International Journal of Trend in Research and Development, Volume 3(6), ISSN: 2394-9333
www.ijtrd.com

A Survey on Automatic Test Case Generation

[1]S.Kannan and [2]T.Pushparaj
[1]Associate Professor, [2]Assistant Professor
[1]Department of Computer Applications, Madurai Kamaraj University, Madurai, Tamil Nadu, India
[2]Department of Computer Applications, PSNA College of Engineering and Technology, Dindigul, Tamil Nadu, India

Abstract-- Software testing is important activity in Software Development Life Cycle. To cut down cost of manual testing and to increase reliability of it, researchers and practitioners have tried to automate it. One of the important activity in testing environment is automatic test case generation - description of a test, independent of the way a given software system is designed. This paper presents a survey on automatic test case generation techniques that are found in the current literature. Problems in usage of certain techniques are identified. Areas that needed future research are presented.

Keywords-- *UML, OMT, UMLTEST, DAS-BOOT, OCL, UMLSC, IOLTS, Heuristic algorithm, AETG*

I. INTRODUCTION

Software organizations spend considerable portion of their budget in testing related activities. A well tested software system will be validated by the customer before acceptance. The effectiveness of this verification and validation process depends upon the number of errors found and rectified before releasing the system. This in turn depends upon the quality of test cases generated.

Through the years a number of different methods have been proposed for generating test cases. A test case is a description of a test, independent of the way a given system is designed. Test cases can be mapped directly to, and derived from use cases. Test cases can also be derived from system requirements. One of the advantages of producing test cases from specifications and design is that they can be created earlier in the development life cycle and be ready for use before the programs are constructed. Additionally, when the test cases are generated early, Software Engineers can often find inconsistencies and ambiguities in the requirements specification and design documents. This will definitely bring down the cost of building the software systems as errors are eliminated early during the life cycle.

II. LITERATURE SURVEY

Several approaches have been proposed for test case generation, mainly random, path-oriented, goal-oriented and intelligent approaches. Random techniques determine test cases based on assumptions concerning fault distribution. Path-oriented techniques generally use control flow information to identify a set of paths to be covered and generate the appropriate test cases for these paths. These techniques can further be classified as static and dynamic. Static techniques are often based on symbolic execution, whereas dynamic techniques obtain the necessary data by executing the program under test. Goal-oriented techniques identify test cases covering a selected goal such as a statement or branch, irrespective of the path taken. Intelligent techniques of automated test case generation rely on complex computations to identify test cases.

Many researchers and practitioners have been working in generating optimal test cases based on the specifications, still 100% testing is an impossibility. Modeling languages are used to get the specification and generate test cases. Since UML (Unified Modeling Language) is the most widely used language, many researchers are using UML diagrams such as state-chart diagrams, use-case diagrams, sequence diagrams, etc to generate test cases and this has led to Model based test case generation.

UML can be visualized as four meta-model architecture with three logical sub packages: Foundation, Behavioral elements and Model management. UML provides capability to explore static and dynamic behavior and physical deployment of a system. The possibility of using UML for software testing was addressed by Clay E. Williams [2]. UML models are built extensively for Object Oriented software systems. Class diagram, state diagram and OMT (Object Modeling Technique) and unified process are used to test Object Oriented systems. Here, the author has provided a framework to show which diagrams can be used for which phase in the testing process. But he claims that this knowledge is not sufficient, since it lacks detail. So, modification is required for testers to create their own models. Another approach could be to build standardized library or to extend the four meta-model architecture.

Even though variety of approaches have been proposed, with the advent of modeling tools like Rational Rose, for a decade there has been constant research on generating test cases based on specifications and design models. For easy understanding, we have classified test case generation approaches mainly into two categories – Specification based test case generation and Model based test case generation.

In this paper survey has been done on the work carried out during the last decade, where most of the researchers have concentrated on Model based test case generation.

A. Specification Based Test Case Generation

Jeff Offutt et al [1, 6] have presented many papers in the field of software testing. In [6] they have proposed a novel technique that adapts predefined state based specification to generate test cases from UML State-Charts. Full predicate test, statement coverage and transition pair tests were the techniques discussed by the authors. UMLTEST – a test data generation tool was integrated with Rational Rose and a case study has been extensively discussed for Cruise Control System. UMLTEST generated 34 test cases eliminating redundancies and the Cruise Control System has 7 functions, 184 blocks and 174 decisions. 24 faults were manually injected into the system and tested. Empirical results showed that Full predicate test found all faults. Transition pair tests covered 18 and statement coverage detected 16. In this work, the author have focused on UML State-Charts to get the specifications. Atac (a data flow coverage testing tool for C) was used to measure coverage with test cases. They were able to cover 163 blocks i.e. 89% and 155 decisions i.e. 95%. They wanted to extend their work to compare with multiple programs and to address all aspects of UML.

Specification Based testing described by Marlon E Viera et al [8] also employs UML state chart diagrams as the basis and automatically generates test drivers and test script to validate the component under test. The authors have implemented their approach in a prototype called Design And Specification-Based Object-Oriented Testing (DAS-BOOT). The Tester indicates the java class to be tested and the corresponding state-chart specification for the class behavior, and defines the representation mapping by associating code to the specification. Tester then chooses test coverage criteria. DAS-BOOT automatically generates test drivers with embedded test oracles. Finally failures detected by test oracles as discrepancies between behavior and state-chart specification are presented to the user. Goal is to define improved specification based coverage criteria suitable for testing Object Oriented systems and to develop techniques for generating test drivers with little human interaction.

B. Model Based Test Case Generation

Constant research and challenge in Model Based approach led Jeff Offutt et al [3, 10] to extend their work from system level analysis using state-chart diagrams to integration level analysis using collaboration diagram. The authors defined criteria for both static and dynamic testing of specification level and instance level collaboration diagrams. An algorithm has been designed to ensure that tests satisfy the formal testing criteria and help the tester to trace the paths. For experimental evaluation the authors have modeled standard cell phone software, which included 6 Class diagrams, 5 State-chart diagrams and 6 Sequence diagrams with 37 alternatives. 81 test cases were generated for State-chart diagrams and 43 for Sequence diagram. A message path coverage criterion is used to generate test cases from sequence diagram where as full predicate coverage has been used for State-charts. 49 faults (31 unit level and 18 integration level) have been inserted for verifying the effectiveness. At unit level, state charts were better compared to sequence but in integration level it was vice-versa. To generate even stronger and reliable test cases they planned to integrate UML specification with OCL (Object Constraint Language).

Many new tools have been build to work with existing tools to generate test cases. Tools such as TGV and UMLAUT were integrated to derive test cases and were evaluated through case study on classical electronic purse system by Lydie du Bousquet et al [4]. TGV, a conformance test generator generates test cases based on specifications and purposes. UMLAUT is a tool dedicated to manipulate UML models. Here, 50 test purposes were expressed with which 2100 test suites were derived. In order to transform test suites to java program, the authors have developed a translator "aut2Java". After performing both manual and automated testing, they compared the result, which showed that automation takes less effort and generates more test suites with minimum errors when compared to manual testing.

An architecture for automated test generation using UML profile and Test directives that consists of Test purposes, Test constraints, and coverage criteria was proposed by Alessandra Cavarra et al [5]. AGEDIS Modeling Language(AML) (named after AGEDIS project) is a subset of UML. Apart from test directives, some expressions and wildcards were presented to show direction and parameter constraints. Intermediate Format Language has been used to transfer the system model to state machine during compilation. Here, Object diagram and State diagram introduce Test directives whereas Class diagram provides information about data-types and operation. AGEDIS Test generation tool combines the principle of TGV and Gotcha which allows user to select test cases according to budget. This has some restrictions as software testing is difficult due to complex design and interactions are harder to measure. Since this project was not based on Object Oriented principles, room for improvement was left to develop automated test generation based on the work.

A framework for model level testing of behavioral UML models was proposed by Andras Toth et al [7]. Planner algorithm has been used for automatic generation of test cases. The input to this frame work is an UML State-Chart exported to tool independent XML(eXtensible Meta Language) format by UML tool. Transformation program (written in java) generates a text file, which can handle a rich subset of UML State-chart formalism by generating automatically a planner model. A planner meta-model was constructed to provide a high level representation of mathematical definition, in-order to keep their methodology open for other planner tools. As a result of this project UML designs can be tested and design flaws can be detected in modeling phase of development process prior to any implementation activities saving a considerable amount of cost and effort.

Matthias Riebish et al [13] presented a procedure for iterative software development process in generating test cases. Scenarios and Use cases fed as input for requirement engineering provided the basis for test case generation. The usage model created from State diagrams served as input for automated statistical testing. The method is supported by the tool "UsageTester" that exchanges model data with other development tools via XML. The authors have applied their approach in an industrial project with insurance domain. From this experience, the authors have established that obtaining test cases systematically can help in documentation of software's usage and interactive behavior. This procedure helps evolution of software development process not only in terms of functionality but also in terms of quality.

An approach that focuses on developing effective techniques and tools for test case generation and coupling them with suitable execution tools for unit, integration and system testing was proposed by Jean Hartmann et al [9]. The authors have integrated their tool with UML to automatically generate black box conformance test early in the development life cycle. For unit and integration testing the authors have derived tests from State-chart diagrams and Sequence diagrams. For system level they have used use-case and Activity diagrams. From these diagrams test cases were derived using Test Generation tool and executed using JUnit or system test tool. User Interface testing tools used are Rational Rose 2003 and Compuserve test partner 3.0. Main aim is to reduce testing cost and effort. The authors have intended to improve performance of test case generation, by proposing precise measurement technique for data coverage.

An approach that allows software testers as well as developers to generate effective test cases based on UML use case has been presented by Ahmed M. Salem et al [11]. The authors have demonstrated their work by performing case studies on Procurement/payment system. Use cases have been documented with pre-condition, post condition, basic and alternate flows. With the help of this test cases have generated in the form of matrix. This approach offers good traceability to original requirements, to test and verify requirements and to discover inconsistency in requirements. Further improvement has been suggested to develop a model for non-functional requirements. A theoretical foundation for model based formal

test case derivation using formal conformance testing for UMLSC (UML State-Chart) has been presented by Stefania Gresi et al [12]. In this approach, authors have used LTSs (Labeled Transition States) labeled over I/O pairs where, a generic transition models a step of associated State-charts. Here both specification and implementation are modeled as IOLTS (Input/Output Labeled Transition States). Test case generation algorithm has been presented which is a mix of process algebra and a simplified version of lambda calculus. Conformance relation and its test case generation algorithm are based on an operational semantics for UMLSC.

A coherent approach that integrates Sequence and State Diagrams for deriving a reasonably complete reference model usable in Industrial contexts, which will then be used for automatically deriving the test cases, has been proposed by Bertolino A. et al [14]. The authors derived test cases based on UML specifications both in component and object based contexts. The goal of this paper was to produce a complete model from sequence and state diagrams and to make Use Interaction Test (UIT) method to work with this model. This method guarantees that all available sequences are covered, but it cannot provide any coverage measure over the implementation system. This approach meets some important requirements imposed by the Industry. Advantage of this approach is that Incompleteness in the model need not be modified leading to low testing effort and generation of accurate test cases that are as much informative as possible.

C. Other Approaches

a. Path Oriented Test case generation

A formal framework using the well-known binary search strategy in path-oriented test case generation has been presented by Sami Beydeda and Volker Gruhn [16]. The proposed algorithm can be classified as dynamic path oriented approach. The authors have addressed the problem of dynamic test case generation approaches and provided a solution with binary search based test case generation algorithm. The path to be covered is considered step by step and test cases are then searched to fulfill them. In this paper the binary search used to determine test cases, requires certain assumptions but allows efficient test case generation. Advantages of this method are that it does not require any parameter calibration and does not need any optimizing techniques to generate test data.

III. INTELLIGENT TECHNIQUE

A new method to testing that uses combinatorial designs to generate efficient test sets was proposed by David M. Cohen et al [15]. The authors have implemented their method in AETG system which is used in varied applications. Here, the tester first identifies parameters that define the space of possible test scenarios. Then the tester uses combinatorial designs to create a test plan that covers all pair-wise, triple or n-way combinations of test parameters. Heuristic algorithm has been developed to generate pair wise testing. Telephone Switch's ability to place phone calls was taken as a problem for testing. This had 4 parameters with 3 values leading to 81 different scenarios. 9 Pair wise test cases were generated for enough coverage. The number of tests required and cost of adding additional parameter by the AETG method grows logarithmically with the number of test parameters. With varied experiments, AETG system showed better coverage that randomly generated test cases. Empirical results show that pair wise testing is practical and efficient for various types of software system.

IV. DISCUSSION

To sum up, there are many techniques available for generating test cases to satisfy test coverage criteria. A random test case generator may create many test data; but might fail to find test case to satisfy requirements. A path oriented approach identifies path for which test case has to be generated, however the path might be infeasible, the test data generator might fail to find an input that will traverse the path. An intelligent approach generates test case quickly but is quite complex. Comparing with these techniques, Model based testing is a valuable one, since it creates flexible, useful test automation from practically first day of development. Models are simple to modify, generate innumerable test sequences and allow the testers to get more testing accomplished in shorter time. Even though varied test case generation approaches are available, model based testing approach has attracted many researchers and still research is being carried out to optimize the generation of test cases with minimum human effort.

References

[1] Jeff Offutt, Aynur Abdurazik, October 1999, "Generating Tests from UML specifications", Second International Conference on the Unified Modeling Language (UML99), pp. 416-429, Fort Collins, CO.

[2] Clay E. Williams, November 1999, "Software testing and the UML", International Symposium on Software Reliability Engineering (ISSRE'99), Boca, Raton.

[3] Jeff Offutt, Aynur Abdurazik, October 2000, "Using UML Collaboration diagrams for static checking and test generation", Third International Conference on UML, York, UK.

[4] Lydie du Bousquet, Hugures Martin, Jean Marc Jezequel, 2001, "Conformance Testing from UML Specification Experience Report".

[5] Alessandra Cavarra, Charles chrichton, Jim Davies, Alan Hartman, Thierry Jeron, Laurent Mounier, September 2000 , "Using UML for Automatic Test Generation", Oxford University Computing Laboratory, Tools and Algorithms for the Construction and Analysis of Systems, (TACAS'2000).

[6] Jeff Offutt, Shaoying Liu, Aynur Abdurazik, Paul Ammann, March 2003, "Generating Test data from State based Specifications", The Journal of Software Testing, Verification and Reliability, 13(1):25-53.

[7] Andras Toth, Daniel Varro, Andras Pataricca, 2003, "Model Level Automatic Test Generation for UML State-Charts", Sixth IEEE workshop on Design and Diagnostics of Electronic Circuits and System, (DDECS 2003).

[8] Marlon E. Vieira, Marcio S. Dias, Debra J. Richardson, "Object-Oriented Specification-Based Testing Using UML Statechart Diagrams", University of California.

[9] Jean Hartmann, Marlon Vieira, Herb Foster, Axel Ruder, "UML based Test generation and Execution". www.gm.fh_koelin.de/~winter/tav/html/tav21/TAV21P6 Vieira.pdf

[10] Jeff Offutt, Aynur Abdurazik, Andrea Baldini, 2004, "A Controlled experiment evaluation of Test cases generated for UML diagram".www.isse.gmu.edu/techrep/2004/04-03.pdf

[11] Ahmed M. Salem, Lalitha Subramaniam, 2004, "Utilizing UML use cases for testing requirements", International Conference on Software Engineering research and practice, (SERP 2004).

[12] Stefania Gresi, Diego Latella, Mieke Massink, 2004, "Formal Test Case Generation for UML State-Charts", Ninth IEEE International Conference on Engineering Complex computer system Navigating complexity in e-Engineering Age.

[13] Matthias Riebish, Ilka Philippow, Marco Gotze, "UML Based Statistical Test Case Generation".

[14] Bertolino A., Marchetti E., 2004, "Introducing a reasonably complete and coherent approach for model based testing", Tools and Algorithms for the Construction and Analysis of Systems, (TACAS'2004).

[15] Cohen, D. M., Dalal, S. R., Fredman, M. L., Patton, G.C., 1997, "The AETG Design: an approach to testing based on Combinatorial design", IEEE trans on Software Engineering, vol. 23, no. 7, pp. 437-444.

[16] Sami Baydeda, Volker Gruhn, 2003, "BINTEST – binary search-based test case generation", In Computer Software and Applications Conference (COMPSAC), IEEE Computer Society Press, 2003.

AUTHORS PROFILE

Prof. T.Pushparaj M.C.A., M.Phil., M.B.A., (Ph.D)., is working in PSNACET. He has rich experience in handling System Software, DBMS, Computer Organization, Software Engineering, Software Project Management. His hobbies are playing cricket, reading books and listening melodies. He is a member in ISTE, IAENG, IACSIT. He has teaching experience of 8 years from a reputed esteemed Institution. He is currently pursuing Ph.D in M.K.U in the area of Software Testing Optimized Testcase generation.