The Factors of Testing Techniques for Object Oriented Software

Sanjeev Patwa[#1], Anil Kumar Malviya[#2], Narendra Kumar[#3]

#1 Department of Science, FASC, MITS, Deemed University, Lakshmangarh, Sikar, Rajasthan, India, +91 9414358986.
#2 Department of Computer Science & Engineering, KNIT, Sultanpur, U.P., India, +91 9415797975
#3 Department of Science, FASC, MITS, Deemed University, Lakshmangarh, Sikar, Rajasthan, India, +91 9413857182

ABSTRACT

One of the major problems within the object oriented software testing area is to select a suitable testing technique. Among numerous available testing techniques, many are rarely used, and few of them are used again and again. Testers have little information about the available techniques, their usefulness, and generally towards suitability of the project at hand, which are based on their decision where testing techniques are to be used. This paper presents the several factors that affect the testing techniques in OO software and on the basis of significance of these factors; one can select the suitable testing technique. The future scope of this article will be to analyze the factors by conducting survey on developers and testers of the industry and can be useful to find their significance.

Key words: Software testing techniques, Object oriented software.

Corresponding Author: Sanjeev Patwa

INTRODUCTION

Software testing is a very broad area, which involves many other technical and non-technical fields, such as specification, design and implementation, maintenance, process and management issues in software engineering. Our study focuses on testing techniques, which can be used in Object oriented software testing that determine different criteria for selecting the test cases and will be used as input to the system under examination. The knowledge for selecting testing techniques should come from studies that empirically justify the benefits and application conditions of the different techniques. It is not unusual for a software development organization to expend between 30 and 40 percent of total project effort on testing. Testing techniques are used to find a suitable set of test cases. There are now tens types of techniques, which raise the question of basic difference between these techniques. We can find distinctions in some literatures, e.g. mechanical (testing books) and technical and context aspects (theoretical research articles, simulations and experiments) of the techniques. However, 'what the best suited techniques for evaluating a given system aspect?' remains an open question [1]. The truth is that, although there are some papers that compare techniques, there are no studies that analyze the

applicability conditions of a technique at length, or assess for each technique, and we don't have the relevant parameters for its applicability conditions.

Although the question of the best-suited techniques for developing the set of test cases for testing a given system apparently poses enormous difficulties, it is, nevertheless, a question which the testers face every time when they have to test a system [2]. The problem we address here is to identify relevant factors which can affect testing techniques used in object oriented software. The aim of solving this problem is to help the testers to choose the best suited testing techniques for each project in selection process and test case generation process. Testing in an OO context must address the basics of testing, that is, a base class and the code that uses the base class. Factors that affect this testing are inheritance, dynamic binding and many other OO characteristics. Daly et al. [3] showed results suggesting that, beyond a certain level, inheritance was a serious hindrance to maintainability. Also, the characteristics of OO technologies (such as decentralized architecture, genericity, encapsulation, inheritance, client-server relationships and polymorphism) set a number of new challenges from the perspective of testing. As a result, traditional testing approaches or techniques must be revisited to take into account OO mechanisms [4, 5, 6].

The paper is organized as follows: Section 2 discusses related work, Section 3 is related to testing methods which can be used in conventional and OO software, and Section 4 describes the factors those affects test techniques in OO software which is the goal of the paper. Section 5 discuses conclusion and future work.

RELATED WORKS IN THE TESTING TECHNIQUE AREA

The studies in the testing area are presented here. They have been divided into a series of studies or lines of research to ease their study [2].

The focus of structural testing techniques is on data flow and control flow testing techniques. This family encompasses several studies by Rapps and Weyuker [7], [8], Weiser et.al. [9], Clarke et.al. [10], [11], Ntafos [12], Zeil [13] and Frankl and weyuker [14, 15 and 16].Other studies follow on from the studies discussed in above section. These studies dealing with data flow testing techniques and seeking to observe the differences between these techniques.The studies by Weyuker [17], Frankl and weiss [18],Bieman and Schultz [19] have been examined. The study by Bieman and Schultz rectifies some of the results obtained by Weyuker. The main emphases of these studies are on experimentation.

Some testing techniques employ knowledge on the geometric shape of failure patterns. This type of research has been conducted by several researchers like Duran and Ntafos [20],Hamlet[21], and Weyuker and Jeng [22].All these studies are an extension or continuation of each other. They all focus on the study of partition and random testing techniques.

Some studies focuses on Functional and Structural Testing Techniques these techniques are conducted by Myers[23], Basili and Selby [24] , and wood et. al.[25]. These are the empirical studies who investigate the differences between functional and structural testing techniques.Other family of research focuses on the analysis of different mutation testing techniques. The main work in this area has been done by Offut and Lee [26]. Wong and Mathur [27] compare mutation testing with other white box testing techniques. Generally these studies aim to find out the costs and benefits of using the different techniques.

The studies on regression testing techniques are experiential, except the theoretical studies by Rothermel and Harrold [28] or the combined study by Rosemblum and Weyuker [29].The study aims at comparing the cost and benefits of applying each technique.Other than above studies

some studies could not be classified in any category. These studies do not have any specific objectives towards these categories, but have contribution to research in the field of software testing. This may be their weakest point. The main studies conducted by Frankl et.al [30], Ntafos[31], and Frankl and Deng[32].Most of the studies are experiential not theoretical. The techniques they have studies vary from one study to another. Graves et.al [33] focuses on the percentage of selected cases and the number of faults detected.

In literature most of the study focuses on testing techniques but rarely take consideration of Object oriented Software. As there are several factors which can affect the testing techniques in Object oriented Software. Testing techniques are used to find a suitable set of test cases [2].These test cases certainly depend on the characteristics of software. The decisions made by developers are not so much haphazard as limited insofar as their knowledge of the techniques is. There are two main reasons why developers do not make good choices [2]:

- The information available about the techniques is normally distributed across different sources of information (books, articles and even people). This means that developers do not have an overall idea of what techniques are available and of all the information of interest about each testing technique.
- They have no access to pragmatic information concerning each testing technique unless they have used it before. Developers do not tend to share the knowledge they acquire by using testing techniques with others. This means that they miss out on the chance of learning about the experiences of others.

The paper emphasis on identifying relevant factors which can affect testing techniques used in object oriented software

TESTING METHODS

Test cases are developed using various test techniques to achieve more effective testing. Testing is based on the creation of test cases and test data by team members using a structural (White Box Testing) and/or a functional (Black Box Testing) strategies.

The main characteristics and comparison between white box testing and black box testing are follows:

White-Box Testing:

- Look at the internal structure of a program or programs,
- Organize the unit test and integration test processes,
- Test the detailed design specifications prior to writing actual code using the static analysis techniques,
- Test the program source code using static analysis and dynamic analysis techniques and ensure that testing focuses on the aspects of a program that are most likely to contain errors.

Advantages of this method:

- All independent paths in a module will be exercised at least once;
- All logical decisions will be exercised;
- All loops at their boundaries will be executed;
- Internal data structures will be exercised to maintain their validity

Black-Box Testing:

- Select a subset of test cases that
  -target error prone situations,
  -test as many conditions as possible at one time,
  -verify the accuracy of a program in a manner that also outrules other error situations,
- Conduct portions of unit testing,
- Conduct portions of system testing

Advantages of this method:

- The number of test cases are reduced to achieve reasonable testing;
- The test cases can show presence or absence of classes of errors

According to above discussion, it seems that, in context of Object Oriented Software, white box testing is more suitable but we cannot ignore the importance of black box testing as within the object oriented software development life cycle, testing can occur at different levels: unit test, integration test, and system or acceptance tests. In these different levels we have to use both testing methods.

Unit - The unit is a class, often a "bigger" or more complex thing than a procedural unit which may be a single function or sub-routine. Complicated by inheritance, encapsulation, and dynamic binding (OO systems). A unit test is more complicated and more effective in the overall OO system than with conventional one.

Integration- It focuses on interactions among classes and complicated by dynamic binding. It recommends that unit be integrated in an incremental fashion, a few at a time, not one integration of the system units like "big bang". This approach lets interface problems to be detected as early as possible but may complicate the design of a test since the cluster of units may not correspond neatly to a functional grouping. When the integration units don't comprise a functional grouping, the test may be of necessity, focus on structural issues (white box testing).

Acceptance- These are largely functional based tests which ensure that all of the used cases appear in a test. It can weigh the tests to the most important functionality (Black Box testing).
The discussion above leads us to conclude that, treating functional and structural testing as two independent test methods; we treat one as supplementation of the other. It means structured testing is supplementary to functional testing. One can perform structural testing as part of functional testing by measuring code coverage and using this measure to enhance tests generated with functional testing [34].

FACTORS AFFECTING TESTING TECHNIQUES IN OO SOFTWARE

In the past two decades, a number of soft-ware integration testing approaches have been used to perform software integration testing, such as top-down, bottom-up, sandwich, and "big bang". Since all of them were designed for integrating components in a traditional program, they might not be applicable to object-oriented programs due to the differences in their structures and behaviors. The first is the structural difference between an object-oriented program and a traditional program [35, 36].
For example, a conventional program consists of three levels of components: a) functions (or procedures), b) modules, and c) subsystems. The structures of these components can be represented (or modeled) as call graphs, data-flow and control-flow graphs. However, an object-oriented program consists of four levels of components: a) function members defined in a class, b) classes, c) groups of classes, and d) subsystems. The other major difference between an object- oriented program and a conventional program is their behaviors. The most important test techniques are illustrated in figure 1.

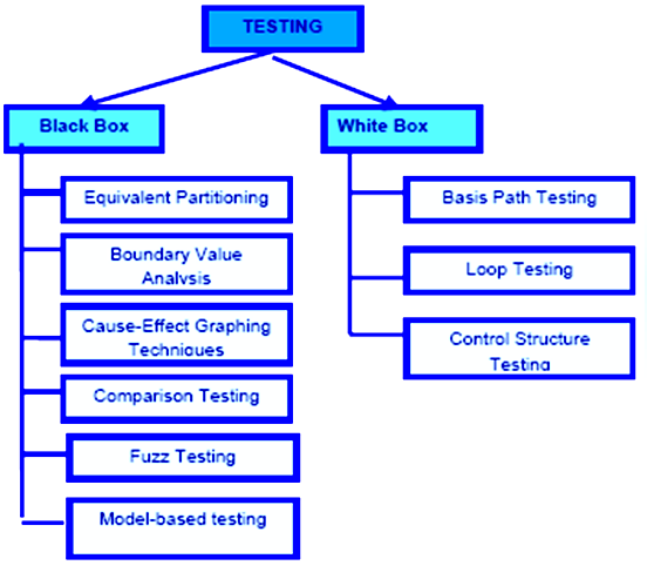

Fig 1. Important Testing Techniques

Here, all the techniques of white box are related to logic, code, loop and internal structure of the programs while all the techniques of black box are related to behaviour and expected results only. These testing techniques can be applied to conventional software as well as OO software. In OO software, white box testing techniques has higher weightage as we have to take several characteristics in consideration like inheritance, polymorphism, massage passing etc. Apart from these well-known characteristics, there may be several other factors, which may impact on selecting the testing techniques for OO software. These factors may be grouped under different categories according to their temperament which are as follows:

A. GENERAL FACTORS

1. Complexity in logic: McCabe's V(G), Halstead's E and program size are well-known complexity measures. There exists a significant relationship between each of these measures. Program size (Kiloline of code: KLOC) is used as a measure of program complexity. As in OO software, complexity is depend on number of classes, their inheritance level, coupling of classes, number of loops and decision statements. The "High"' level of these factor means the program size is greater and program is complex.

2. Program Categories: There may be four program categories, which may impact in selecting the testing techniques: operating system, communication control program, database management system, web application, and languages processor.

3. Difficulty of programming language: As difficulty level of OO programming languages may be higher than conventional programming language and we need to have a solid understanding of Memory management and OOP to use them effectively, and thus this may be a factor, which affects the testing technique.

4. Amount of programming effort: The deliberate programming effort may be regarded as effective for reducing the number of errors made. This is calculated in man-years.

5. Level of programming technologies: The programming technologies are classified into four categories: design techniques, documentation techniques, programming techniques (including programming languages) and development of computer access environment.

6. Percentage of reused modules: When people develop some new software products or when they update the old version of their software products, they usually keep some of the module of the code which can be reused, and added in some new ones.

7. Programming language: Different programming languages have different complexities and structures. Therefore the possibility for different languages to introduce errors is different and accordingly testing techniques may differ.

B. ANALYSIS AND DESIGN

8. Requirements Analysis: Requirements usually come from customers. Based on the requirements software developers generate specifications. Usually customers and developers meet to verify the requirements and achieve understanding of the problems. Requirement analysis is fundamental for selecting testing technique.

9. Volume of program design documents: This factor includes the volume of designing documents/files for the program. In the case of OO programs designing volume may be higher as compared to conventional programmes.

10. Design Methodology: Different design methodologies for the same software may have different impacts on the quality of the final software products. There are different types of design methodologies: structured design, functional design and Object-Oriented Design Approaches.

11. Relationship of Detailed Design and Requirement: At the end of the design phase, detailed design is compared with requirements. Inspections are performed to verify whether the function designs meet the requirements. In OO software, there are many designing approaches are available like UML, ERD and state transition diagrams. Modifications can be made to remove the misunderstanding between the customers and developers.

12. Frequency of program specification change: As the program specification changes the design, technique and also the logic of program and this also affects the testing technique of OO software.

13. Development Management: Development management includes all the organization and decision-making activities. From the specification phase to design, code, and testing and even operational phase, development managers schedule the meeting time, keep all participants in touch and keep track of the development progress and work standards, give instructions and finally make decisions.

14. Work Standards: Work standard is the norm that needs to be obeyed by the developing team. This could be company standard or group standard. Work standard indicates the products to be made at each phase, design document format, design document description level and content, testing technique, and the items to be checked in verifying the design documents.

C. CODING

15. Programmer/Tester Skill: Skills has direct impact on the selecting of testing technique for software products. This can be defined as the average number of years of programming experience of programmers: This can be calculated as the ratio between total year of experience of all programmers/tester and total no of programmers/testers.

16. Programmer/Tester Organization: Programmer/Tester organization (PO) is defined as the percentage of high-quality programmers. PO is computed as follows: PO = n_h / n ; where n_h is the number of programmers, whose programming experience is more than six years, and n be the total number of programmers in organization. As PO is high, we can select better testing technique.

17. Development Team Size: Team size has impact on the selection of testing technique. Some one believes that large team size will improve the quality of the software since there are more people involved in the development process. But others claim that smaller but experienced development team will be better.

18. Program Workload: During the software development, stress factors in terms of "work contents" such as schedule pressure and too much work are the major factors. This includes tester's mental and physical stress. This factor affects in selecting the Testing technique.

19. Domain Knowledge: Domain knowledge refers to the programmer's and tester's knowledge of the input space and output result. Insufficient knowledge may cause problems in coding and testing procedures.

20. Human Nature: This refers to the tester and developers characteristics, including the ability to avoid the making of working mistakes, careless work omission and selecting the testing technique.

D. Hardware Systems

In making the testing of object oriented software primary memories, processors, storage devices and system, software may have their impact as test cases that have to be tested or executed on software which are installed in machines.

Apart from these all factors, there may be several OO software matrices, which might impact on testing techniques of such software. These metrics measures program or design complexities. These are of direct relevance to testing techniques. A complex design (OO software) is likely to require more test efforts to obtain a given level of defect density than traditional software with less complexity

CONCLUSION AND FUTURE SCOPE

The software testing techniques particularly used for Object oriented software has been a field of great interest to SE researchers. Considering the pace of technology change in software engineering, it is however, evident that OO programs requires a great deal and more resources are currently being spent. The final quality of the software system will depend on these testing techniques.

The hurdle, which software tester or developer faces in selecting the testing technique, is lack of information. The information on testing technique, can be used for OO software, is distributed across different sources of information, and at worst, nonexistent. Further, research in this field is needed to grow to undertake the challenges that we are facing. In this paper, we try to identify some factors, which may impact on selecting the testing technique for OO software. Although, all the factors, which we have discussed in this article, may not directly impact on OO software testing techniques; as a future scope we further need a deep survey from the people involved in these types of software development or testing to analyse the significance of these factors. Finally, we suggest that, a comparative study between testing techniques for traditional software and OO software will be an interesting field of research in near future.

REFERENCE

[1] Bertolino, A. 2004. Guide to the knowledge area of software testing. *Software Engineering Body of Knowledge IEEE Computer Society* (February. http://www.swebok.org).

[2] Vegas, S. and Basili. 2005 A characterization schema for software testing techniques. *Empirical Software Engineering*, 10, 437–466.

[3] Daly, J., Brooks, A., Miller, J., Roper, M., and Wood, M. 1996. Evaluating inheritance depth on the maintainability of object-oriented software. *Empirical Software Engineering* 1(2): 109–132.

[4] Binder, R. 1996. Testing object-oriented software: a survey. Software testing, Verification & Reliability 6(3/4):125–252.

[5] McGregor, J. D. 1997. Quality assurance. Journal of Object-Oriented Programming. Monthly columns.

[6] Kung, D., Hsia, P., and Gao, J. (eds.) 1998. Testing object-oriented software. IEEE Computer Society.

[7] Rapps, S. and.Weyuker, E 1985. Selecting software test data using data flow information, IEEE Transactions on Software Engineering, SE-11(4), 367-375.

[8] Rapps S. and Weyuker, E.J. 1982. Data Flow analysis techniques for test data selection. In proceedings of the 6th International conference on Software Engineering. pp. 272-278, Tokio, Japan.

[9] Weiser, M.D., Gannon, J.D., and McMullin, P.R 1985. Comparison of Structural test coverage metrics. IEEE Software, 293, 80-85.

[10] Clarke, L.A., Podgurski, A., Richandson, D.J. and Zeil, S.J. 1985. A comparison of data flow path selection criteria. In Proceedings of the 8th International conference on Software Engineering. pp. 244-251, London, U.K.

[11] Clarke, L.A., Podgurski, A., Richandson, D.J. and Zeil, S.J. 1989. A formal evaluation of data flow path selection criteria. IEEE Transaction on Software Engineering, 15 (11), 1318-1332.

[12] Ntafos, S.1988. A comparison of some structural testing strategies, IEEE Transaction on Software Engineering, 14(6), 368-371.

[13] Zeil, S.J. 1988. Selectivity of data flow and control flow path criteria. In proceedings of the 2nd workshop on Software testing, Verification and analysis, 215-222. Baull, Canada.

[14] Frankl P.G. and Weynker, E.J. 1991. Assessing the fault detecting ability of testing methods. In proceedings of ACM SIGSOFT'91, Conference on Software for Critical

Systems, pp. 77-91. ACM Press.

[15] Frankl, P.G. and.Weynker, E.J. 1993a. An analytical comparison of the fault detecting ability of data flow testing techniques. In proceedings of the 15th International Conference on Software Engineering, pp. 115-124, Maryland.

[16] Frankl, P.G.and Weynker, E.J. 1993 b, A formal analysis of the fault detecting ability of testing methods, IEEE Transactions on Software Engineering,19(3), 202-213

[17] Weyuker, E. 1998. An empirical study of the complexity of data flow testing. In proceedings of 2nd workshop on Software testing, Verification and Analysis, 188-195.

[18] Frankl, P.G. and Weiss, S.N. 1993. An experimental comparison of the effectiveness of branch testing and data flow testing. IEEE Transactions on Software Engineering, 19(8), 774-787.

[19] Bieman, J.M. and Schultz, J.L. 1992. An empirical evaluation (and specification) of the all -du-paths testing criteria. Software Engineering Journal.43-51.

[20] Duran, J.W. and Ntafos, S.C.1984. An evaluation of random testing. IEEE Transactions on Software Engineering .SE-10(4), 438-444.

[21] Hemlet, R.G. 1998. Special section on software testing. Communications of the ACM, 31(6), 662-667.

[22] Weyuker, E.J. and Jeng, B. 1991. Analyzing partition testing strategies, IEEE Transaction on Software Engineering, 17(7), 703-711.

[23] Mayers, G.J. 1978. A controlled experiment in program testing and code walk-troughs/inspection. Communication of the ACM, 21(9):760-768.

[24] Basili, V.R.and Selby, R.W. 1987. Comparing the effectiveness of software testing strategies. IEEE Transactions on Software Engineering, .SE-13(12), 1278-1296.

[25] Wood, M., Rooper, M., Brooks, A. and Miller, J. 1997. Comparing and combining software defect detection techniques: A replicated empirical study. In proceedings of the 6th European software Engineering Conference, Zurich, Switzerland.

[26] Offut, A.J. and Lee, S.D. 1994. An empirical evaluation of weak mutation. IEEE Transactions on Software Engineering, 20(5), 337-344.

[27] Wong E. and Mathur, A.P. 1995. Fault Detection effectiveness of mutation and data flow testing. Software Quality Journal, 4, 69-83.

[28] Rothermel G. and Harrold, M.J. 1998. Empirical Studies of a safe regression test selection Technique. IEEE Transactions on Software Engineering. 24(6), 401-419.

[29] Rosenblum, D.S., and weyuker, E.J 1997. Using coverage information to predict the cost effectiveness of regression testing strategies, IEEE Transactions on Software Engineering, 23 (3), 146-156.

[30] Frankl, P.G., Hemlet, R.G., Littlewood, B. and Strigini. L.1998. Evaluating testing method by delivered reliability. IEEE Transactions on Software Engineering.24 (8), 586-601.

[31] Ntafos, S.C. 1981. On testing with required elements. In COMPSAC'81.Proceedings of the 5th Annual International Computer Software and Applications conference, pp. 132-139, Chicago.

[32] Frankl, P.G.and Deng, Y., Comparison of delivered reliability of branch, data flow and operational testing. In proceedings of the ACM SIGSOFT 2000, International Symposium on Software Testing and Analysis, pp. 124-134, USA.

[33] Graves, T.L., Harrold, M.J., Kim, J., Porter, M. A., and Rothermel, G. 1998.An empirical study of regression test selection techniques. In Proceedings of the 1998 international

Conference on software Engineering, pp.188-197, Japan.

[34] Mathur, A. P. 2008.Foundations of Software testing, First Ed. Pearson Education, New Delhi, India.

[35] Kung, D.C. and Hsia, P. 2009. Object-Oriented Software Testing: Some Research and Development, IEEE Xplore.

[36] Patwa, S., Malviya, A.K. and Kumar, N. 2011. Conventional software testing: A discussion, Journal of Current Computer Science and Technology, 1(8), 435-441.