) Missense Mutation: A point mutation is observed when change in a single nucleotide coding sequence can result in gene sequence coding for entirely different amino acid [22].

Similar to the above state in a software, if the codes are mutated with very less variations such as single operator, we can refer to it similar to a missense mutation.

(ii) Nonsense mutation: The alteration in DNA sequence is observed due to change in one DNA base pair. Thus, this altered DNA sequence disrupts the protein formation [23].

Instead of a single code, a set of codes can be altered similarly to get many nonsense mutants in a software.

(iii) Insertion: The protein developed becomes malfunctioned due to insertion of DNA molecule thereby altering the base pair sequence [24].

A similar situation in a software can occur when values inserted have inaccuracies and such mutants may not actually catch errors. This could be a difficult situation when software developed is to serve accurate values regarding medical sample variations or a signal controlled environment.

(iv)Deletion: The gene alteration occurs as a result of removal of a piece of DNA [25]. Hence, the protein produced as a result of translation has undergone functional changes.

Removal of certain codes in a program to create mutants can cause functional testing requirements which may not be attained. This is a situation where mutants can catch errors typically.

(v) Frameshift: As a result of adding or deleting DNA base pairs to a sequence, there is a change in conformation of reading frame of the code sequence. Thus, in frameshift mutation, the code for each specific amino acid is altered in the complete genetic sequence. Hence, the protein formed will not be functional as expected. This type of frameshift is validated either by insertion or deletion of DNA bases [25].

A frameshift can be compared to a mutant situation where codes in a software are altered invariably here and there. In such situations, no mutant is expected to be an equivalent one. Test data should be effective to catch the errors here.

(vi)Repeat Expansion: These expansion mutations are common in cases where the nucleotide sequence gets repeated in more numbers considering the entire row. In this type of mutation, the sudden alteration occur when short DNA sequences get repeated. Hence, thy also effect the translation mechanism thereby developing a nonfunctional protein moiety [26, 27].

A repeat expansion may be referred to a redundancy in codes or test data that may not be useful in testing a software.

## VIII. PROPOSED FRAMEWORK

Cancer cells and Clinical Trials: Cancer patient samples from various stages are collected in order to test the mutation rate of the cells. These cancer cells are highly mutagenci in nature which is depicted with the aid of various characterization techniques. There are 4 stages in the entire clinical trial process [34] Stage I indicates the clinical trials test conducted for a small group which less than 100 patient samples with novel biomedical composition to evaluate the efficacy and safety. Stage II accounts for a larger group analysis of the novel composition which is cleared from stage I. This validates the further safety concerns and efficacy of the medical component. Stage III investigates further efficacy of the composition with the aid of larger patient samples ranging from hundreds to

thousands with a detailed comparison on standard equipment. This also validates the negative effects on to the samples evaluated and combine the safety results followed up by previous stages [35]. Stage IV clinical trials are the final analysis before the medical composition enters into the market scenario. This elaborates on the bio-availability of the novel composition to the general population samples and compile information on the detailed effects for a long term usage. [35, 36].

For analyzing the above mentioned categories of cancer cells in the clinical trial framework, we have listed few parameters such as Age, Gene mutation rate, Flow cytometry results, Necrotic and apoptotic cell count.

The age group of the patients with pancreatic cancer has been observed to be 40-60 years with increased genetic mutation rate [37]. Apoptosis is a form of cellular death occurring as a part of normal cell growth and differentiation in all multicellular organisms. While, necrosis is a form of cellular injury resulting in the death of cells by the process of autolysis initiated by lysosomes.

The flow cytometric analysis gives the apoptotic effect of cancerous cells which provides characteristic differentiation of fluorescently labelled cells. Thus, they are excited by the laser to emit light at varying wavelengths depicting four different quadrants. In the figure below, Q1 indicates early necrotic cells while Q2 is late necrotic cells. On the other hand, Q3 specifies the features of early apoptotic cells whereas Q4 indicates late apoptotic cells.

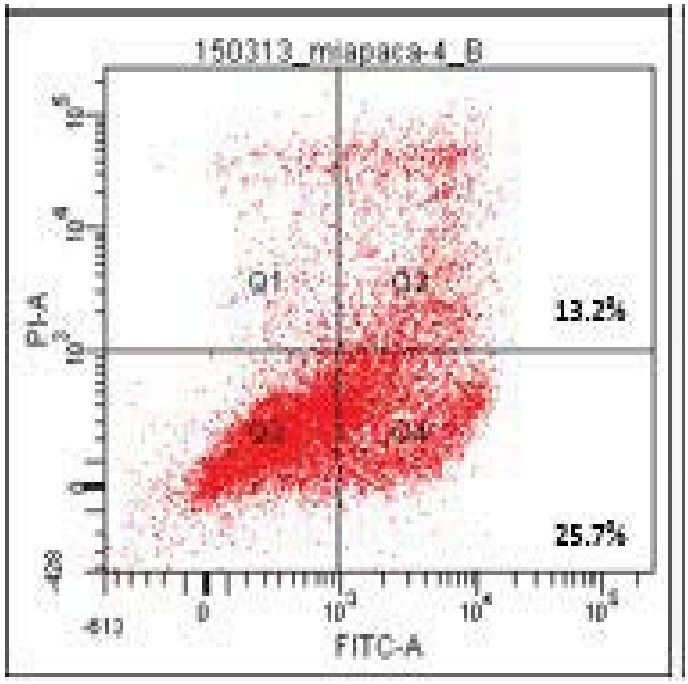


**Fig. 4.**

Flow cytometry result with pancreatic (Mia-Paca 2) cancer cells indicating 13.2% late necrotic cell count and 25.7% late apoptotic cell count.

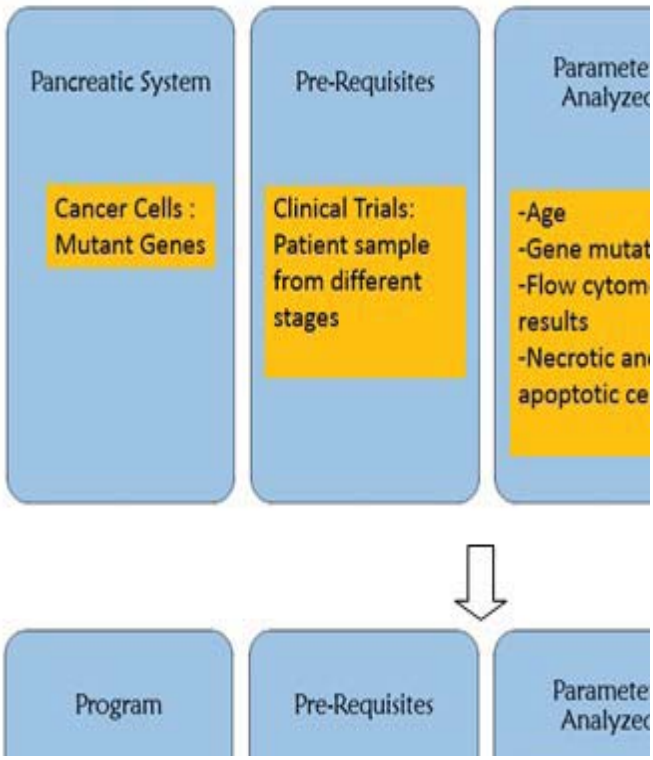


**Fig. 5. Oncological Software Mutation (OSWM)**

The framework OSWM proposed is developed from the oncological test pattern. A similar test suite is suggested for any mutant developed for software. Multiple mutants are generated. The pre-requisites used are sample programs which can be similar to these. The parameters used are the age of the program which is expected to be in the developing stage or to be specified as a modification of an already existing software.

Mutant operators, methods, modules, functions, mutants that are killed or equivalent become the parameters for analysis. While in oncological tests the determinant of cancer factors are an increased p53 expression, in software tests, the mutants' score determine the fault factors.

The situation where mutants are injected with the test cases should have effective test data. All mutants effectively take the same set of test data to satisfy one test case and then followed by a test suite of the test cases. For example; a boundary value of 14, 15, 16 used as test data should go through all mutants developed to see their outputs whether as expected or not. This is similar to a cancer cell test where the cellular samples go through different characterization methods to analyze the stage and type cancer and its diagnosis.

The study argues that an intelligent software with an efficient immune system can therefore aim at testing, diagnosing and healing on its own. A test suite developed from this framework can be trained as an intelligent system with data patterns to generate automated mutants and capture errors as human beings can do manually. The framework is a continuation from the cancer tests analysis obtained from flow cytometric results which is a characterizing equipment. This equipment's functionality can be performed in a machine trained environment for software mutation testing thus making an intelligent system.

## IX. CONCLUSION AND FUTURE WORKS

The study was successful in analyzing the fundamental comparison of concepts in mutation testing software and cancer cell analysis at medical aid. Effectively we developed a framework OSWM, which prove that any software developed fresh or modified version, can depend on the bio-immune system where a human body can tolerate to fault levels functionally but may have problems structurally that is within the system.

We propose to extent this work as a practical test tool in the future with multiple test cases in the suite using the framework.

## REFERENCES


[1] Aggarwal, K.K. and Singh, Y. Software Engineering, New Age International Publishers, Third Edition–2008.

[2] Myers, G.J., Sandler, C. and Badgett, T., The art of software testing. John Wiley & Sons. 2011

[3] Krepska, E., Bonzanni, N., Feenstra, A., Fokkink, W.J., Kielmann, T., Bal, H.E., Heringa, J.: Design issues for qualitative modelling of biological cells with petrinets. In: Fisher, J. (ed.) FMSB 2008. LNCS (LNBI), vol. 5054, pp. 48–62. Springer, Heidelberg (2008)].

[4] Autili, M., Di Salle, A., Gallo, F., Perucci, A. and Tivoli, M., September. Biological Immunity and Software Resilience: Two Faces of the Same Coin?. In International Workshop on Software Engineering for Resilient Systems (pp. 1-15). Springer, Cham. 2015

[5] Meyer, B. Seven Principles of Software Testing. Computer, 41(8), 99-101. 2008

[6] Popentiu-Vladicescu, F. and Albeanu, G.. Nature-inspired approaches in software faults identification and debugging. Procedia Computer Science, 92, pp.6-12. 2016

[7] Spillner, A., Linz, T. and Schaefer, H.. Software testing foundations: a study guide for the certified tester exam. Rocky Nook, Inc. 2014.

[8] Enoiu E.P., Sundmark D., Čaušević A., Feldt R., Pettersson P. Mutation-Based Test Generation for PLC Embedded Software Using Model Checking. In: Wotawa F., Nica M., Kushik N. (eds) Testing Software and Systems. ICTSS 2016. Lecture Notes in Computer Science, vol 9976. Springer, Cham.2016

[9] My, H.L.T., Thanh, B.N. & Thanh, T.K."Survey on Mutation-based Test Data Generation", International Journal of Electrical and Computer Engineering, 5(5) 2015

[10] Fraser, G. & Zeller, A. "Mutation-Driven Generation of Unit Tests and Oracles", IEEE Transactions on Software Engineering, vol. 38,no.2,pp.278-292. 2012.

[11] Fraser, G. & Arcuri, A. "Achieving scalable mutation-based generation of whole test suites", Empirical Software Engineering, vol. 20, no. 3, pp. 783-812. 2015

[12] Ayari, K., Bouktif, S. & Antoniol, G. Automatic mutation test input data generation via ant colony. 2007

[13] May,PeterS.."Test data generation: two evolutionary approaches to mutation testing", University of Kent at Canterbury (United Kingdom), ProQuest Dissertations Publishing, 2007.

[14] Yue Jia, Mark Harman.,qa An Analysis and Survey of the Development of Mutation Testing. IEEE,vol 37(5) 649-678, 2010.

[15] Sharma S, Javadekar SM, Pandey M, Srivastava M, Kumari R, Raghavan SC. "Homology and enzymatic requirements of microhomology-dependent alternative end joining". Cell Death & Disease. 6 (3), 2015.

[16] Chen J, Miller BF, Furano, "Repair of naturally occurring mismatches can induce mutations in flanking DNA", Vol 3, 2014.

[17] "Cancer Is Partly Caused By Bad Luck, Study Finds". Health news from NPR. 2017.

[18] Carol Bernstein, Anil R. Prasad, Valentine Nfonsam and Harris Bernstein. DNA Damage, DNA Repair and Cancer.Intechopen. 2013.

[19] Croce CM. "Oncogenes and cancer". The New England Journal of Medicine. 358 (5): 502–11. 2008.

[20] Lim LP1, Lau NC, Garrett-Engele P, Grimson A, Schelter JM, Castle J, Bartel DP, Linsley PS, Johnson JM. "Microarray analysis shows that some microRNAs downregulate large numbers of target mRNAs". Nature. 433 (7027): 769–73, 2005.

[21] Balaguer F, Link A, Lozano JJ, Cuatrecasas M, Nagasaka T, Boland CR, Goel A "Epigenetic silencing of miR-137 is an early event in colorectal carcinogenesis". Cancer Research. 70 (16): 6609–18. 2010.

[22] Freese E. "The difference between spontaneous and base-analogue induced mutations of phage T4. Proceedings of the National Academy of Sciences of the United States of America. 45 (4): 622–33, 1959.

[23] Goh AM, Coffill CR, Lane DP. "The role of mutant p53 in human cancer". The Journal of Pathology. 223 (2): 116–26, 2011.

[24] Ellis NA, Ciocci S, German J. "Back mutation can produce phenotype reversion in Bloom syndrome somatic cells". Human Genetics. 108 (2): 167–73, 2001.

[25] Hogan, C. Michael. "Mutation". In Monosson, Emily. Encyclopedia of Earth. Washington, D.C.: Environmental Information Coalition, National Council for Science and the Environment. OCLC 72808636, 2010.

[26] Richards RI, Sutherland GR. "Dynamic mutation: possible mechanisms and significance in human disease". Trends Biochem. Sci. 22 (11), 1997.

[27] Viviana Salinas-Rios, Boris P. Belotserkovskii, and Philip C. Hanawalt. "DNA slip-outs cause RNA polymerase II arrest in vitro: potential implications for genetic instability". Nucleic Acids Res. 39 (15) ,2011.

[28] Dimasi, Joseph A; Grabowski, Henry G; Hansen, Ronald W. "Innovation in the pharmaceutical industry: New estimates of R&D costs". Journal of Health Economics. 47, 2016.

[29] Van Spall HG, Toren A, Kiss A, Fowler RA. "Eligibility criteria of randomized controlled trials published in high-impact general medical journals: a systematic sampling review". JAMA. 297 (11): 1233–40, 2007.

[30] A. Jemal, R. Siegel, E. Ward, Y. Hao, J. Xu, T. Murray, and M.J. Thun, Cancer statistics. CA. A Cancer journal for clinicians 58,71 2008.

[31] S. Yohe, Y.Masahiro, K.Jun-ichiro, K.Yoshikatsu, and M.Yasuhiro, Antitumour activity of NK012,SN-38 incorporating polymeric micelles, in hypovascular orthotopic pancreatic tumour. European journal of cancer 46 ,650 ,2010.

[32] A.L. Warshaw, and C. Fernandez-del, Pancreatic carcinoma. The new engalnad journal of medicine 326,455, 1992.

[33] Snima KS, Nair RS, Nair SV, Kamath CR, Lakshmanan VK.Combination of Anti-Diabetic Drug Metformin and Boswellic Acid Nanoparticles: A Novel Strategy for Pancreatic Cancer Therapy. 11(1),93-104,2015.

[34] Medical Devices, Premarket Clinical Studies for Investigational Device Exemption". US Food and Drug Administration. 17 March 2017. Retrieved 2 October 2017.

[35] Sertkaya, Aylin; Wong, Hui-Hsing; Jessup, Amber; Beleche, Trinidad. "Key cost drivers of pharmaceutical clinical trials in the United States". Clinical Trials. 13 (2): 117–126, 2016.

[36] Lang T, Siribaddana S. "Clinical trials have gone global: is this a good thing?". PLoS Medicine. 9 (6): e1001228, 2012.

[37] J. Permert, I. Ihse, L. Jorfeldt, H. von Schenck, H.J. Arnqvist and Larsson J, Pancreatic cancer is associated with impaired glucose metabolism. European Journal of Surgery 159 (2), 1993.