Automated Model-Based Test Case Generation Using Uml Activity Diagram: A Review

Rozi Nor Haizan Nor, Md Abdul Monim, Yusmadi Yah Jusoh, and Nur Ilyana Ismarau Tajuddin

Abstract Software or application testing is a process of executing a program with the goal of finding defect to make better system. In software testing phase, writing test cases is one of the important activities. Manually writing test cases approach is lengthy of time period and need more effort to accomplish the process. This paper describes test case, test case generation techniques, different types of software testing approaches and comparison of testing tool. Test cases usually writing at the beginning of testing process from the set of software requirements. Test cases are created by using two different approaches. One is manually written test cases and another is automatically generated test cases. Manually written test cases are a very lengthy process and need to give a lot of effort to make good quality test cases. On the other hand, automatically generated test cases are generated by automatically using some software tools and it saves a lot of time and effort.

Keywords Test case generation · Testing technique · Model based testing · Automated test case generation · UML diagram

1 Introduction

A software system or a web application had to go through into a development life cycle and testing is an important phase of this software/application development life cycle. Software testing can be done by manually or automatically. In manual

R. N. H. Nor (✉) · M. A. Monim · Y. Y. Jusoh
Department of Software Engineering and Information System, Faculty of Computer Science and Information Technology, Universiti Putra Malaysia, Serdang, Malaysia
e-mail: rozinor@upm.edu.my

Y. Y. Jusoh
e-mail: yusmadi@upm.edu.my

N. I. I. Tajuddin
Tamhidi Centre, Universiti Sains Islam Malaysia, Nilai, Malaysia
e-mail: nur_ilyana@usim.edu.my

H. Kim and K. J. Kim (eds.), *IT Convergence and Security*, Lecture Notes in Electrical Engineering 782, https://doi.org/10.1007/978-981-16-4118-3_11

testing, the testing requires doing all the process by manually includes input, 2 analysis, and writing and managing the test cases. Manual testing deals with human interaction with all the process from beginning to end of the process and it is a time-consuming process. A human can get tired of doing all the process continuously. On the other hand, in automated testing most of the testing processes are automated. Generating test cases, executing the test cases and producing the test result are done by automatically. Though, one of the software testing fundamental is hundred percent automation is not possible in the field of software testing. Some task still need the intervention of human. Based on software testing approaches there are mainly three type of testing techniques have been using in software testing, those are 8 Specification-based testing, Code-based testing, and Model-Based testing. In Specification-based testing techniques test cases are made by directly from the software specification or some other kind of document that may have a different type of flow or direction. Therefore, in Code-based testing techniques, all the statement or module of the code have to execute at least once during the testing process and that process are followed by test cases that have written based on the code [1]. Lastly, in Model-Based testing approaches, model are developed from requirements and that model is the main resource to generate the test cases by manually or automatically. Writing or generating good quality test cases is very important for the test execution process.

2 Literature Review

This section describes test case, test case generation techniques, different types of software testing approaches and comparison of testing tool. A test case is a set of conditions, test inputs or variables under which a tester will determine whether a system test satisfies requirements or works properly. Test cases usually writing at the beginning of testing process from the set of software requirements. Test case needs to be ready before starting the test execution. Testers are followed by the step as mentioned in a test case using test data and compare the output after the test execution. Test cases are created by using two different approaches. One is manually written test cases and another is automatically generated test cases. Manually written test cases are a very lengthy process and need to give a lot of effort to make good quality test cases. On the other hand, automatically generated test cases are generated by automatically using some software tools and it saves a lot of time and effort. Based on software testing approaches there are mainly three type of testing techniques have been using in software testing, those are 8 Specification-based testing, Code-based testing, and Model-Based testing. In Specification-based testing techniques test cases are made by directly from the software specification or some other kind of document that may have a different type of flow or direction. Therefore, in Code-based testing techniques, all the statement or module of the code have to execute at least once during the testing process and that process are followed by test cases that have written based on the code [1]. Lastly, in Model-Based testing approaches, model are developed from requirements and that model is the main resource to generate the test cases by manually or automatically. Writing or generating good quality test cases is very important for the test execution process.

2.1 Model-Based Testing

"Model-based Testing is a testing technique where the runtime behavior of an implementation under test is checked against predictions made by a formal specification or model" [2]. Jupudy et al. [3] presented in their paper, Model-based testing comes under the black-box testing approach in which the internal design of the system is not taken into account during the test case generation. The process for MBT starts with generating functional test models based on the software requirements then this model is used for generating the test cases. The final test execution can be done either by manual or automated test execution. For test case generation it is based on the information provided by the test models so that the models should need to include the system behavior that the tester wishes to test. Model-Based Testing is a very suitable way to represent any system. Models can be as simple as a graph, flowchart, or diagram. A model of a software or system is a portrait of its behavior where the behavior can be described in terms of the input sequences that accepted by the system, set of actions, conditions and the flow of data through the system's module as shown in Fig. 1. These models can be created using any modelling tools. There were several Model-Based Testing tool available in the market that can generate test cases using different types of input format. The list is shown in Table 1.

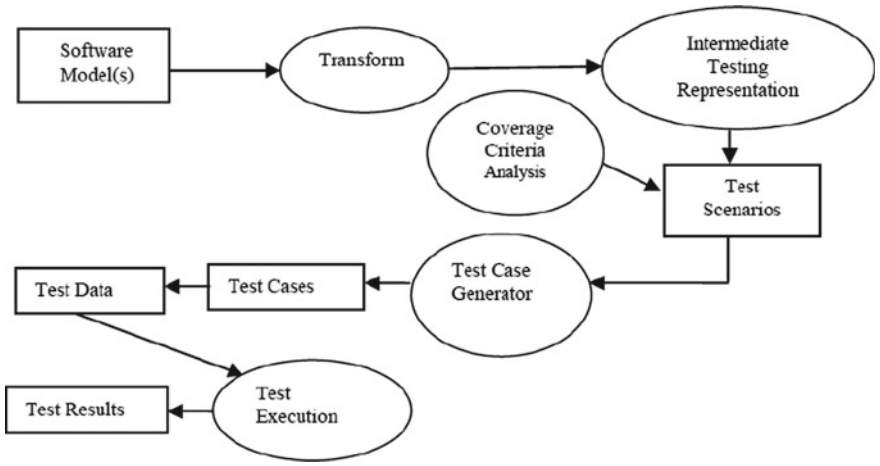

Fig. 1 A typical model-based testing (MBT) process Swain et al. [4]

Table 1 Different types of MBT tool for test case generation

No	Tool	Input format	Type	Description
1	BPM-Xchange	BPMN, UML	Commercial	BPM-Xchange creates test cases from business process models based on different criteria (statement, branch, path, condition). It can import models from several modelling tools, and can export test cases to Excel, HP Quality Center, etc
2	Conformiq designer	UML State Machines, QML	Commercial	Models can be created as UML State Machines and in Qtronic Modelling Language (QML). Test cases can be exported to test management tools or TTCN-3
3	fMBT	Custom (AAL)	Open source	fMBT (free Model-Based Testing) generates test cases from models written in the AAL/Python pre/post condition language using different heuristics (random, weighted random, look ahead)
4	Graphwalker	FSM	Open source	Test generation from Finite State Machines. Search algorithms: A* or random, with a limit for various coverage criteria (state, edge, requirement). Formerly called as MBT
5	JSXM	EFSM (Stream X-machines)	Academic	JSXM is model animation and test generation tool that uses a kind of EFSMs as its input. The generated tests can be transformed to JUnit test cases
6	MaTeLo	Markov chains	Commercial	MaTeLo (Markov Test Logic) is a commercial product to generate functional test cases. Strategies: random generation oriented by profiles, all transitions coverage. Can be connected to numerous test platforms
7	MISTA	Petri net	Academic	MISTA generates test cases from high-level Petri nets, and using a mapping it can generate executable test code for various platforms (JUnit, NUnit, Selenium). It can be used for functional or security testing
8	ModelJUnit	EFSM	Open source	Model J Unit allows to write simple finite state machine (FSM) models or extended finite state machine (EFSM) models as Java classes, then generate tests from those models and measure various model coverage metrics
9	MoMuT::UML	UML state machines, OOAS	Academic	MoMuT is a family of automated, model-based test case generation tools that can work off UML State Machines, Assume-Guarantee Contracts (REQS), and Object Oriented Action Systems (OOAS). The tools feature a fault-based test case generation strategy (using mutation operators)

(continued)

Table 1 (continued)

No	Tool	Input format	Type	Description
10	RT-Tester	UML/ SysML, Matlab	Commercial	RT-Tester starts from UML/SysML or Matlab models, transforms them to a internal representation based on Kripke structures, transform requirements to LTL formulae, and generates test cases using an SMT solver based on the goals from requirements and various model coverage criteria
11	SmartestingC ertifyIt	UML + OCL	Commercial	This tool is the successor of BZ Testing-Tools and Leiros Test Generator, and now is a commercial product from Smartesting. The SUT is given with UML models enriched with OCL constraints, the tool generates tests to satisfy various model coverage criteria
12	Spec explorer	Model programs in C#	Commercial	Spec Explorer is the successor of the AsmL. The Spec Explorer is now integrated into Visual Studio. The models can be written in C#, and test generation is directed with test 13 purposes written in the Cord language
13	Tcases	Custom	Open source	Tcases is a combinatorial testing tool where the inputs of thesystem could be specified in an XML file (with conditions, failure values, don’t cares,etc.). Test cases can generate n-wise or randomized test suites
14	Test cast	UML State Machines	Commercial	Test Cast MBT Edition generates TTCN-3 test cases from UML State Machines based on requirement and model structural coverage. The product is the successor of the MOTES prototype tool
15	Test optimal	(E) FSM	Commercial	Test Optimal supports FSM and EFSM modelling with several test case generation algorithms. It has various plug-ins for online testing web application, windows applications, database and web services, etc. It also supports data-driven testing and pair-wise algorithms right within the model and has the facility for performing load testing using the same models

2.2 Automated Test Case Generation

Rushby [5] elaborates in his paper, most of the process of software test execution and monitoring is now automated in modern software development life cycle. But for the test case generation has remained the existing labor-intensive manual task in some of the software development practice. But technology is changing by its own way, so this existing technique is also changed and now many methods or approach now become available to automate this process. If a tool can be able to generate the test cases automatically by using any appropriate input and any method then it can be called automated test case generation process. Then, Automated Model-Based test case generation is a technique that a visual model is used for generating the test cases based on some method or algorithm into that tool.

2.3 Related Works

Sumalatha and Raju [6] described in their paper, there are a lot of models and each describes different aspects of software or system behavior. Like, control flow, data flow and program dependency graph manifest how the implementation behaves by representing its source code structure. Decision table and state machines are used to describe external behavior. There are many models have been using in software testing those 14 are finite state machines, state charts, Unified Modeling Language (UML), Markov chains and grammars. Model-Based Testing (MBT) is the next step in the evolution of software testing. Model-Based testing has been proven to allow more effective work and increase attention on the substance of testing. Priya and Sheba [7] conducted a survey about test cases generation from different types of UML model and presented all the survey report in their paper. They choose five UML model that combination of UML structural and behavioral diagram those are, (i) Activity diagram, (ii) Sequence diagram, (iii) Class diagram, (iv) State-chart diagram and, (v) Collaboration diagram. They also described some test case generation techniques using those above UML diagram, some of the technique was based on single UML diagram and some of the technique was based on combination of two UML diagram. [8] presented a review study about different test case generation techniques, test case selection method, test case minimization techniques, test case prioritization techniques and some test case evaluation techniques. The principal techniques were critical path method, code based test generation, GUI based test case generation, and Dynamic path testing and evolutionary testing. And they also provided some algorithm for generating test case such as, Graph traversal algorithm, and Genetic algorithm. An Executable Test Generation from UML Activity Diagram Using Genetic Algorithm approach was proposed [9]. They described an automated test case generation techniques from UML activity diagram using Genetic algorithm. They used UML to XMI as a modelling interchange and also used an API named Robotium, based on Genetic Algorithm (Table 2).

Table 2 Summary table of some related works

No	Author	Title	Focus area/ Approach	Description
1	Kaur and Gupta [10]	Automated model–based test-path generation from UML diagrams via graph coverage techniques	-New approach for graph covering technique. -Automated tools. -Chinese postman algorithm. -Prefix based algorithm	They used a tool named Test Optimal, is an integrated next-generation test design and test automation toolset powered by Model-Based Testing (MBT) to test case generation and test automation
2	Priya and Sheba [7]	Test case generation from UML models—a survey	-Survey of five UML diagram. - Class diagram - State chart diagram - Sequence diagram - Activity diagram - Collaboration diagram.	Performed a literature review for test case generation from UML structural and behavioural diagram, test case generation by a combinational approach and, different type
3	Chouhan et al. [11]	Test case generation on the origin of activity diagram for navigational mobiles	- UML activity diagram - Model-basted testing - Mobile systems - Navigation systems	This work proposes a model for test case generation for navigation mobile application based on activity diagram. And the complexity calculated by Cyclomatic Complexity. The proposed mode introduces an algorithm that automatically creates a table called Activity Dependency Table (ADT) and then uses it to create a directed graph called Activity Dependency Graph (ADG). Finally, the ADG with the ADT is used to generate the final test cases

(continued)

Table 2 (continued)

No	Author	Title	Focus area/ Approach	Description
4	Suhag and Bhatia [12]	Model-based test cases generation for web applications	- MBT for web application - Using web diagram & sequence diagram	Test case generation technique has been applied to a web application which allows sharing of previous year papers, notes and other academic notices among each other
5	Hooda and Chhillar [8]	A review: study of test case generation techniques	Review study based on some test case generation techniques, - UML diagrams - Critical path method - Code based test generation - GUI based test case generation - Dynamic path testing and evolutionary testing - Graph traversal algorithm - Genetic algorithm	Presented a review study about different test case generation techniques, test case selection method, test case minimization techniques, test case prioritization techniques and some test case evaluation techniques

(continued)

Table 2 (continued)

No	Author	Title	Focus area/ Approach	Description
6	Jain et al. [13]	Automatic test case generation using UML models	- Automated test case generation - UML activity diagram - UML use-case diagram - Full predicate coverage criteria - XML standard for exchanging UML models	This paper describes an approach for test case generation from combination of UML Activity Diagram and Use Case Diagram. At first, AD are converted into activity Graph. Then extracting concurrent control flow path from activity graph. For CCFP they used depth first search (DFS) algorithm. Finally they use a combinational approach using use-case and activity diagram to generate test case
7	Shah et al. [14]	Automated test case generation using UML class & sequence diagram	- UML class diagram - UML sequence diagram - Model-based testing - Object oriented language - Test Automation	Visual Paradigm tool is used to create class diagram and again same tool is used to create sequence diagram. The developed diagrams then have exported into XML format. Total coding have done with C# and generated test cases saved into txt file

(continued)

Table 2 (continued)

No	Author	Title	Focus area/ Approach	Description
8	Anbunathan and Basu [9]	Executable test generation from UML activity diagram using genetic algorithm	- UML activity diagram - Genetic algorithm - Test automation - Pairwise testing	Activity Diagram is created to capture input scenarios. XMI file obtained from this AD is parsed to extract model information. A Control Flow Graph (CFG) is derived from edges by sorting the edges using Breadth First Search (BFS) algorithm. A recursive algorithm is developed to obtain path test cases from CFG. To generate test scripts, Robotium APIs are identified from Edges and Robotium API database using Genetic Algorithm
9	Patil and Jadhav [15]	Functional test case generation based on model driven testing using FSM and UML activity diagram	- UML activity diagram - Finite state machine - Model-based testing	Input FSM, Activity diagram in the form of XML. Then Process FSM $\oplus$ Activity Diagram. Activity Dependency Table for Activity Diagram and DFSM Graph Generator for FSM Output is test cases with all path, prioritization and removal of redundancy

(continued)

Table 2 (continued)

No	Author	Title	Focus area/ Approach	Description
10	Teixeira and Silva [16]	EasyTest: an approach for automatic test cases generation from UML activity diagrams	- UML activity diagram - Model-based testing - Test automation - EasyTest tool	Presents an automatic approach to generate test cases from UML activity diagrams using gray-box technique. The EasyTest approach comprises three phases, 1) importing activity diagrams in XMI; 2) test cases generation; and 3) applying test cases. Used Activity Dependency Graph and Activity Dependency Tree for sequencing the test path. Then applied into automated test case generation

3 Conclusion

This paper presented the background of different kind of test case generation techniques, methods, approaches, and tools were discussed. A summary table was provided based on existing similar research study. If the software testing stage can be finished early in software development life cycle then the total development process will be shortened and the software product is possible to deliver early. Manual test case writing approach is lengthy of time period and need more effort to accomplish the process. Manually have to write all test cases from the requirement and then execute the test cases are also done by manually. Writing test cases from the requirement is a very long process, boring and error-prone. Hence, automated test case generation is the way to solve this issue. Therefore, this paper contains an overview of test case generation and testing technique, Model-Based Testing, automated test case generation, comparison of different existing Model-Based Testing tools, related works that include a table of summary of some related previous works.

References

1. Kaushik S, Tyagi K (2016) Critical review on test case generation systems and techniques. Int J Comput Appl 133(7):24–29. https://www.ijcaonline.org/archives/volume133/number7/23798-2016907916
2. Sharma HK, Singh SK, Ahlawat P (2014) Model-based testing: the new revolution in software testing. Database Syst J 4(1):26–31
3. Jupudy I, Saraf N, Manjula R (2016) Comparative analysis of model based and formal based software testing methods. Int J Adv Res Comput Sci Softw Eng 6(3):49–58
4. Swain SK, Pani SK, Mohapatra DP (2010) Model based object-oriented software testing. J Theor Appl Inform Technol Vol-14. http://www.jatit.org/volumes/research-papers/Vol14No1/4Vol14No1.pdf
5. Rushby J (2007) Automated test generation and verified software. In Conference on verified software: theories, tools, and experiments (VSTTE). Zurich, Switzerland, pp 161–172
6. Sumalatha VM, Raju GSVP (2012) UML based automated testcase generation technique using activity-sequence diagram. Int J Comput Sci Appl (TIJCSA) 1(9):58–71
7. Priya SS, Sheba PD (2013) Test case generation from UML models–a survey. Int Conf Inform Syst Comput (ICISC) 3(1):449–459. https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.414.104&rep=rep1&type=pdf
8. Hooda I, Chhillar R (2014) A review: study of test case generation techniques. Int J Comput Appl 107(16):33–37. https://www.ijcaonline.org/archives/volume107/number16/18839-0375
9. Anbunathan R, Basu A (2017) Executable test generation from uml activity diagram using genetic algorithm. Int J Comput Sci Inf Technol Secur (IJCSITS) 7(3):1–6
10. Kaur P, Gupta G (2013) Automated model-based test path generation from UML diagrams via graph coverage techniques. Int J Comput Sci Mob Comput 2(7):302–311. https://ijcsmc.com/docs/papers/July2013/V2I7201365.pdf
11. Chouhan C, Shrivastava V, Sodhi SP, Soni P (2013) Test case generation on the origin of activity diagram for navigational mobiles. Int J Adv Comput Eng Netw 1(1):32–36. http://iraj.in/journal/IJACEN/paper_detail.php?paper_id=21&name=Test_Case_Generation_on_The_Origin_of_Activity_Diagram_For_Navigational_Mobiles
12. Suhag V, Bhatia R (2014) Model based test cases generation for web applications. Int J Comput Appl 92(3):23–31. https://doi.org/10.1007/978-3-319-09153-2_19

13. Jain SP, Lalwani KS, Mahajan NK, Gadekar BJ (2014) Automatic test case generation using UML models. Int J Adv Comput Eng Netw 2(6):30–34. https://ieeexplore.ieee.org/abstract/document/4418295
14. Shah S, Shahzad R, Bukhari S, Humayun M (2016) Automated test case generation using UML class & sequence diagram. Br J Appl Sci Technol 15(3):1–12
15. Patil SS, Jadhav PA (2017) Functional test case generation based on model driven testing using FSM and UML activity diagram. Int J Adv Res Comput Sci 8(5):1527–1530. https://search.proquest.com/openview/cbc76318c598ff45d1a9fe6cb762447c/1?pq-origsite=gscholar&cbl=1606379
16. Teixeira FA, and Silva GB (2017) EasyTest: an approach for automatic test cases generation from UML activity diagrams. Adv Intell Syst Comput Inform Technol New Generat pp 411–417. https://doi.org/10.1007/978-3-319-54978-1_54