State-chart based UML diagrams for Test Case Generation

[1]Sudhir, [2]Sandeep Dalal

[2]Assistant Professor, [1]Research Scholar

[1,2]Maharshi Dayanand University, Rohtak, India-124001

Abstract: ***UML (Unified Modelling Language) based generation of test cases has gained much attention of researchers in recent past due to extreme suitability of UML diagrams to model the user requirements and its higher capability of modelling the system to generate optimal test sequences in software. UML possess variety of graphical notation representations like class diagram, activity diagram, state-chart diagrams, component diagram etc. which are efficiently capable of expressing and modelling the software system in a graphical form. Numerous researchers have worked in this direction to generate optimal test sequences using state-chart diagrams, which represents the behavioral aspect of the system. This paper basically summarizes and review the research work of various researchers to generate test cases using UML state chart diagrams.***

Index Terms – Software Testing, Test Sequence Generation, UML diagrams

---

I. INTRODUCTION

The features of an excellent test model comprise of three aspects or characteristics namely representation, simplification and pragmatic. These characteristics forms the basic structure of a model as shown in figure 1. However, these characteristics of the model are not sufficient for generating effective test cases and evaluation process. According to Binder, "A test model provide the complete, consistent and accurate requirement of the system" [1].

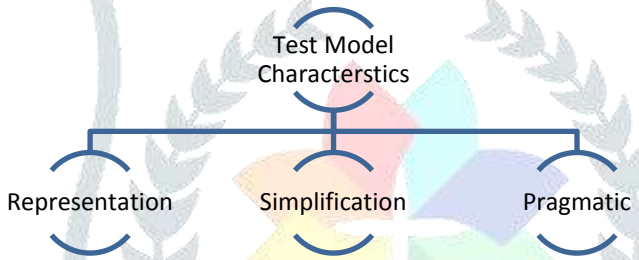

Figure 1. Characteristics of a test model

The main challenge before the testing community is to develop an abstract model up to the right level. The complete information should be available in the model to generate the effective test cases. Moreover, a large number of the test cases can be generated by using the technique of automated test case generation. But, quantity never takes guarantee that all generated test cases are accurate and efficient. Farooq suggested that "Quality of the generated test cases are equally important to quantity" [2].

II. UML DIAGRAMS IN SOFTWARE TESTING

UML is a standard language or notation for specifying, visualizing, constructing and documenting the artifacts of software system. UML mostly uses graphic notations to express software designs and requirements. Unified modeling language (UML) is a class of variety of models such as activity, collaboration, component, sequence, use case and deployment models. The types of UML diagrams namely structural and behavioral diagrams have been discussed in figure2. The test requirements are derived from these UML models as shown in figure 3. The following 4 depicts the suitability of UML diagrams according to different level of testing. Class diagrams have been using in unit testing as the focus of unit testing is to test the particular method or component. Collaboration and class diagrams are suitable for integration testing. Use case, state machine and activity diagrams are appropriate for system level testing.

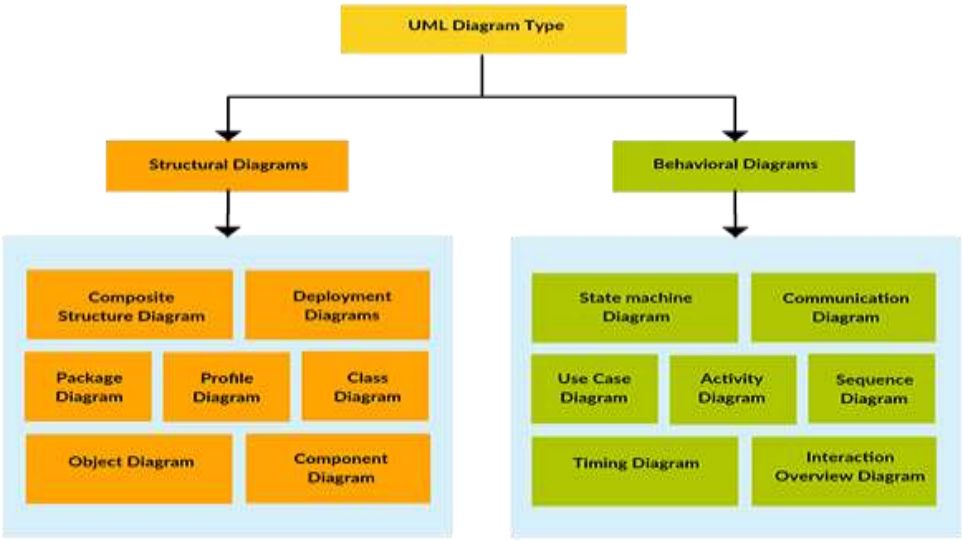

Figure 2. Types of UML Diagrams [17]

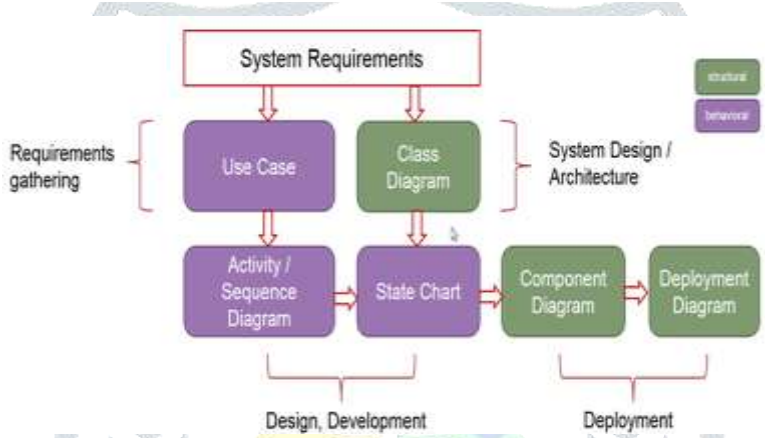

Figure 3. UML Diagrams for User Requirements [17]

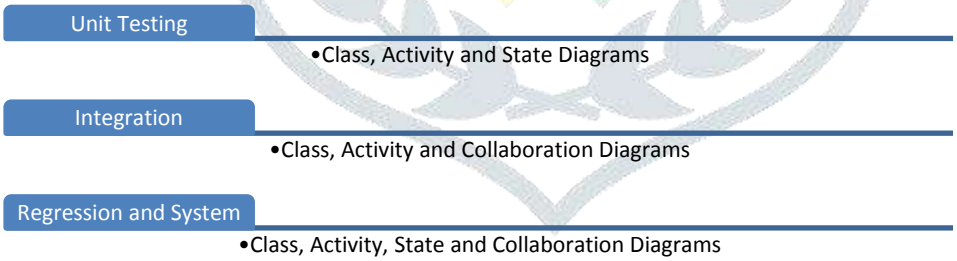

Figure 4. Usage of UML Diagrams at Various Testing Levels

III. Related Work

This section discusses about various test case generation techniques using state chart-based UML diagrams. State chart diagrams have been widely used by many researchers in the field of software testing for effective generation of test sequences by enhancing branch coverage, path or transition coverage, state coverage criteria etc. Researchers have deployed meta heuristic and nature inspired algorithms to these generated test sequences for test suite optimization i.e. selection, minimization and prioritization of test cases. Thus, it can be concluded from table 1 described below that UML (state chart) based test case generation techniques are gaining popularity and will prove a milestone for researchers in near future.

Table 1. Comparison of Research Work Based on UML usage for Test Case Generation

Authors	Proposed Work
Frohlich and Link [3]	Authors presented an approach for generating test sequences from the use case textual description. The proposed approach for generation of test sequences actually transforms the problem for test case design in a planning problem. Artificial intelligence planning approaches were used by the authors to find paths in the state machine to satisfy all the transition preconditions. Their approach basically needs certain kind of manual annotations to these UML diagrams. Here, authors have utilized "transition coverage" parameter for the state model.
Hartmann *et al.* [4]	Authors proposed an automatic methodology for test sequence generation depending upon the interactions of a user with system. These interactions were modelled using semi-automatic conversion of use cases textual description into activity diagrams. This approach comprises of marking the activity diagram with their test requirements before generating test cases.
Nebut *et. al.* [5].	This approach dealt with generating the test case scenarios using use case models. Authors proposed the approach that improved the use case model in a semi-automatic way with executable conventions (pre- and post-conditions). The executable conventions are then utilized to formulate a simulation model for generation of valid test sequences from use cases. Thereafter, the generated test scenarios replaced every use case in the test sequence. The generated test sequences are capable of achieving better statement coverage. The limitation of the approach is that it assumes a firm mapping of user requirements to software code.
Offutt and Abdurazik [6]	Authors presented an approach for test case generation using UML state chart diagrams. Authors employed change events for "Boolean class attributes" to generate test cases automatically. Authors successfully developed numerous valuable coverage criteria based on UML state-chart diagrams. The presented approach focused on class-level testing and was capable of achieving "transition coverage, full predicate coverage and transition-pair coverage". Every outbound transition of every source state initially generates a test case which makes transition valid and later on, those test cases were generated which make the transition invalid.
Kansomkeat *et. al.* [7]	Authors have proposed a unique method for using UML state chart diagrams for generating test sequences. Authors transformed the state chart diagram in a "flattened hierarchical structure" of states known as "Testing Flow Graph (TFG)". Thereafter, TFG is traversed right from the root node to every leaf node for test case generation. test sequences were generated from event sequences of TFG. The performance evaluation of their proposed approach was based on TFG used the parameters of state coverage and transition coverage. The experimental results concluded the worth of the approach in terms of better state and transition coverage.
Kim *et. al.* [8,9]	Authors devised a novel method to generate test cases for "class level testing" using UML state chart diagrams. Authors transformed state charts in to "Extended Finite State Machines (EFSMs)" to descend test cases. An EFSM was a normal representation form of the state-chart. Authors transformed EFSM into a flow graph representing the flow of data as well as control in the state-chart. At the end, authors used "conventional data flow analysis technique" for test sequence selection from the flow graph.
Hartmann *et. al.* [10]	Authors have developed an approach for integration testing of "distributed components" by using state-chart diagrams. They amplified the UML portrayal with some particular notations to set up a design-based environment for software testing. The basic assumption of the proposed approach was that the dynamic behavior of every component of systems was being modelled using UML state-charts. These distinct state-charts were then merged to frame a global behavioral model. The communication among components was defined using "CSP (Communicating Sequential Processes)" formalism. The relations among different components were then represented by annotating the state-chart diagrams. The obtained global FSM model which matches to the integrated system behavior is then utilized to generate the test sequences. "category partition method "is then used for automatic test generation.
Gnesi *et. al.* [11]	The process of verifying and validating a software system for fulfilling the specified end user requirements is known as conformance testing. A conformance relationship is based on the correctness parameter of the implementation regarding the formal specification Authors developed a novel mathematical methodology for automatic test case generation and conformance testing using

	UML state charts. Authors have formulated relation of conformance testing of "input-enabled transition systems" with "transitions labeled by input/output-pairs (IOLTSs)". IOLTSs presents an appropriate semantic model for the behavior characterized by a subset of the state-charts. Authors also developed a test case generation algorithm for state chart model.
Kosmatov *et. al.* [12]	Authors discussed the usage of a novel approach for automatic test case generation using set of test conditions and prescribed input domain. The developed approach basically accomplishes a boundary value investigation of input values of discrete neighborhood. Thereafter, a cost minimization (maximization) function is utilized within the domain for automatic test cases generation.
Gallagher *et. al.* [13].	Authors worked towards extending an approach developed for single class testing and found its applicability for integration testing of multiple classes. The proposed work represents the data flow information of various interacting classes in data flow graph form. Later, conventional data flow testing techniques are applied for generation of test cases for inter-class testing. A substantial contribution of the proposed research work is the development of the novel technique, which can represent data flow graphs in the form of relational database. Therefore, database queries can be used for utilizing "definition-use (DU)" data for deriving DU-paths.
Scheetz *et. al.* [14]	Authors have presented an approach for generation of system (black box) test cases by utilizing an "AI planner (Artificial Intelligence planner)". Authors utilized UML class diagrams and state diagrams for representation of conceptual architecture of a SUT (System Under Test). They could establish a new representation method at application level which permits the statement of test objectives available at that level along with their mapping on a planner representation. The proposed method tries to maps the initial and goal conditions into a problem statement for the planner. The planner produces a plan depending on the input. Further, the authors performed a translation of the plan to yield executable test cases. The objective of a test case in a "goal directed view" is to attempt to alter the state of the complete system to the goal state. The planner then decides the operators which can best accomplish the anticipated goal states.
Offutt *et.al.* [15]	Authors presented a methodology for system-level testing constructed on formal specifications. Their research work developed a model for descending test inputs from system level formal specifications based on some formal criteria using test data selection. Various testing criteria based on specifications are: "transition coverage criterion, full predicate coverage criterion, transition-pair coverage criterion and complete sequence coverage criterion". The basic assumption of the work was that the software's functionality specified in formal system specifications can be defined in terms of transitions and states. The approach first scanned the state-based specifications to produce the guard conditions for every transition. Further, this specification graph is used to generate edges which are annotated with transition conditions. Subsequently, authors proposed test requirements for every coverage criterion and generated test specifications for every test requirement.
Briand and Labiche [16]	Authors proposed an automatic methodology for originating test requirements using UML state-chart for fulfilling different coverage criteria like "all-transitions, all-transition pairs" etc. The focus of their proposed work was towards generating test requirement for specified transition test sequence by mining constraints of test data. The target was achieved by building an "Invocation Sequence Tree (IST)" which hold the interactions between state dependent objects. This IST comprises of essential information about invocation conditions of every event and action.

IV. CONCLUSION

This paper throws light on various UML based test case generation techniques developed so far different researchers. The UML state chart diagrams-based test case generation techniques have been discussed in the paper along with their details, applicability and usage. It is evident from the literature review conducted that the UML based test case generation techniques are the center area of research in software testing. More research needs to be conducted in the area to enhance the capability and efficiency of testing using latest algorithms and approaches.

REFERENCES

[1] R.V. Binder. Testing object-oriented software: a survey. Software Testing Verification and Reliability, Vol. 6, No. 3/4, pp. 125 - 252, 1996.

[2] Farooq, U. "Model Based Test Suite Minimization using Metaheurstic". Doctor of Philosophy, School of Computer Science and Security, Edith Cowan University, 2011.
[3] P. Fröhlick, and J. Link. Automated Test cases Generation from Dynamic Models. In Proceedings of the European Conference on Object-Oriented Programming, Springer-Verlag, Vol. 1850, pp. 472-491, 2000.
[4] J. Hartmann,M.Vieira, H.Foster and A.Ruder. A UML-based Approach to System Testing, Journal of Innovations system Software Engineering, Vol. 1, pp.12-24,2005.
[5] C. Nebut, F. Fleurey, Y. L. Traon, and J. Jean-Marc. Automatic Test Generation: A Use Case Driven Approach, IEEE Transaction on Software Engineering, Vol. 32, No. 3, pp. 140-155, 2006.
[6] A. J. Offutt, A. Abdurazik. Generating tests from UML specifications, In Proceedings of the 2nd International Conference on UML, Lecture Notes in Computer Science, Springer-Verlag GmbH, Fort Collins, TX, Vol. 1723, pp. 416 - 429, January 2001.
[7] S. Kansomkeat, and W. Rivepiboon. Automated-generating test case using UML statechart diagrams, In Proceedings of SAICSIT, ACM, pp. 296–300, 2003.
[8] Y.G. Kim, H.S. Hong, D.H. Bae, and S.D. Cha. Test cases generation from UML state diagrams, IEE Proceedings of Software, Vol. 146 , No. 4, pp. 187–192, August 1999.
[9] H.S. Hong, Y.G. Kim, S.D. Cha, D.H. Bae, and H. Ural. A Test sequence Selection Method for Statecharts, Software Testing Verification and Reliability, Vol. 10, No. 4, pp.203-227, 2000.
[10] J. Hartmann, C. Imoberdorf, and M .Meisinger. UML-based integration testing, ACM SIGSOFT Software Engineering Notes, In Proceedings of International Symposium of Software Testing and Analysis, Vol. 25, August 2000
[11] Stefania Gnesi, Diego Latella and Mieke Massink, Formal Test case generation for UML Statecharts, In Proceedings of the Ninth IEEE International Conference on Engineering Complex Computer Systems Navigating Complexity in the e-Engineering age (ICECC 2004), pp. 75-84, 2004
[12] Nikolai Kosmatov, Bruno Legeard, Fabien Peureux, and Mark Utting. Boundary Coverage Criteria for test generation from formal models, In Proceedings of 15th International Symposium of Software Reliability Engineering (ISSRE '04), 2004.
[13] L. Gallaghar, J. Offut, and A. Cincotta. Integration Testing of Object-Oriented Components Using Finite State Machines, Software Testing, Verification and Reliability, 2007.
[14] M. Scheetz, Von A. Mayrhauser, and R. France. Generating test cases from an object-oriented model with an AI planning system. In Proceedings of the 10th International Symposium on Software Reliability Engineering (ISSRE 99), IEEE Computer Society Press, pp. 250 – 259, 1999
[15] J. J. Li. Prioritize code for testing to improve code coverage of complex software, IEEE International Symposium on Software Reliability Engineering (ISSRE). IEEE Computer Society: Washington, DC, pp. 75–84, 2005.
[16] L.C. Briand, Y. Labiche. A UML-Based Approach to System Testing, Journal of Software and System Modelling, Vol. 1, No. 1, pp. 10-42, 2002.
[17] https://www.quora.com/What-are-UML-diagrams-related-to-structural-and-behavioral-modeling-respectively