TESTING AND VERIFICATION OF SOFTWARE MODEL THROUGH FORMAL SEMANTICS: A SYSTEMATIC REVIEW

Deepak Arora[1], Bramah Hazela[2]

[1]*Professor, Computer Science & Engg, Amity University, Lucknow*
[2]*Assistant Professor, Computer Science & Engg, Amity University, Lucknow*

Abstract

UML is a standard language used in business modeling for specifying, visualizing and constructing artifact for software and non software systems. It provides the capability to explore the static structure as well as dynamic behaviour of any large and complex software system. It consists of different software design patterns, templates and frameworks with unique diagrams to represent different aspect of software design during its development phase. Model based verification has been a key area to be explored to establish the model consistency and validation formalization. Through massive survey it is found that still the literature is lacking the well formed rules and semantics for UML model verification at the early stages of any software development. This research work emphasises the development of novel techniques for the verification and validation of different UML models. It also focus on automated test case generations using formal semantics based on different pre-established mathematical theories related to graphs. Testing of any software can be broadly classified into three parts: test case generation, test execution and evaluation. Various tools and techniques have already been proposed by many researchers for automation of model verification specifically for object oriented software designs. In this paper authors have summarized and analyzed different approaches and methodologies related to automated verification of UML models and formalization of rules and semantics in order to automate the test case generation and its evaluation.

Keywords: *UML, Formalization, Automatic Test Case Generation, Model Based Verification and Testing.*

----------***----------

INTRODUCTION

Software modelling can be best described by the use of formal semantics. Selection of a proper model is the basis of modelling. There are various methods for modelling a software system that includes Unified modelling language (UML), Graph Transformation system (GTS), Abstract State Machine (ASM), State Diagrams. Unified modelling language provides broader range of language construct specifications to be constructed. A key areas in successfully using UML2 is understanding the semantics of the augmented language construct. For example, what is the meaning of a class diagram in terms of a component diagram, or what is the meaning of a state machine in terms of an activity diagram. Is there any approach for transformation of UML behavioural diagrams that will help in software testing and verification? These are not easy questions to answer and involve understanding the semantics of each individual construct.

All phases of software development starting from requirements analysis, design, implementation is discuss and maintain by UML diagram. The main goal is to model the software system before you build it. UML 2 specification defines two major kinds of UML diagram: structural diagrams and behavioural diagrams. Structure diagrams show the static structure of the system and how they are related to each other. The elements in a structure diagram represent the meaningful concepts of a software system, and may include abstract, real world and implementation concepts. Behavioural diagrams show the dynamic aspect of the objects in a system, which can be described as a series of changes to the system over time.

UML models are important source of information for design of test cases and verification. Model-based test case generation can be planned at an early stage of the software development life cycle, allowing to carry out coding and testing in parallel. Software testing is an important method to assure software quality. Traditional testing method based on handwork is low efficiency and cause the increase of test cost and time. With the development of testing technology, it advances high demand for to automate software system. The automated test cases generation is viewed as a guarantee to carry out effective and maintainable software testing. Model-based test case generation and verification technique becomes an obvious choice in software industries and is the focus of this work.

This survey aims at summarizing the automated testing and verification by covering the research queries below. The research queries aims at finding the efficient procedures for automated testing and verification of software model in practice. The queries are:

RQ1- What are the various approaches for transformation of UML diagrams to other graphical structure?
RQ2-Which is the most widely used technique?
RQ3-What are the broad areas covered by these transformation techniques?
RQ4- What is the need of formalization of UML diagrams?
RQ5- What are the benefits of using various techniques?

2. SOURCES OF INFORMATION

This paper presents a systematic review of the work done in the field of automatic generation of test case particularly related to UML based automated test case generation and verification of software system. In order to gain a broader perspective, various papers and journals were searched. The following six databases were covered:
I. ACM Digital Library (www.portal.acm.org).
II. IEEE Xplore (www.ieeexplore.ieee.org).
III. Springer LNCS (www.springer.com/lncs)
IV. Science Direct (www.sciencedirect.com).
V. Journal of Object technology (www.jot.fm)
VI. Google Scholar (www.googlescholar.com)

3. SEARCH CRITERIA

The initial search criteria was kept broad in order to include the articles with different uses of terminology. The key words used were <transformation> and (UML or <UML Diagrams>) and <finite state machine> and <generation of grammar> and <software testing and verification), and the database fields of title and abstract were searched. The start year was set to 1990 to ensure that most relevant research within the field would be included, and the last date for inclusion is publications within 2014.

The ultimate goal of software testing is to help designers, developers, and managers construct systems with high quality. Thus, research and development on testing aim at efficiently performing effective testing to find more errors in requirement, design and implementation. Progress toward this destination requires fundamental research, and the creation, refinement, extension, and popularization of better methods. The evolution of definition and targets of software testing has directed the research on testing and verification techniques.

4. DATA COLLECTION

The list of journals and conference proceedings with no. of paper referred is given below,
I. IEEE Transactions of Software Engineering: 14
II. Journal of system and software: 02
III. Software testing verification and reliability: 03
IV. ACM computing surveys: 05
V. IBM system of journals: 01
VI. IEEE Computer and application software: 05
VII. Computer and Information sciences: 04
VIII. Journal of object technology: 01
IX. International workshop of automation on software test: 02
X. IEEE conference on software maintenance: 06
XI. Proceedings of International conference on UML: 06
XII. Software engineering & Advanced applications: 02
XIII. Software reliability engineering: 02
XIV. ACM SIGSOFT software engineering: 07
XV. Computer science and Information technology: 03
XVI. Others: 06

The Unified Modeling Language (UML) [47] is a visual modeling language that comprises fourteen types of diagram representations to show structural and behavioural characteristics of any system. Now a days, there are many studies that are focused on test cases generation from UML specification and can be found in [20, 22, 25 26].

5. RESULTS

The following section reflects the results related to the research question.

5.1 RQ1- What are the Various Approaches for Transformation of UML Diagrams to other Graphical Structure?

There are various approaches for transformation of UML diagrams to FSM (Finite state machine), Abstract state machine (ASM) and other graphical structure which would be helpful for further verification of software system.

5.2 RQ2: Which is the Most Widely used Technique?

The most widely used techniques involve Model based software testing. One of the oldest approaches for model based testing is by using Use Case, class and State diagram. In these approaches, the models are transformed into its equivalent usage models to describe behaviour and usage of software system. Kansomkeat [41] proposed an approach using only state chart diagrams. The main advantage of this approach was the capability of automation.

5.3 RQ3- What are the Broad Areas Covered by these Transformation Techniques?

The broad areas covered by these techniques includes
Compiler construction tool, real time embedded systems, artificial intelligence planning, spread sheets, OO systems, SOA interacting services.

5.4 RQ4- What is the need of Formalization of UML Diagrams?

Formalization of UML has become a prominent domain of research for the last few years. In this research query we will discuss a few works done in this domain related to formalization of UML static and dynamic models.

In all research works, UML diagrams have been Formalized using other formal languages. A context free grammar can be generated for widely used UML diagrams that are used in the design phase of the software life cycle namely Class, Sequence and State Chart diagram. In [69], we propose a Context Free Grammar for the analysis phase to establish traceability of requirements and consistency verification of UML Use case, Activity and Class Diagram. A UML compiler has been proposed in [68] that is a framework for syntactic and semantic verification of UML diagram. This

work proposes a grammar for Class and Sequence diagram. Here, we have taken into consideration the State Chart diagram also because the State Chart diagram depicts the state change of an object at runtime. We can also formalized other behaviour UML diagrams.

5.5 RQ5- What are the Benefits of using Various Techniques?

Each technique has its own advantages and disadvantages. A UML State chart [39] covers various test criteria such as transition coverage, full predicate coverage, transition pair coverage and complete sequence coverage. It also helps in performing class level testing. Activity Diagrams on the other hand can represent both conditional and parallel activities. A fork construct is used for concurrent activities in activity diagram. Sequence diagrams [44] describe sequence of actions that generate in a system over time. It captures invocation of methods from each object and order in which it occurs. Collaboration Diagrams [67] covers the dynamic aspect of testing better than any other UML model. Therefore, it can easily represent dynamic behaviour of the system along with good graphical representation of system scope requirements.

6. CONCLUSIONS

Unified Modeling Language (UML) has now become a de facto standard in the field of software testing and verification. New formalization techniques for the generation of test case from these UML diagrams need to be explored.

The overall objective of the study was to gather sufficient data to understand the nature of the various testing techniques available. These techniques will be more effective with formalized UML diagram.

The number of techniques proposed for test case generation is very large. We need to better understand the difference between these techniques and explore new methods for further improvements.

FUTURE WORK

Hence, the next step in this field of research will involve surveying and finding new possibilities in this area. Further, possibility of automation in test case generation via UML diagrams will also be explored simultaneously. Other possibilities include the improvements in testing techniques from UML diagrams. Further possibility is to automate test case generation and to explore other methods of using formal methods in software testing and verification.

REFERENCES

[1]. Heumann J. (2001) Generating Test Cases from Use Cases, Rational Software, IBM,

[2]. Bird, D. L. Munoz, C. U. (1983) Automatic generation of random self-checking test cases, IBM Systems Journal

[3]. Tsai, W.T. Volovik, D. Keefe, T.F. Fayad, M.E. (1988) Automatic test case generation from relational algebra queries, Computer Software and Applications Conference.

[4]. Tsai, W.T. Volovik, D. Keefe, T.F. (1990) Automated test case generation for programs specified by relational algebra queries, Software Engineering, IEEE Transactions on software engineering

[5]. Wang, C.J. Liu, M. (1993) Automatic test case generation for Estelle. International Conference on formal engineering methods, Network Protocols

[6]. Ammann, P.E. Black, P.E. Majurski, W. (1998) Using model checking to generate tests from specifications, Second International Conference on Formal Engineering Methods

[7]. Memon, A.M. Pollack, M.E. Soffa, M.L (1999) Using a goal-driven approach to generate test cases for GUIs, International Conference on Software Engineering

[8]. Cunning, S.J. Rozenblit, J.W. (1999) Automatic test case generation from requirements specifications for real-time embedded systems, IEEE International Conference on Systems, Man, and Cybernetics, vol 4, pp345-378

[9]. Gutierez J., Escalona M.J. and Torres M.M. (2006) An Approach to Generate Test Cases from Use Cases, Proceedings of the 6th International Conference on Web Engineering, pp. 113-114.

[10]. Hui L. and Hee B.K.T. (2006) Automated Verification and Test Case Generation for Input Validation, International Workshop on Automation on Software Test (AST‟06), pp. 29-35.

[11]. Nebut C, Fleurey F, Traon Y.L. and Jezequel J.M. (2006) Automatic Test Generation: A Use Case Driven Approach, IEEE Transactions on Software Engineeering, Vol 32, No. 3, pp. 140-155.

[12]. Wee K.L., Siau C.K. and Yi S. (2004) Automated Generation of Test Programs from Closed Specifications of Classes and Test Cases, Proceedings of the 26th International Conference on Software Engineering (ICSE‟04)

[13]. Wei-Tek Tsai, Yinong Chen (2005) WSDL-Based Automatic Test Case Generation for Web Services Testing, IEEE International Workshop on Service-Oriented System Engineering

[14]. Bor-Yuan Tsai, (2002) An Automatic Test Case Generator Derived from State-Based Testing, IEEE Trans. on Software Engineering, vol 7, pp781-812

[15]. Chow, Tsun S. (1978, May) Testing Software Design Modelled by Finite-State Machines, IEEE Trans. on Software Engineering, Vol. SE-4, No. 3

[16]. Turner, C. D.; Robson D. J. (1993) The State based Testing of Object-Oriented Programs, Conference on Software Maintenance

[17]. Hoffman, D; Strooper, P., ClassBench (1996-2003) A methodology and framework for automated class testing, SVRC, University of Queensland; Software-Practice & Experience, Technical Report

[18]. Proceedings of the Fourth International Conference on Quality Software (QSIC‟04).

[19]. Frohlick P. and Link J. (2004) Automated Test cases Generation from Dynamic Models, in the Proceedings of the

European Conference on Object-Oriented Programming, Springer Verlag, LNCS 1850, pp. 472-491
[20]. A. Bertolino, F. Basanieri, (2000) A practical approach to UML-based derivation of integration tests, in: Proceedings of the Fourth International Software Quality Week Europe and International Internet Quality Week Europe (QWE), Brussels, Belgium.
[21]. Riebisch M., Philippow I, and Gotze M. (2003) UMLBased Statistical Test Case Generation, in the Proceeding of ECOOP 2003, Springer Verlag, LNCS 2591, pp. 394-411.
[22]. Hartmann J., Vieira M., Foster H., Ruder A. (2005) A UML-based Approach to System Testing, Journal of Innovations System Software Engineering, Vol. 1, PP. 12-24
[23]. Valdivino Santiago1, Ana Silvia Martins do Amaral1, N. L. Vijaykumar (2008) QSEE project
[24]. F. Basanieri, A. Bertolino, E. Marchetti (2002) The cow suit approach to planning and deriving test suites in UML projects Proceedings of the Fifth International conference on the UML, LNCS, 460, springer-Verlag GmbH, Dresden, Germany, pp. 383–397.
[25.] Offutt J., Abdurazik A. (1999) Generating tests from U.ML specifications. Proc. 2nd Int. Conf. UML, Lecture Notes in Computer Science, Fort Collins, TX, Springer-Verlag GmbH, vol. 1723, pp. 416– 429.
[26]. Offutt J., LIU S., Abdurazik A. (2003) „Generating test data from state-based specifications", Software, Testing, Verification, Reliability, Vol 13, pp. 25–53.
[27]. Kansomkeat S., Rivepiboon W (2003) "Automated-generating test case using UML statechart diagrams". Proc. SAICSIT 2003, ACM, pp. 296– 300.
[28]. Cavarra A., Crichton C., Davies J. (2004) "A method for the automatic generation of test suites from object models", Information and Software Technology, 46, (5), pp. 309–314.
[29]. Hartmann J., Imoberdorf C., Meisinger M. (2000) UML-based integration testing, ACM SIGSOFT Software Engineering Notes, Proceedings International Symposium, Software Testing and Analysis, vol. 25.
[30]. Kim Y.G., Hong H.S., Bae D.H.(1999), "Test cases generation from UML state diagram", Proc.
Software 146, (4), pp. 187–192.
[31]. Zhenyu Dai, Mei-Hwa Chen (2007) Automatic Test Case Generation for Multi-tier Web Applications, 9th IEEE International Workshop on Web Site Evolution.
[32]. Shengbo Chen, Huaikou Miao, Zhongsheng Qian (2007) Automatic Generating Test Cases for Testing Web Applications, International Conference on Computational Intelligence and Security Workshops.
[33]. P. Samuel R. Mall A.K. Bothra (2008) Automatic test case generation using unified modeling language (UML) state diagrams, The Institution of Engineering and Technology.
[34]. Yuan-Hsin Tung, Shian-Shyong Tseng, Tsung-Ju Lee, and Jui-Feng Weng (2010) A Novel Approach to Automatic Test Case Generation for Web Applications, 10th International Conference on Quality Software.
[35]. Cao Xizhen Qian Hongbing (2010) Research on test cases automatic generation technique based on AADL model, 3rd International Conference on Advanced Computer Theory and Engineering (ICACTE).
[36]. Shahzad, A. Raza, S. Azam, M.N. Bilal, K. Inam-ul-Haq Shamail, S (2009) Automated optimum test case generation using web navigation graphs International Conference on Emerging Technologies.
[37]. Shaoying Liu Nakajima, S. (2010) A Decompositional Approach to Automatic Test Case Generation Based on Formal Specifications, Fourth International Conference on Secure Software Integration and Reliability Improvement (SSIRI)
[38]. Baikuntha Narayan Biswal (2008) A Novel Approach for Scenario-Based Test Case Generation, International Conference on Information Technology, vol 43, PP 244-247
[39]. Philip Samuel Rajib Mall (2009) Slicing-Based Test Case Generation from UML Activity Diagrams, ACM
SIGSOFT Software Engineering Notes, Volume 34 Number 6, Page 1
[40]. Santosh Kumar Swain (2010) Test Case Generation from Behavioral UML Models, International Journal of Computer Applications, Vol 6, No.8.
[41]. Kansomkeat (2010) Generating Test Cases from UML Activity Diagrams using the Condition-Classification Tree Method, 2nd International Conference on Software Technology and Engineering (ICSTE)
[42]. Xi Wang, Liang Guo, Huaikou Miao (2008) An Approach to Transforming UML Model to FSM Model for Automatic Testing, International Conference on Computer Science and Software Engineering, IEEE, vol 34
[43]. Binder R. V., Testing Object-Oriented System Models, Patterns, and Tools, Addison-Wesley, NY, 1999
[44]. Emanuela G. Cartaxo, Francisco G. O. Neto and Patreıcia D. L. Machado (2007) Test Case Generation by means of UML Sequence Diagrams and Labeled Transition Systems,
[45]. Byoungju Choi, Hoijin Yoon, Jin-Ok Jeon (1999) A UML-based Test Model for Component Integration Test, Workshop on Software Architecture and Component, Japan, pp63-70
[46]. Zhang Mei, Liu Chao, Sun Chang-ai (2001) Automated Test Case Generation Based on UML Activity Diagram Model, Journal of Beijing University of Aeronautics and Astronautics(in Chinese), vol. 27 No. 4, pp 433-437
[47]. Grade Booch, James Rumbaugh, Ivar Jacobson (2001) The Unified Modeling Language User Guide, Addison-Wesley
[48]. Beizer (1995) Black-box Testing: Techniques for functional testing of software and systems, John Wiley & Sons, Inc, New York
[49]. Paul C. Jorgrnsen (1995) Software Testing: A Craftsman's Approach , CRC Press Inc
[50]. B. A. Kitchenham (2002) Preliminary guidelines for empirical research in software engineering,IEEE Test Selection from UML Statecharts, vol 28, pp 721-734
[51]. M. Sarma, R. Mall (2009) Automatic generation of test case specification for coverage of system state transition, Information on software Technology, vol 51, pp 418-432
[52]. Stefania Gnesi (2004) Formal Test Case generation for UML state charts, Proceedings of 9th International

Conference on engineering complex computer system navigating complexity, vol 34, pp 1050-4729
[53]. Shinpei ogata and Saeko Matsuura (2010) A Method of Automatic Integration Test Case Generation from UML-based Scenario WSEAS transactions on information science and applications, Issue 4, Volume 7,
[54]. Chen Minsong (2006) Automatic test case generation for UML activity diagram, AST, vol 78, pp456-478
[55]. X. Hou (2010) Integration testing system scenario generation based on UML, International conference on computer, mechatronics, control and electronic engineering, vol 72, pp 271-273
[56]. Li Liuying Qi Zhichang {1999) Test Selection from UML Statecharts, IEEE Test Selection from UML Statecharts, vol 99, pp 198-264
[57]. Andreas Heinecke, Tobias Bruckmann, Tobias Griebe, Volker Gruhn (2010) Generating Test Plans for Acceptance Tests from UML Activity Diagrams, 17th IEEE International Conference and Workshops on Engineering of Computer-Based Systems, vol 14, pp 425-654
[58]. Wang Linzhang, Yuan Jiesong, Yu Xiaofeng, Hu Jun, Li Xuandong and Zheng Guoliang (2004) Generating Test Cases from UML Activity Diagram based on Gray-Box Method, Proceedings of the 11th Asia-Pacific Software Engineering Conference (APSEC‟04), pp 1362-1530
[59]. Tsai, W.T. Volovik, D. Keefe, T.F. Fayad, M.E. (1988) Automatic test case generation from relational algebra queries, Computer Software and Applications Conference.
[60]. D. Gelperin and B. Hetzel (1988) The Growth of Software Testing, Communications of the ACM, Volume 31 Issue 6, pp. 687-695
[61]. Shireesh Asthana, Saurabh Tripathi, and Sandeep Kumar Singh (2010) Novel Approach to Generate Test Cases Using Class and Sequence Diagrams, Springer Verlag, CCIS 95, pp 155-167
[62]. Supaporn Kansomkeat, Phachayanee Thiket, Jeff Offutt (2010) Generating Test Cases from UML Activity Diagrams using the Condition Classification Tree Method, 2nd International Conference on Software Technology and Engineering(ICSTE), vol 45, pp 456-489
[63]. Briand L. and Labiche Y. (2002) A UML-Based Approach to System Testing, in the Journal of Software and Systems Modeling, Springer Verlag, Vol. 1, pp. 10-42
[64]. Matthew Kaplan, Tim Klinger, Amit M. Paradkar, Avik Sinha, Clay Williams, Cemal Yilmaz (2008) Less is More: A Minimalistic Approach to UML Model-Based Conformance Test Generation, International Conference on Software Testing, Verification, and Validation, IEEE, vol 69, pp 82-93
[65]. Baikuntha Narayan Biswal (2008) A Novel Approach for Scenario-Based Test Case Generation, International Conference on Information Technology, vol 43, PP 244-247
[66]. B. Beizer (1999) Software Testing Techniques, Second Edition, Van Nostrand Reinhold Company Limited, ISBN 0-442-20672-0
[67]. Abdurazik, A. Offutt, J. Using UML Collaboration Diagrams for Static Checking and Test Generation UML 2000 - The Unified Modeling Language. Advancing the Standard. Third International Conference, York, UK, October 2000, Proceedings, Springer, 2000, vol 1939, pp 383-3
[68]. J. Chanda , A. Kanjilal, S. Sengupta , “UML-Compiler: A Framework for Syntactic and Semantic Verification of UML Diagrams” Proceedings of ICDCIT 2010, Bhubhaneswar, India
[69]. Jayeeta Chanda, Ananya Kanjilal, Sabnam Sengupta, Swapan Bhattacharya, “Traceability of Requirements and Consistency Verification of UML Use case, Activity and Class Diagram: A Formal Approach”, Proceedings of IEEE ICM2CS New Delhi, Dec 14-15, 2009.