IJARCCE

ISSN (Online) 2278-1021
ISSN (Print) 2319 5940

International Journal of Advanced Research in Computer and Communication Engineering
ISO 3297:2007 Certified
Vol. 6, Issue 4, April 2017

Design & Development of Model Based Adaptive Testing for Software Quality Assurance

Ms. Ranjana Dalwani[1], Prof. Makrand Samvatsar[2]

M. Tech (SS), Department of CES, PCST, Indore[1]

Professor, Department of CES, PCST, Indore[2]

Abstract: Testing is very important phase of software development lifecycle which includes verification and validation of parameters used for evaluating the software. It aims towards creation of defect free codes with better quality and reliability. Imperfection ID and forecast alongside requires reviewing of item from client end. It directs the improvements to be driven ceaselessly in nearness of testing systems. We have experienced the thorough investigation of different research articles which covers the conceivable outcomes of applying testing through various procedures like segment testing and model based testing. Subsequent to dissecting the issue connected with early era of experiments and the parceling rationales we have recommended a few changes utilizing exhaustive strategies towards test handle enhancements and quality primitives. Analytical evaluations are showing the benefits of work and its probable improvements over other traditional approaches. This paper presents a numerical evaluation of the result on different operating situations and test case conditions. For measuring the robustness of suggested approach several scenarios are made and values are taken. From these values it is clearly identified that the suggested approach is showing better in the direction of software testing and ought to have great future ahead.

Index Term: Software Testing, Automated Testing, Partition Testing, Model Based Testing, Generic Testing, Quality Assurance.

I. INTRODUCTION

In today's world most of the manual operations are converted to software solutions. The working of these programs makes the reduction in user's effort though the dependency is made here for performing the accurate working. Here if the existing attributes of the software fails then it might deviate the system in different directions such as business loss, information loss, commercial downfalls and sometime the losses goes more vibrant. So the ever increasing dependability and parametric growth of software solutions rises in the testing parameters and techniques. The aim is towards creation of defect free codes by means of superior excellence. Automation testing is also known as Test Automation, is when the tester writes scripts and uses software to test the manufactured products. These procedures involve computerization of a labor-intensive process. It is used to re-run the test scenarios that were performed manually, quickly, and repeatedly. Separately from deterioration testing, automation testing is also used to check the submission from load, performance, and stress point of view. It increases the test coverage, improves accuracy, and saves time and money in comparison to manual testing.

Effective Automated Testing

The essential issue we confront in the midst of testing is managing the gigantic number of experiments we need to make and execute. For better outcomes test scope criteria are also incorporated into this computerized part. It describes the principles used to create test cases from the product show. There are two sorts of criteria: information stream and control-stream. They portray the exertion and the way of the outcomes made consequently by a MBT approach [1]. Amid the time a couple attempts in programmed test information eras have been made.A portion of the specialists found that the parcel based information based testing is lower execution and exactness while contrasting and irregular testing. Additionally the disappointment rates secured by arbitrary testing are more than the parcel testing. The higher likelihood of blunder identification in arbitrary testing with finish scope of sources of info utilizing allotment testing can be utilized consolidating for better serving the imperfection evacuation.

- **Adaptive Testing:** It offers some different aspect of testing. It is a feedback oriented testing presenting its strong nature against the irregular and parcel testing. Despite the fact that its application cost and multifaceted nature is higher than others. Segment testing with corresponding distribution is appeared to perform in any event and in addition irregular testing as far as these criteria [3].
- **Component Testing:**The utilization of arbitrarily built test sets to programming parts would seem to offer an indistinguishable advantages from for compilers. Thus the creator considered the ramifications of adding arbitrary

IJARCCE

ISSN (Online) 2278-1021
ISSN (Print) 2319 5940

International Journal of Advanced Research in Computer and Communication Engineering
ISO 3297:2007 Certified
Vol. 6, Issue 4, April 2017

testing to the British Computer Society segment testing standard.

- **Random testing:**For Better comprehension let N be the aggregate number of components in the information space, and assume we need to haphazardly choose and contributions for testing the framework. It might be conceivable on the premise of any likelihood dispersion, i.e., the n sources of info can be chosen autonomously with the in spite of the fact that testing without-substitution is clearly more proficient, the handy execution of a without-substitution inspecting plan is troublesome and regularly not savvy.
- **Model Based Testing:** Testing methodologies which uses model is called model based testing (MBT). Model based testing (MBT) refers to the type of process that focuses on deriving a test model using different types of formal ones, at that point changing over this test show into a solid arrangement of experiments [5]. Models are the middle of the road curios between necessity particular and last code.

II. BACKGROUND

Software size and its handling complexity are increased as the development process proceeds and a modular design gets into shape. Lots of interaction needs to be taken over for getting to and measuring the quality of limits and branches. For successfully measuring the conduct of the product some unwavering quality enhancements and judgment techniques is utilized alongside improvement or after advancement. This procedure is named as programming testing which isused to identify the bugs in program codes parallel as the code progress. Testing involves various resource utilizations and traversing the code blocks at least once for analyzing the behaviour and integration actions. Measuring the strength of code and estimating the testing assets for expelling the bugs may expand the unwavering quality and henceforth it is incorporated into programming improvement life cycle.

The testing can be classified by their working style and conduct. For the most part it is orders as black box and white box testing. In discovery testing the codes are tried as a joined useful pieces and its inner structure is not known. The information sources extents are utilized for applying numerous executing conditions and their conduct. It serves a parametric model which works as a finite time events and calculates the responses of the overall software's. The white box testing as an approach aims towards detecting the internal structure bugs and problems. Its goal is reliability detection and estimation by completely analyzing the internal structure a logic handling by the use of control graphs. It is having very large input space and hence it is quite complicated to handle such methods. Totally the plan is to the discovery of fault that leads to crash that means of above testing strategy. Now, the tester requires effective testing which identifies all the defects with least endeavors. It can be made achievable by successful employments of both black box and white box testing. This work specifically focuses on adaptive partition testing based on design models like UML for improved test cases generation and evaluations.

In random testing, each test case is selected independently. To make the detection of the first failure quicker (that is, to reduce the number of test cases needed to detect the first failure), Malaiya introduced an antirandom testing technique, where the first test case is selected randomly, and each subsequent test case is selected by choosing the one whose total distance to all the formerly executed test cases is utmost. Here in anti-random testing, the total quantity of test cases has to be decided in the first place. The randomness of this method is also very limited because only the first test case is selected randomly; the sequence of all the subsequent test cases is deterministic. The focus will be on determining which combination of UML diagrams, and their associated constraints, may be used to automatically, or semi-automatically, generate test cases for adaptive divisiontesting. Prototype tools will be developed in future to demonstrate the techniques and strategies derived from the proposed investigation. In summary, the study aims to:

(i) Determine what information is necessary to test the integration of components in the process of system composition;
(ii) Given item 1, investigate which individual or combination of UML diagram types, offer sufficient information to generate test cases; The results of this aim, will affect aspects of activities 1 to 5 in the figure
(iii) Develop a strategy that reports on the amount of testable information contained in a model.

III. RELATED STUDY

During the last few years software testing had grown immensely with their strategies. Loads of new and overpowering procedures are produced which enhances testing exhibitions and declines their expenses. Among them, some methodologies demonstrates their solid nearness in the individual regions and are identified with their work are taken here as writing.

In the paper [7], a dynamic divisionstrategy is presented for selecting the test cases through some online feedback mechanism. Here the approach is focuses on online medium for generating the test sequences and starts with selecting criteria of online partition.Additionally the testing is not in light of the codes or interior structure of the projects rather it utilizes just some metadata data and passing and coming up short state of beforehand executed experiments.In spite of the fact that, tackling all issues identified with testing prophet is not possible every time on account of their high

unpredictability. In the paper [8], a portion of the assessment is perform on irregular testing technique for discovery testing. Here the arbitrary experiments are created for recognizing the aggregate bugs from test prophets. It likewise empowers the scope, on the off chance that it is high the likelihood of blunder discovery is increasingly and if the lesser scope is accomplished then the experiments amount is expanded. Some more arrangement is given on the versatile arbitrary testing (ART) are shown in paper [9]. Here the ART is totally examined for exhibiting the conduct of strategy with higher identification rates of shortcomings in contrasting and ordinary irregular testing. The paper likewise proposes couple of new ART calculations fir additionally expanding the viability. The recommended calculation gives comparable working yet the overhead connected with the testing gets lessened. It guarantees that experiments keep up to be generally widened by just selecting new cases from segments which encase no previous experiment. Aside from all the above advantages some more change is given in anti-random testing in [10]. The proposed strategies fundamentally free the reliance of just numerical contributions of anti-random testing. The proposed procedure is more blame discovery rates than any of the irregular testing variations and is tried on different applications. Quantities of experiments are practical and test for recognition of issues embedded by utilizing change testing.

The paper [11] covers some part of parcel testing rationales and conquers its current issue. The work distinguished that if the sub-areas of the testing contributions for the parcel is not homogeneous than their execution are not as wanted and their prosperity likewise not contribute so much certainty. Despite the fact that the code scope parameters in testing is taken for ground as best practice dependably.The paper [12] centres towards additionally enhancing the execution and mistake discovery likelihood of versatile arbitrary testing. It principally expands the blame uncovering capacity of irregular testing by presenting the ART in light of two point apportioning. As indicated by the new calculation of ART-TPP the given are of trying sources of info are partitioned into at least two segment in view of midpoint hypothesis as opposed to direct division of\r rise to division. Here the primary purpose of division instatement is haphazardly produced.The paper [13] concentrates on one of the major troublesome with testing which is its robotized era. This programmed era is performed by earlier making a portion of the era foundation. It diminishes the endeavors and cost of the testing makes the procedure really robotized. The paper concentrates on era of experiments from the utilization of hereditary calculations.The paper [14] deals with automatic generation of feasible independent paths and software test suite optimization using artificial bee colony (ABC) based novel search technique. In this approach,ABC consolidates both worldwide hunt techniques done by scout honey bees and neighborhood look strategy done by utilizing honey bees and passerby honey bees. Test Cases are created utilizing test way succession examination technique as the wellness esteem target work.The paper presents a novel approach to generate the automated test paths [15]. Due to the delay in the development of software, testing has to be completed in a small time. This leads to mechanization of testing since its efficiency and also requires less manpower. In this proposed approach, by using one of the most standard Unified Modelling Language (UML) Activity Diagram, construct the Activity Dependency table (ADT), then generate the Test paths. Then the test paths are prioritized by means of the Tabular search algorithm.

The paper [16] shows that the genetic algorithms can be used to automatically generate test cases for path testing. Using a triangle classification program for instance, analyze comes about demonstrate that Genetic Algorithm based test information can more viably and effectively than the current strategy does. Some authors also proposes a novel approach for diverse model based test case generation [17]. It selects a subset of the generated test suite in such a way that it can be realistically executed and analyzed within the time and reserve constraint, while preserving the fault enlightening control of the unusual test suite to a maximum extent. In this article, to address this problem, we introduce a family of similarity-based test case selection (STCS) techniques for test suites generated from state machines. Model based slicing [18], including the various general approaches and techniques used to compute slices. To understand and test a large software item is an extremely difficult assignment. One approach to utilize this is program cutting system that deteriorates the huge projects into littler ones and another is a model based cutting that breaks down the vast programming engineering model into littler models at the early phase of SDLC (Software Development Life Cycle). An arranged study of the most unmistakable strategies for programmed era of programming experiments, checked on in self-standing segments proposed in [19]. The strategies displayed include: (a) basic testing utilizing typical execution, (b) show based testing, (c) combinatorial testing, (d) arbitrary testing and its assortment of versatile irregular testing, and (e) look based testing. In general, the paper goes for giving a presentation, exceptional and (moderately) short review of research in programmed experiment era, while guaranteeing extensiveness and legitimacy.

IV. PROPOSED SOLUTION

This work evaluates the novel generic adaptive partition testing suggested in [21] the design consideration of models logic. Here the technique generatesthe test information which is altogether circulated in general locale of the isolated segment. In this manner, the recommended technique can turn formal identification to finish investigation and the likelihood of blame discovery is additionally expanded. Generally the versatile irregular testing is of just two structures; separate based and parceling based. The recommended cross breed approach is mix of both the measures for viable assurance of shortcomings with least number of experiments. The main problem with testing is about managing

IJARCCE

ISSN (Online) 2278-1021
ISSN (Print) 2319 5940

International Journal of Advanced Research in Computer and Communication Engineering
ISO 3297:2007 Certified
Vol. 6, Issue 4, April 2017

the expansive number of automated test suite creation with smaller size & less complexity. Consequently, we are focusing on automatic and effective test case handling concept taking in mind the early generation of test cases.

We have parted our work in two identified domains: First is adaptive divisiontesting for optimization and second is design based test data generation (Early Generation). Usually an infinite number of possible tests could be generated from a model. The test forecaster choose test production criterion to select the uppermost priority tests or to ensure good coverage of the system behaviour. One common kind of test generation criteria is based on structural model coverage, using well known test design strategy of path based testing. This is an automated process that generates the required number of high-level (abstract) test cases from the test model using web computation. It could be of control and data flow and applies with a detailed analysis for generating the codes functional blocks where separate input can be passed.

This work suggested a model based testing. It enhances the performance of combinatorial and model based testing in many aspects. It is intended that our strategies will convey to a expensive, how much in order is enough to enable routine creation of test cases in an optimized manner. Our research domain can be separated into two majorly identified domain of interaction testing which reduces the test case size & complexity & a model based approach for early test case generation. For supporting the defined objective we had proposed a design architecture of the concept along with that a test case reduction algorithm IPOG and a DDE algorithm.

It typically involves the following steps

1. Building an abstract model of the behaviour of the designed software system applied under testing process. The model forms a subset of the system requirements.
2. Definition of test selection criteria. The criteria of interaction testing can be defining what test cases to create from the replica.
3. Validating the replica. This is typically done by example conceptual test cases from the model and analyzing them. This step is performed to detect major errors in the replica that may even hold back creation of test cases.
4. Forming abstract tests from the model, using the defined test selection criteria. At this stage, the generated test cases are articulated in provisions of the abstraction used by the copy.
5. Transforming (concretize) the abstract test cases into executable test cases.
6. Executing the test cases. At execution time, an adaptor constituent transforms the output of the scheme to the concept of the model.
7. Analyzing the execution result.

In our research & implementation we tries to prove that the given algorithm & design is well defined for improving efficiency and performance through multiple parameters (Size, Time, Complexity, Cost etc.). It is a well-defined dynamic approach for quality improvements because it provides effective error detection at very low cost

V. RESULT ANALYSIS

The suggested system is completely implemented on the .NET framework which provides various features for serving the complete feature in the form a tool view. Here the tool is also been able to analyzed the generation process on the basis of some of the well known factors such as number of generated test, compete coverage achieved by the generated test, generation time, the system resources such as CPU and RAM utilized etc. The robust experimental analysis shows the tool behaviour setting a milestone in the field of test case generation and coverage analyses.

Table 1: Selective Test Generation

S. No	Test Data	Test Type	Selected Attributes	CPU Utilization	RAM Utilization	Page Faults
1	Test Set 1	Manual	**h , j**	**1%**	**30.05%**	**0**
2	Test Set 2	Manual	**Netscape6.1,XP-Pro**	**5%**	**30.30%**	**0**
3	Test Set 3	UML	**VTR,PTB**	**7%**	**28.76%**	**0**
4	Test Set 4	UML	**Trst , Frst**	**3%**	**29.33%**	**0**

Table 2: Pseudorandom Selective Test Generation

S. No	Test Data	Test Type	Reduced Test Counts	CPU Utilization	RAM Utilization	Page Faults
1	Test Set 1	Manual	**13**	**2%**	**30.094%**	**0**
2	Test Set 2	Manual	**42**	**3%**	**30.34%**	**0**
3	Test Set 3	UML	**6**	**2%**	**28.34%**	**0**
4	Test Set 4	UML	**4**	**1%**	**29.39%**	**0**

IJARCCE

ISSN (Online) 2278-1021
ISSN (Print) 2319 5940

International Journal of Advanced Research in Computer and Communication Engineering
ISO 3297:2007 Certified
Vol. 6, Issue 4, April 2017

Time Based Coverage Analysis

Now the above generated comma separated value (csv) files are passed as an input to the ComCoverage tool which analyses the coverage achieved without partition logic, with partition logic and with pseudorandom generation stages of the solution phases. Table gives an idea that how many possible combinations of test cases are possible and number of test cases selected by the suggested system which is giving 100% coverage in significantly lesser time, which saves lots of efforts of testers.

Table 3: UML InputCoverage for Test Case Analysis through ComCoverage

UML Diagram		Extraction	No. of Pairs Covered in all Combinations	No. Of Test Case Generated	Coverage Achieved	Time Required
1	S1	Success	6	32	96.9%	1.3522
	P1	Success	6	32	100%	1.3522
	R1	Success	6	32	100%	1.3522
2	S2	Success	4	19	94.73%	1.2549
	P2	Success	4	19	100%	1.2549
	R2	Success	4	19	100%	1.2549

Summary: Theabove table shows an effective early test case generation feature of our tool which implies on the data extraction from UML activity diagram. We have developed an algorithm for getting this result. We shows the performance and result evaluation of our tool on the basic of 7 parameters. Firstly the algorithm is capable of extracting the correct data from given UML textual notations. The table shows how our tool effectively reduces the test size in very less time and gives maximum coverage. The feature which we have proposed and implemented is not present in any combinatorial testing tool and serves as add on module for our research. In future its improved versions are likely to be developed.

VI. CONCLUSION

Software testing with adaptive behaviour will always allows some open process for testing and its re-execution. The effective test cases can be determined if the test comes from complete regions and covers at least once each type of input. But all of certain such heavy numbers of inputs are not tested with some minimum attempts. Hence, a new mechanism is required which reduces the test size but increases the code coverage. It works towards assuring the reliability of the system. This paper proposed an integration of adaptive partition testing with design model based logics towards effective and early identification of bugs according even with their priority levels also. Means the module which is most critical should be tested more. It overcomes the existing issues of high testing cost and computation complexity. On the preliminary evaluations the works seems to provide effective solution of testing domains.

VII. FUTURE WORK

Some problems and concepts that remain unaddressed and can be performed in future are as follow:

(i) In future with the help of proposed scheme we are looking for an enhancement of design based development testing to requirement based development testing.

(ii) We can also extends our work towards inculpating all the various tool which we used as analysis tools such as ComCoverage, UML to text converter like PlantUML in an single entity.

(iii) In our proposed scheme, we provide the simulation for various test performance parameters for combinatorial test. In future we can also simulate some other test parameters such as unit test, branch condition, equivalence divisionand other methods for design base test scenarios development.

REFERENCES

[1] Jon Edvardsson, "A Survey on Automatic Test Data Generation", in Proceedings of the Second Conference on Computer Science and Engineering in Linkoping, pages 21{28.ECSEL, October 1999.

[2] Arilo C. Dias Neto, Rajesh Subramanyan, Marlon Vieira & Guilherme H. Travassos, "A Survey on Model-based Testing Approaches: A Systematic Review", in WEASELTech'07, November 5, 2007, Atlanta Georgia, USA, ACM, ISBN 978-1-59593-880-0/07, June 2007.

[3] Renee C. Bryce, Ajitha Rajan & Mats P.E. Heimdahl, "Interaction Testing in Model-Based Development: Effect on Model-Coverage", in 13th Asia Pacific Software Engineering Conference (APSEC'06), ISBM-0-7695-2685-3/06, Aug 2007.

[4] Usman Farooq, Chiou Peng Lam & Huaizhong Li, "Towards Automated Test Sequence Generation", in Proceedings of 19th Australian Conference on Software Engineering ASWEC 2008 (pp. 441-450). Australia: Dec 2008.

[5] Robert M. Herons, "Oracles for Distributed Testing", in School of Information Systems, Computing, and Mathematics, Brunel University, Uxbridge, Middlesex, UB8 3PH, UK, 2010.

[6] Suresh Thummalapenta, Saurabh Sinha, Debdoot Mukherjee & Satish Chandra, "Automating Test Automation", in Publication of IBM T.J. Watson Research Center, Sep 2011.
[7] Sinaga, A., Zhou, Z., Susilo, W., Zhao, L. & Cai, K. 2009, "Improving software testing cost-effectiveness through dynamic partitioning", in B. Choi (eds), Proceedings of the 9th International Conference on Quality Software, IEEE, Los Alamitos, USA, pp. 249-258.
[8] Andrea Arcuri, Muhammad Zohaib Iqbal and Lionel Briand, "Random Testing: Theoretical Results and Practical Implications", in International Symposium on Software Testing and Analysis (ISSTA), ACM, 2010.
[9] T.Y. Chen, G. Eddy, R. Merkel and P.K. Wong, "Adaptive Random Testing Through Dynamic Partitioning", in Proceedings of the Fourth International Conference on Quality Software (QSIC'04), IEEE, doi:0-7695-2207-6/04, 2010
[10] Kulvinder Singh, Rakesh Kumar and Iqbal Kaur, " Effective Test Case Generation Using Anti Random Software Testing", in International Journal of Engineering Science and Technology Vol. 2(11), 2010, 6016-6021
[11] Marcel Böhme, "Software Regression as Change of Input Partitioning", in ICSE Doctoral Symposium , IEEE, Zurich, Switzerland , doi:978-1-4673-1067-3/12, 2012
[12] Chengying Mao, "Adaptive Random Testing Based on Two-Point Partitioning", in International Journal of Informatica, Volume 36, 2012
[13] Rakesh Kumar, Surjeet Singh, Girdhar Gopal, "Automatic Test Suit generation with Genetic Algorithm", in IJETCAS, ISSN (Online): 2279-0055, 2013
[14] Renee C Bryce, Sreedevi Sampath & Atif M Memon, "Developing a Single Model and Test Prioritization Strategies for Event-Driven Software", in IEEE Transactions on Software Engineering, Vol. 37, No. 1, Jan 2011.
[15] Soma Sekhara Babu Lam, M L Hari Prasad Raju, Uday Kiran M & Swaraj Ch, "Automated Generation of Independent Paths and Test Suite Optimization Using Artificial Bee Colony", in International Conference on Communication Technology and System Design, Published by Elsevier Ltd, ISSN 1877-7058, 2012.
[16] Premal B. Nirpal & K. V. Kale, "Comparison of Software Test Data for Automatic Path Coverage Using Genetic Algorithm", in International Journal of Computer Science & Engineering Technology (IJCSET), ISSN : 2229-3345, Vol. 1 No. 1, Sep 2012.
[17] A.V.K. Shanthi & G. MohanKumar, "A Novel Approach for Automated Test Path Generation using TABU Search Algorithm", in International Journal of Computer Applications, ISSN 0975 – 888,Volume 48– No.13, June 2012.
[18] Rupinder Singh & Vinay Arora, "Literature Analysis on Model based Slicing", in International Journal of Computer Applications, ISSN 0975 – 8887, Volume 70– No.16, May 2013.
[19] Saswat Anand, Edmund Burke et. al., "An Orchestrated Survey on Automated Software Test Case Generation", in Journal of Systems and Software, Feb 2013.
[20] Junpeng Lv, Hai Hu, Kai-Yuan Cai, and Tsong Yueh Chen, "Adaptive and Random Partition Software Testing", in IEEE Transaction of Systems , Man and Cybernetics:Systems, ISSN 2168-2216 ,doi: 10.1109/TSMC.2014.2318019, 2014.
[21] Ms. Ranjana Dalwani1 & Prof. Makrand Samvatsar, "Generic Adaptive Partition Testing Using Design Models for Software Quality Assurance", in IJARCCE ISSN (Online) 2278-1021 ISSN (Print) 2319 5940, Vol. 5, Issue 7, July 2016.