International Journal of Information and Computation Technology.
ISSN 0974-2239 Volume 3, Number 7 (2013), pp. 625-632

http://www. irphouse.com /ijict.htm

Software Testing through Evidence Gathering

Priyanka Mathur [#1] and Swati V. Chande[*2]

[1]Department of Computer Science, The IIS University, Jaipur, INDIA
[2]International School of Informatics and Management, Department of Computer Science, Jaipur, INDIA.

Abstract

An Evidence-based approach is using a best available evidence for making a thoughtful decision about a given set of problem. Evidence-based approach is an amalgamation of individually gained expertise with the evidences gathered through an organized research based approach. Its basic principles are that all practical decisions made should 1) be based on research studies and 2) that these research studies are selected and interpreted according to some specific norms characteristic for Evidence Based Practice[EBP].

Many software testing techniques are proposed to test various types of software and based upon the evidences gathered an effective testing methodology is adopted for a software. This is evidence based approach for software testing.

The software techniques are classified on the basis of three criteria's a) stepwise code reading b) performing functional testing by adopting boundary value analysis and c) all the statements are covered using structural testing. The study compares the strategies with respect to fault detection effectiveness and fault detection time complexity.

In Evidence-based software engineering (EBSE), all the experiences are properly documented in order to inform software practice adoption decisions. In this research paper, the study factor would be the technology of interest. The technological specifications should be very detailed and not at a very high level of abstraction that is the software lifecycle and all the design methods should be properly read and documented and only then should the engineer collect evidences on it and design the software generation model.

In this paper we analyze the gathered evidence so as to classify the testing strategies on the basis of applicability and types of testing.

Various software testing strategies were studied in which white box testing and structural testing are the most preferred methods when using GA (Genetic Algorithm) and SA(Simulated Annealing) as a technique.

1. Introduction

Meeting the objective of the paper software testing techniques identified are

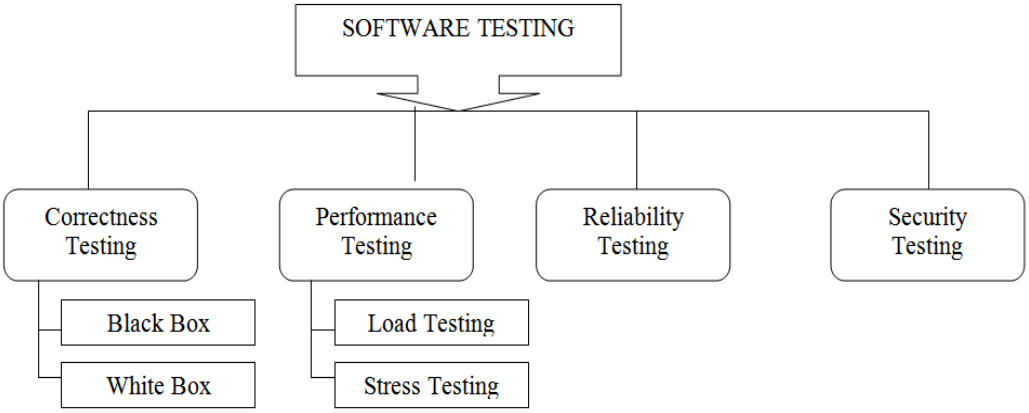

Fig. 1: Testing Techniques.

In accordance with the aforesaid objectives, research papers from eminent researchers were studies and Basili (1990) suggested that for the success of a software product software testing and fault detection activities should be exactly and adequately understood, as they are very crucial for the success of a software product. Thus an experimentation methodology is applied to test the software effectively. Some common testing techniques are applied to different types of software and software testing effectiveness is measured on the basis of several factors like:

- Testing technique
- Software type
- Fault type
- Tester experience

And an intercommunication among all these factors.

The most commonly referred software testing techniques are

- Functional testing (black box approach)
- Structural testing (white box approach)
- Code reading

In Code reading identification of subprograms, evaluation of their functionality is done. Further all subprograms are integrated and again their functionality is tested.

Bergstra (2012) suggested a new software testing strategy: Instruction Sequence testing. In the paper the researcher compares testing from the point of view of two different classical definitions of testing. The first definition by King (1976) says that "in testing a small sample of data that the program is expected to handle program is presented to the program. If the program is judged to produce correct results for the sample it is assumed to be correct." And the second definition as per Singh (2012) is "Testing is the process of executing a program with the intent of finding faults".

A comparative analysis of the first theory reflects that it's a program working on a machine that produces output, and there may or may not be any human intervention in testing.

The second definition reflects that a test is successful if and only if it finds faults. The author crafts a term **Polinseq** which means polyadic Instruction Sequence testing wherein a program is tested instruction wise, marking a difference between program testing and software testing.

The complexity of the program and program testing makes **Polinseq** as a good testing technique but not a risk free technique.

Harman (2009) ET al. is of the opinion that software testing is the process to measure the quality of developed software. Quality here encompasses completeness, correctness, security and other non functional requirements like reliability, capability, maintainability, efficiency, portability, compatibility and usability.

Evidence based software testing holds a good weightage in this field and more than half the percentage papers are published regarding software testing.

Evidence based or Search based test data generation is the possible number of inputs to the program or test cases and their fitness function. As an example taken in this report to achieve branch coverage the fitness function accesses the closeness of test to executing an uncovered branch; in order to find worst case execution time, fitness is simply the duration of execution for the test case in question.

Gathering data from the research paper presented by Khan(2010) the conclusions drawn are that:

- The main aim of testing can be quality assurance, validation and verification.
- Automated testing can be performed in well controlled softwares.
- A successful testing technique uncovers an undiscovered error.
- Generally software testing is done to affirm the quality of software by systematically testing the software in controlled circumstances.

The testing techniques listed above are suitable for Object Oriented Paradigms. There has been much other work on structural test data generation for the OO paradigm. And to analyze the gathered evidence a comparative study of objective/fitness function and problems faced in are listed in Table 1

Table 1: Comparative study of objective/fitness function.

Testing technique	**Technique**	**Description**	**Objective /Fitness function**	**Problem faced**	**Source**
Structural Testing	GA, SA (Simulated annealing)	Branch coverage, data flow coverage,decision-coverage.	Maximise path-coverage	tended to avoid the branches that were hard to cover.	Girgis, Xiao [7]
Structural Testing	GA	Combine non-functional testing goals with coverage based adequacy criterion as a multi objective problem.	Maximise branch coverage and dynamic memory allocation	Easy to detect faults may become harder to detect when they interact	Lakhotia [8]
White-box testing	GA	Test data generation	Maximise path coverage	Manual target paths identification requires tester creativity, and more time	Ahmed and Hermadi[9]
Structural Testing	Genetic algorithms (GAs) and evolutionary strategies (ESs)	Test data generation	Maximise coverage	one parameter may not have effect on another function	Alba and Chicano[10]
Unit testing	GP	Distance function	Automatic bug fixing	GP is computationally expensive	Arcuri[11]
Structural Testing	GA	Test data generation for OO software	Maximise data-flow (d-u) coverage		Liaskos[12]
Model Based Testing	ACO	Automatic test sequence generation	Maximise all-state coverage and feasibility		Li et al.[13]

Mutation Testing	Genetic Programming	Generate and evaluate test cases for the mutation testing.	Generation of test data to kill mutants	GP is computationally expensive	Emer and Vergilio[14]
Temporal Testing	Evolutionary Algorithm	Verifying worst/best case execution time	Optimise worst/best case execution time	EA alone is not sufficient for a thorough and comprehensive test of real–time systems.	Pohlheim and Wegener [15]
Regression testing using slicing	Manual	Coverage-focused, slicing			Gupta [16]
Integration testing and software regression at the integration level.		Procedural-design firewall			Leung and White [17]
Unit testing		Data flow coverage based			Harrold and Soffa [18]
Regression testing		Modification-focused, minimization, branch and bound algorithm			Fischer , Hartman and Robson [19]

Thus a comparative analysis of the techniques used and the testing strategies used reflect that if more efforts are done on path coverage then White box testing can prove to be very effective.

References

[1] V. R. Basili and R. W. Selby, "Comparing the effectiveness of software testing strategies," IEEE Trans. Software Eng., Vol. Se-13, No. 12, December 1987.

[2] Bergstra, J.A.: About Instruction Sequence Testing. arXiv:1201.3929v1 [cs.SE], 18 Jan 2012.

[3] King, J.C.: Symbolic execution and program testing. Communications of the ACM 19 (7), 385–394, 1976

[4] Singh, Y.: Software Testing. Cambridge University Press, Delhi, India, (2012), ISBN 978-1-107-01296-7.

[5] Mark Harman, S. Afshin Mansouri and Yuanyuan Zhang, "Search Based Software Engineering: A Comprehensive Analysis and Review of Trends Techniques and Applications". Technical Report TR-09-03, April 9, 2009

[6] Mohd. Ehmer Khan, "Different Forms of Software Testing Techniques for Finding Errors". IJCSI International Journal of Computer Science Issues, Vol. 7, Issue 3, No 1, May 2010

[7] Girgis, M. R. (2005). Automatic Test Data Generation for Data Flow Testing using a Genetic Algorithm. Journal of Universal Computer Science, 11(6), 898–915. AND Xiao, M., El-Attar, M., Reformat, M., and Miller, J. (2007). Empirical Evaluation of Optimization Algorithms when used in Goal-oriented Automated Test Data Generation Techniques. Empirical Software Engineering, 12(2), 183–239.

[8] Harman, M., Lakhotia, K., and McMinn, P. (2007a). A Multi-Objective Approach to Search-based Test Data Generation. In Proceedings of the 9th annual Conference on Genetic and Evolutionary Computation (GECCO '07), pages 1098–1105, London, England. ACM.

[9] Ahmed, M. A. and Hermadi, I. (2008). GA-based Multiple Paths Test Data Generator. Computers & Operations Research, 35(10), 3107–3124.

[10] Alba, E. and Chicano, F. (2008). Observations in using Parallel and Sequential Evolutionary Algorithms for Automatic Software Testing. Computers & Operations Research, 35(10), 3161–3183.

[11] Arcuri, A. (2008). On the Automation of Fixing Software Bugs. In Proceedings of the Doctoral Symposium of the IEEE International Conference on Software Engineering (ICSE '08), pages 1003–1006, Leipzig, Germany. ACM.

[12] Liaskos, K., Roper, M., and Wood, M. (2007). Investigating Data-Flow Coverage of Classes Using Evolutionary Algorithms. In Proceedings of the 9th annual Conference on Genetic and Evolutionary Computation (GECCO '07), 70 pages 1140–1140, London, England. ACM.

[13] Li, H. and Lam, C. P. (2005a). An Ant Colony Optimization Approach to Test Sequence Generation for Statebased Software Testing. In Proceedings of the 5th International Conference on Quality Software (QSIC '05), pages 255–264, Melbourne, Australia. IEEE Computer Society.

[14] Emer, M. C. F. P. and Vergilio, S. R. (2002). GPTesT: A Testing Tool Based On Genetic Programming. Proceedings of the 2002 Conference on Genetic and Evolutionary Computation (GECCO '02), pages 1343–1350, New York, USA. Morgan Kaufmann Publishers.

[15] Pohlheim, H. and Wegener, J. (1999). Testing the Temporal Behavior of Real-Time Software Modules using Extended Evolutionary Algorithms. Proceedings of the Genetic and Evolutionary Computation Conference (GECCO '99), volume 2,page 1795, Orlando, Florida, USA. Morgan Kaufmann.

[16] Gupta, R., Harrold, M.J., and Soffa, M.L. 1992. An approach to regression testing using slicing. In Conference on Software Maintenance 1992 (Cat.No.92CH3206-0). IEEE Comput. Soc. Press, 299-308.

[17] Leung, H.K.N. and White, L. 1990. A study of integration testing and software regression at the integration level. In Proceedings. Conference on Software Maintenance 1990 (Cat.No.90CH2921-5). IEEE Comput. Soc. Press, 290-301.

[18] Harrold, M.J. and Souffa, M.L. 1988. An incremental approach to unit testing during maintenance. In Proceedings of the Conference on Software Maintenance - 1988 (IEEE Cat. No. 88CH2615-3). IEEE Comput. Soc. Press, 362-7.

[19] Hartmann, J. and Robson, D.J. 1990. Techniques for selective revalidation. IEEE Software.7(1),31-6.