ISSN 1512-1232

ADVANCED MODELING BASED TEST GENERATION ARCHITECTURE (AMBTGA) APPROACH

Er.Amit Kanungo, Prof.Amit Sinhal

[1]M.TECH (IT), Technocrat Institute of TechnologyBhopal (M.P.)-462021, India; Amitkanungo11@gmail.com

[2]Department of Information Technology, Technocrats Institute of Technology,
Bhopal (M.P.)- 462021, India; amit_sinhal@rediffmail.com

Abstract

Pair wise testing is a measurement based testing technique which requires the combination of discrete input into reduced number of paired sets. Quality assertion of software is primarily done by means of testing, an activity that faces constraints of both time & resources. So combinatorial testing is a well accepted & dynamic approach for quality improvements because it provides affecting error detection at very low cost, hence an efficient strategy is require to reduce the number of test cases formed by above method. We also know that the problem of generating a minimum test set for pairwise testing is NP Complete.

In this paper we present a model based combinatorial approach design architecture. It builds a UML diagrams to visualize the test requirement specification & show that how a pair wise coverage problem is been represented by deterministic decision making through NP complete. This U M L N a v i g a t i o n a l d e s i g n architecture is used for reducing the test case generation complexity by solving the NP problem.

Keywords: *Pairwise Testing; combinatorial testing; UML modelling; NP coverage problem; navigational design architecture.*

I. INTRODUCTION

The main problem we face during testing is managing the large number of test cases we need to create and execute. The idea of pairwise testing is to generate a list of test sets that capture all possible pairs of parameter values from each parameter [1]. We proposed a new strategy how different combinations of specification of system could be tested efficiently using UML Model based combinatorial approach (AMBTGA).It refers to the processes and techniques for the automatic derivation of abstract test cases from abstract formal models, the generation of concrete tests from abstract tests, and the manual or automated execution of the resulting concrete test cases.

UML is a powerful modeling language used to represent the research problems visually [2]. A lot of literature is available on modeling problems by the use of UML, but limited research papers are reported in literature on applications of UML for the pairwise research orientation problems. By the use of UML, pairwise software testing problems can be solved and performance can be judged after modeling of the problem [3,4]. We present the proposed UML based design process architecture which is a five phase model maturity used for combinatorial test suite creation methodology. The schemas will assess the diagrams in a model for sufficient test related information. The intention, of the proposed exploration, is not to impose restrictions upon the modeling process; however, it is intended that our strategies will convey to a designer, how much information is sufficient to enable automatic generation of test cases so as to reduce the number of test suites.

An objective of this exploration is to present a designer with confirmation that the diagrams in a system model include sufficient information for automatically generating a

ISSN 1512-1232

suite of test cases [5]. There are two major aspects to the proposed study, which will be explored in five phases process. The first aspect relates to the testable information contained in a UML model [6]; while the second aspect relates to the development of a technique to generate a test suite from the acquired diagram data [7]. Initially, as part of the first phase, we must determine what information is been collected by requirement gathering phase then select a proper design model for test case extraction. Once a taxonomy of this generic information is established, we must determine which diagrams can provide the necessary information. In the second phase the navigational diagrams that offer this information are identified after that we will apply the test data on validation tool PICT to clarify the maturity of test cases [8]. The templates will form part of an application which produces a report on the amount and quality of test related information contained in a model's diagrams. Then we will explore which techniques might be suitable for the test data extraction process. Because some of the available information will come from different diagram types. Once all the information is been gathered we evaluate the result through injecting some fault in software & analyze our AMBTGA architecture for these fault detection Once the information is extracted we will develop a test suite format, that will enable the execution of test cases against the SUT.Finally, we must then evaluate our technique for effectiveness and efficiency, against other random test case generation methodologies.

II. BACKGROUND

(A) The Purpose of the Study

The proposed study will investigate and develop strategies and techniques to derive effective test cases from system-level, Model based combinatorial approach (AMBTGA). 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 pairwise testing[9,10,11,12].. The activities in AMBTGA approach diagram is numbered 1 to 5, are the main areas of focus. Prototype tools will be developed in future to d e m o n s t r a t e the techniques and strategies derived from the proposed investigation.

In summary, the study aims to:

1. Determine what information is necessary to test the integration of components in the process of system composition;
2. 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 effect aspects of activities 1 to 5 in figure;
3. develop a strategy that reports on the amount of testable information contained in a model;
4. develop a UML based technique for information extraction based on the information required for component integration, from single and multiple UML diagram types;
5. evaluate our overall strategy and techniques.

(B) Why UML & Combinatorial testing

UML based combinatorial approach is an innovative and high-value approach compared to more conventional functional testing approaches. The main expected benefits of AMBTGA may be summarized as follows:

(i) Contribution to the quality of functional requirements: Modeling for test generation is a powerful means for the detection of "holes" in the specification (undefined or ambiguous behavior).Independence from the test execution robot[13].

(ii) Contribution to test generation and testing coverage: Automated generation of test cases; Systematic coverage of functional behavior; Automated generation and maintenance of the requirement coverage matrix; Continuity of methodology (from requirements analysis to

ISSN 1512-1232

test generation)[14].

(iii) Contribution to test automation: Definition of action words (UML model operations) used in different scripts; Test script generation; Generation of skeleton code for a library of automation functions; Independence from the test execution robot[15].

(C) Combinatorial Or Pairwise Approach

Pairwise testing is a combinatorial specification based testing approach which is used to reduce the number of test cases.Emperical results shows that pairwise testing is a practical and effective for various types of software system. The problem of generating minimum test suites is NP complete [7]. So various strategies is been identified to solve these NP complete approach. We proposed here UML as a visualization of strategy describes here consists of identifying a canonical NP-complete problem on which GA s work well, and solving other NP problem indirectly by mapping them onto the canonical problem[16]. It significantly improves the performance of pairwise testing.

III. PROPOSED METHOD

A typical deployment of AMBTGA goes through five stages.

Setting Up Test Criteria. Usually an infinite number of possible tests could be generated from a model. The test analyst chooses Test Generation Criteria to select the highest priority tests or to ensure good coverage of the system behaviors. One common kind of test generation criteria is based on structural model coverage, using well known test design strategy of pair-wise testing. Another useful kind of test generation criteria ensures that the generated test cases cover all the requirements, perhaps with more tests for requirements that have a higher level of risk so in this paper we are combining the pairwise approach with new architecture through UML Navigational approach.

Test Model Designing. The model, generally called the test model, represents the expected behavior of the System Under Test (SUT). Standard modeling languages, such as Unified Modeling Language (UML) are used to formalize the control points and observation points of the system, the expected dynamic behavior of the system[17].

Test Suite Creation. This is an automated process that generates the required number of high-level (abstract) test cases from the test model. Each generated abstract test case is typically a sequence of high-level SUT actions, with input parameters and expected output values for each action of the test repository is done by updating the test model, then automatically regenerating the test suites.

Perform Tests. Generated concrete tests are typically executed within a standard automated test execution environment, such as PICT (Pairwise Independent Combinatorial Testing) tool[8]. Alternatively, it is possible to execute tests manually – i.e. a tester runs each generated test on the SUT, records the test execution results, and compares them against the generated expected outputs. Either way, when the tests are executed on the SUT, we find that some tests pass and some tests fail. The failing tests indicate a discrepancy between the SUT and model, which need to be investigated to decide whether the failure is caused by a bug in the SUT.

Analysing The Result: Analyse the real system which is to be tested and accepted by user. The effectiveness of test cases can be evaluated using a fault injection technique called mutation analysis. Mutation testing is a process by which faults are injected into the system to verify the efficiency of the test cases. For this we are using pairwise approach whose problem domain is NP Complete so the solution must be in accordance.

ISSN 1512-1232

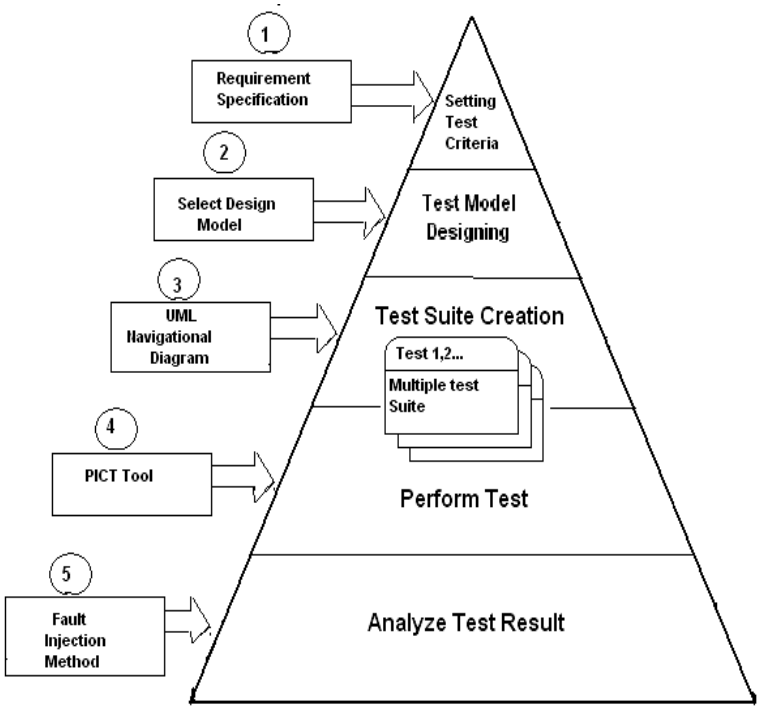

Fig:- A UML Model based combinatorial Architecture(UMBCA) for test suite preperation

Process Structure

The diagram set will be selected, based on a series of problem aspects that relate to the the type of testing being performed, referred to as the 'test objectives'. The series of problem are presented in figure below. In relation to the SUT, we have defined 'AMBTGA' as the test process objective. The diagram set will potentially include Use Case diagrams, Sequence diagrams, Activity diagrams, Class and Object diagrams and State diagrams. The assessment of a model's maturity will help designers to recognize the attributes that need to be included in each diagram type, and hence in a model, to improve the quality of the test cases that may be generated automatically from an pairtest approach [18]. In the past five or so years, we have seen a steady stream of research into software testing techniques and tools3, to find efficient and effective methods of revealing design and implementation faults as early as possible, in SDLCs that include UML modeling. Amid all this research, various UML diagrams have been targeted in association with the different testing phases of development.

ISSN 1512-1232

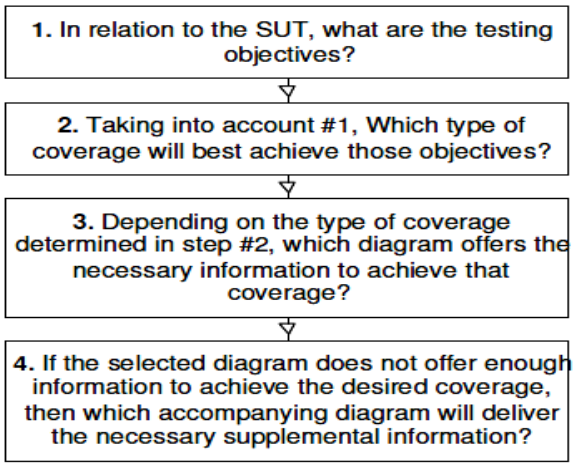

Fig:- A UML Based Process Overview

IV. EMPERICAL RESULT

Result Analysis:

For implementing test result we are using PICT as test tool & Fault Injection method to analyze the effectiveness & complete coverage of test suites verification. PICT was designed with three principles in mind:

(1) Speed of test generation,
(2) Ease of use, and
(3) Extensibility of the core engine.

Although the ability to create the smallest possible covering array was given less emphasis, the efficiency of PICT's core algorithm is comparable with other known test-generation strategies. The generation algorithm does not assume anything about the combinations to be covered. It operates on a list of combinations that is produced in the preparation phase.

This flexibility of the generation algorithm allows for adding interesting new features easily. The algorithm is also quite effective. It can compute test suites that are comparable in size to other tools that exist in the field, and it is fast enough for all practical purposes. (For instance, for 50 parameters with 20 value each (2050), PICT generates a pairwise test suite in under 20 seconds on an Intel Pentium 2.4GHz computer that is running Microsoft Windows XP SP2.

This activity is performed under the umbrella of constructive timing analysis, and its goal is to come up with a schedule for a system. This timing schedule must be assessed through a validation timing analysis activity, and the extended model of built-in contract 284 7 Assessing Quality-of-Service Contracts testing is specifically geared toward that. It is concerned with how the timing requirements that are specified in terms of a timing contract between two interacting components can be assessed and validated dynamically. In order to apply this extended model, the tested component needs to be augmented with an additional testing interface that provides timing notification services [19]. The tested component needs to be augmented with a test case generator, typically a random generator that applies a high volume of tests, and measures their timings. More effective test case generation techniques represent more sophisticated optimization strategies such as evolutionary algorithms [20]. They can be applied in the same way as random testing, but result in much more accurate timings.

Permanent testing or monitoring of QoS requirements in general or real time requirements in particular, beyond deployment, can be carried out through a supplementary quality assurance technique, built-in QoS testing. This provides a built-in testing framework that can be used to

ISSN 1512-1232

check code and data integrity, residual defects, deadlocks and timing, permanently during runtime of a component- based application.

Empirical Result

For Empirical Analysis of initial test factors of pairwise testing, consider the system in Figure. Pairtest System S has three input variables A, B, and C. Assume that set D, a set of test data values, has been selected for each of the input variables such that D(A) = {1, 2}; D(B) = {L, R}; and D(C) = {5, 6}.

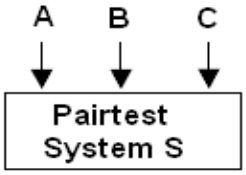

Fig A Pairtest system S With 3 Parameters

The total number of possible test cases is 2×2×2 = 8 test cases. The pairwise test set has a size of only 4 test cases and is shown in Table 1.

Test Suite	Parameter A	Parameter B	Parameter C
TS1	1	L	5
TS2	1	R	6
TS3	2	L	6
TS4	2	R	5

Table 1: Pairwise test suites for Pairtest System S
Test ID Parameter A , B & C

In this example, the pairwise test set of size 4 is a 50% reduction from the full combinatorial test set of size 8. You can see from the table that every pair of values is represented in at least one of the rows. If the number of variables and values per variable were to grow, the reduction in size of the test set would be more pronounced.

By applying the test suite approach on small development process, we identifies that as the test factors increases the coverage provided by our strategy of UML pairtest approach on random test.

Graph is shown as below:

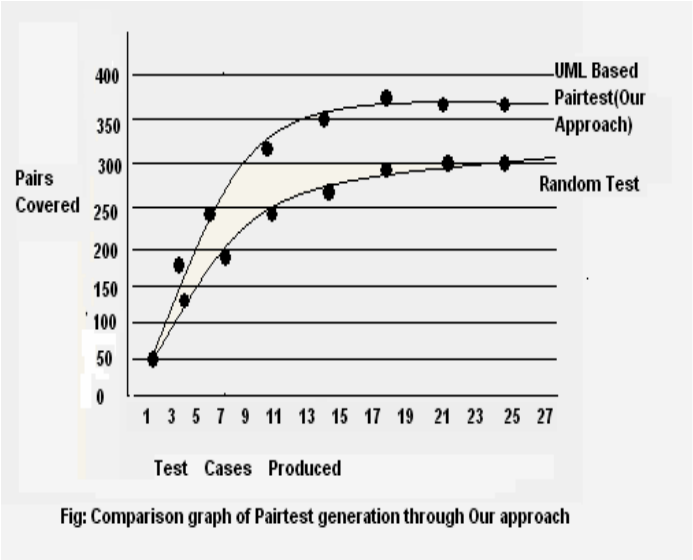

Fig: Comparison graph of Pairtest generation through Our approach

ISSN 1512-1232

So from the initial test factor result we also identifies that our approach is best suitable for real life practice & efficient in performance. We also concluded that by following the systematic AMBTGA approach it is been quite easy for testers to generate the test suite & provide the maximum coverage for fault detection.

V. EXPECTED BENEFITS

Expected benefits of AMBTGA over other approach UML based combinatorial approach is an innovative and high-value approach compared to more conventional functional testing approaches. The main expected benefits of AMBTGA may be summarized as follows:

Contribution to the quality of functional requirements:

(a) Modeling for test generation is a powerful means for the detection of "holes" in the specification (undefined or ambiguous behavior).

(b)Independence from the test execution robot.

Contribution to test generation and testing coverage:

(a) Automated generation of test cases;
(b) Systematic coverage of functional behavior;
(c) Automated generation and maintenance of the requirement coverage matrix;
(d) Continuity of methodology (from requirements analysis to test generation).

Contribution to test automation:

(a) Definition of action words (UML model operations) used in different scripts;
(b) Test script generation;
(c) Generation of skeleton code for a library of automation functions;
(d) Independence from the test execution robot.

VI. CONCLUSION

The idea of UML-based pairwise testing is to use an explicit abstract model of a SUT and its environment to automatically derive tests for the SUT: the behavior of the model of the SUT is interpreted as the intended behavior of the SUT. The technology of AMBTGA test case generation has matured to the point where large-scale deployments of this technology are becoming commonplace. The prerequisites for success, such as qualification of the test team, integrated tool chain availability and methods, are now identified, and a wide range of commercial and open-source tools are available.

Although AMBTGA will not solve all testing problems, it is an important and useful technique, which brings significant progress over the state of the practice for functional software testing effectiveness, and can increase productivity and improve functional coverage.

REFERENCES

[1] Pairwise Testing concept & strategy available from;
http://www.pairwise.org/tools.asp.
[2] B.Selic, and J.Rumbaugh, "UML for Modeling Complex Real Time Systems", Available Online Via www.rational.com/Products/Whitepapers/100230.Jsp.
[3] E.Holz, "Application of UML within the Scope of New Telecommunication Architectures", GROOM Workshop on UML, Mannheim:Physicaverlag, 1997.
[4] G.Booch, J.Rumbaugh and I.Jacobson, "The Unified Modeling Language User Guide",

ISSN 1512-1232

Addision Wesley, Reading, MA, 1999.
[5] Cle´mentine Nebut, Franck Fleurey, Yves Le Traon, "Automatic Test Generation: A Use Case Driven Approach", IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, VOL. 32, NO. 3, MARCH 2006.
[6] A.Abdurazik, J.Offutt, and A.Baldini. "A controlled experimental evaluation of test cases generated from UML diagrams". Technical report, George MasonUniversity, Department of Information and Software Engineering, 2004.
[7] Lei, Y and Tai, K.C., In-Parameter-Order: A Test Generating Strategy for Pairwise Testing, IEEE Tra ns. on Softwa r e Engineer ing, 2002, 28 (1), 1-3.
[8] J.Bach and P. Shroeder. Pairwise testing - a best practice that isn't. In P r oceedings of the 22nd Pa cific Northwest Softwa r e Qua lity Confer ence, pa ges 180– 196, 2004.
[9] Mandl, R., Orthogonal Latin Squares: An Application of Experimental Design toCompiler Testing, Comm. ACM, 1985, 28 (10), 1054-1058.
[10] Cohen, D.M., Dalal, S.R., Fredman, M.L., and Patton,G.C., The AETG system: An Approach to Testing Based on Combinatorial Design, IEEE Tra nsa ction on Softwar e Engineer ing, 1997, 23 (7), 437-443.
[11] Colbourn, C. and Dinitz, J. (Ed.) The CRC Handbook of Combinatorial Design, CRC Press, 1996.
[12] A.Andrews, R.France, S.Ghosh, and G.Craig. "Test adequacy criteria for UML design model. Software Test Verification and Reliability", 13:97–127, 2003.
[13] Stefania Gnesi, Diego Latella, and Mieke Massink. "Formal test- case generation for UML state charts". In ICECCS '04: Proceedings of the Ninth IEEE International Conference on EngineeringComplex Computer Systems Navigating Complexity in the e-Engineering Age, pages 75–84, Washington, DC, USA, 2004. IEEE Computer Society.
[14] J.Hartmann, C.Imoberdorf, and M.Meisinger. "UML-based integration testing". In ISSTA '00: Proceedings of the 2000. ACM SIGSOFT international symposium on Software testing and analysis, pages 60–70, New York, NY, USA, 2000.ACM.
[15] S.Helke, T.Neustupny, and T.Santen, "Automating Test Case Generation from Z Specifications with Isabelle," ZUM '97: The Z Formal Specification Notation, LNCS 1212, pp. 52-71. J.P. Bowen, M.G. Hinchey and D. Till, eds. Springer-Verlag, 1997.
[16] Davis, Lawrence (1985). Job Shop Scheduling with Genetic Algorithms, Proc. Int'l Conference on Genetic Algorithms a nd their Applica tions.
[17] S.R.Dalal, A.Jain, N.Karunanithi, J.M.Leaton, C.M.Lott, G.C.Patton "Model-Based Testing in Practice" To a ppea r in Proceedings of ICSE'99, May 1999 (ACM Press).
[18] P.E.Ammann and A.J.O_utt. "Using formal methods to derive test frames in category partition testing". In Ninth Annua l Conference on Computer Assurance (COMPASS'94), Ga ithersburg MD, pa ges 69–80,1994.
[19] P.E.Ammann and P.E.Black. "A specification-based coverage metric to evaluate test sets". In Proceedings of Four th IEEE Inter na tiona l High-Assurance Systems Engineer ing Symposium (HASE 99), pa ges 239–248. IEEE Computer Society, November 1999. Also NIST IR 6403.
[20] P. E. Ammann, P. E. Black, and W. Majurski." Using model checking to generate tests from specifications". In P r oceedings of the Second IEEE Inter na tiona l Confer ence on For ma l Engineering Methods (ICFEM'98), pages 46–54. IEEE Computer Society, Dec. 1998.

Article received: 2012-08-21