A Comparative Evaluation of Test Cases Generation Approaches based on Unified Modeling Language for Web-based Applications

Dhafer AbdulAmeer AbdulMonim
Department of Computer Science, Open Educational College
Ministry of Education
Najaf, Iraq

ABSTRACT
Currently, web-based applications have a huge role in providing online services across the world in various sectors such as social media, e-commerce, e-education, e-health, etc., due to the rapid growth, dynamic nature and heterogeneity of web-based applications, Rigorous testing techniques are required to produce reliable web-based applications. Therefore, the process of testing web-based applications is a very important issue. Software testing is an important and essential stage in software development to improve its quality and reliability. There are many methods presented in the literature for creating test cases. Besides, you can take the test manually or automatically using different test tools. Therefore, this paper focuses on reviewing the literature and reviewing many different existing methods performed on Model Based Testing (MBT) for web-based applications. In addition, this research paper offers a comparative evaluation based on testing approaches for web-based applications that used UML diagrams in order to provide guidance for researchers to select an appropriate MBT approach and provide a basis for relevant future research.

Keywords
Model-Based Testing (MBT), Unified Modeling Language (UML), Web-Based Applications, Test Case Generation.

1. INTRODUCTION
In recent years, web-based applications have become used in important aspects of life, such as social media, education, entertainment, and online shopping [1]. A web-based application usually consists of a graphical interface that the user can interact with by navigating between pages through the browser [2]. There are two main types of web-based applications; One that only displays web pages is called static, and the other that allows the user to interact by entering information for the web page is called dynamic [3]. Given the importance of web-based applications in many areas, it is necessary to conduct testing with methods appropriate to their dynamic environment. Although testing using traditional testing techniques is expensive and time consuming [4]. Software testing has an essential and important role to verify that the system behavior matches its requirements and operates accurately and reliably [5]. The testing process is the process of assessing the quality of the system under test (SUT). The stages of the testing process are creating test cases, executing test cases, and finally evaluating test cases [6]. A test case is a series of inputs that lead to the expected output values. There are two techniques for generating test cases. The first is structural testing, which generates test cases based on source code and is called white box testing. In addition, functional testing, in which the generation of test cases depends on the specifications of the systems, is called black box testing [7]. Software testing is an important and costly process in the software development life cycle. In the current researches, new methods have been explored to create test cases based on system models, characterized by low test cost and time [8]. Model-based testing (MBT) is one of the black box testing approaches that have a role in improving the traditional testing process. MBT is a convenient approach to web-based application testing because it has the advantage of lower cost and time detection of defects compared to traditional testing approaches [10]. In MBT, test cases are created from abstract models to capture SUT behavior. System models are created depending on the system requirements during the design phase [11]. One of the models suitable for generating test cases is the behavioral models of the system [12]. In the literature, there are different types of models that are used to generate test cases, including behavioral models, such as state diagrams, petri networks, and Finite State Machines (FSM) [13]. Due to the importance of MBT and Unified Modeling Language (UML), as well as the widespread use of UML, UML diagrams have been used to create test cases. [14]. In fact, there are many proposed test case generation approaches for web-based applications. Therefore, this paper reviews the most prominent approaches to generate test cases based on UML models for web-based applications to identify existing MBT methods. Furthermore, this paper presents a comparative evaluation made on MBT approaches for web-based applications. Comparison criteria include UML models of representation, test type, test technique, automation tool, and approach's limitations.

The rest of the paper is organized as follows. Section 2 provides the basic concepts for MBT process. Section 3,4 describes the UML diagrams and test case generation respectively. In section 5, a review of test case generation approaches presented. Section 6 provides results and discussion for a comparative evaluation of current literatures. Finally, section 7 presents the conclusions and directions for future research.

2. MODEL-BASED TESTING
Model-based testing (MBT) is a testing approach that relies on generating test cases from a system design model and executing automatically generated test cases to detect many errors compared to manual testing [15]. Systems models are expressed in various ways such as a use case diagram or a class diagram [16]. There are some tools available to implement MBT. Testing tools can help reduce time and cost and improve automation testing functionality [17]. Due to the high complexity of some models, the number of test cases generated is very large. So executing all test cases for SUT is usually not feasible. Thus, different test coverage criteria are used to define

the testing process [18]. For example, all paths coverage i.e. each path is executed at least once. On the other hand, branch coverage i.e. every possible branch of the decision point is executed [19]. The algorithm for automatically generating test cases is the core of the MBT approach (i.e. the algorithm outputs a set of test cases) applied to the specification model either offline or with an internet connection [20]. The generated test cases can be executed on the system under test [21]. Then the results of the test implementation are compared with the expected results in the specification test model [22]. In the case of offline testing, the test cases generated are written down as scripts. Using offline testing can improve the final test set before execution and store for later use [23]. On the other hand, in the test case with internet connection, test cases are generated, executed and evaluated together, and then the next test input is generated based on the test output [24]. In this case, online testing is adaptable and more convenient than offline testing [25]. In MBT, the last step is to analyze the output of the test cases that have been executed to generate the test report based on the test results [26]. For example, requirements not covered by test cases and requirements that are validated at the end of the test can be checked [27]. The importance of MBT is to increase the effectiveness and efficiency, which is related to the criteria for choosing a test model if the system model is structural or behavioral [28]. MBT generally generates test cases from an abstract model of the system, including formal, semi-formal specifications such as UML diagrams [29]. UML relies on behavior models based on expected inputs and outputs to generate a test case in its implementation [30]. UML behavior models relied on expected inputs and outputs in order to generate test cases [30]. MBT based on different methods to generate test cases such as Labelled Transition Systems (LTS), Finite State Machine (FSM), Message Sequence Charts (MSC), and others [31]. One of the most used models for MBT is UML Diagrams [32], which will be explained in the next section.

3. UNIFIED MODELING LANGUAGE

Unified Modeling Language (UML) is a standard visual modeling language designed for documenting and designing object-based systems [33]. It is provided by the Object Management Group (OMG) [34]. Because UML is a visual language that supports modeling of system structure and behavior, it has been widely used for software design and modeling [35]. Currently there are many applications that have used UML to design, test, maintain, etc. in a model-driven software development environment [36]. The UML contains a number of diagrams to display specific features of the system [37]. UML models fall into two categories that include organization diagrams and behavior diagrams. Organization Diagrams (Body Diagrams mean the static structure of a system and its representation of the various levels of abstraction and implementation). By contrast, behavioral schemas (behavioral schemas mean the complex action of objects in a system) [38]. Structural models are represented by class diagrams, component diagrams, deployment diagrams, object diagrams, package diagrams, profile diagrams, and compound diagrams. Interaction diagrams fall into the category of behavioral models. Other diagrams in this category are use case diagrams, case diagrams, and activity diagrams [39].

4. TEST CASE GENERATION

The testing process consists of three phases: generation of test cases, test execution, and test evaluation. Generating test cases is the most challenging compared to the other two phases [40]. Generating test cases manually is usually error-prone and time-consuming, so automating the generation of test cases is more efficient and effective [41]. In addition to saving time and effort and reducing the number of errors. Moreover, the automated test reduces the cost compared to the manual test and also increases the reliability [42]. It is more efficient and reusable to generate test cases from models at an early stage of system design, which can then be easily updated if specifications change [43]. Generating test case from UML models is more effective because it allows the testing process to be executed in parallel with the software development cycle, resulting in reduced development time and cost and improved software quality and reliability [44].

5. TEST CASES GENERATION APPROACHES

Web-based application is significantly different from traditional application. It is either dynamic or interactive. Therefore, traditional testing methods and tools are not sufficient and suitable for web-based application testing [45]. Therefore, a variety of web-based applications testing approaches has been proposed. This section gives a review of several of these approaches.

Rika and Tonila [46] proposed an approach to testing web-based applications with a high level of abstraction. The proposed approach is based on static HTML links and does not integrate any dynamic aspects of the program. Through the approach any web-based applications can be materialized with a UML model. The model is supported by a tool that generates tests consisting of a sequence of URLs and another tool that creates a static graph based on HTML links. Based on the navigation path of web-based applications s and to create linked test cases semi-automatically the proposed approach used the UML class diagram to define white box test criteria. On the other hand, Cho and Chong [47] proposed an approach to generating test cases for web-based applications s from a UML sequence diagram. The proposed approach generates three types of test cases to be tested, starting with a single web page test, then cross web page testing and finally integrated web page testing. The test was conducted with the help of a testing tool called OnlineTestWeb. Another approach to web-based applications testing suggested by Huang and Chen [48]. Their approach is generating test cases by using activity diagram for web-based applications testing. The approach focuses on testing the functional scenario of a web-based applications. The approach is based on the use of HttpUnit (a web test tool) and test cases in the form of test codes. Besides using the Web Application Scenario Automated Testing Tool (WASATT), test cases are created.

Finally compile and run the test codes. The proposed approach can reduce errors and the work can be extended to include a fully functional model of the web-based applications, moreover carefully checking the dynamic semantics of the web page. The Use Transition Case Model (UCTM) is the proposed approach by Li et al. [49]. An approach that designs web-based applications as UML state diagrams and uses a hierarchical profile called. By traversing UCTM from top to bottom, each use case is described as a sequential graph, which automatically converts it into a Restricted Message On Vertex Graph (RMOVG). The header in RMOVG represents one message in the sequence diagram. According to the CMC standards, each message must be passed at least once. Test cases generated from RMOVG satisfy CMC and result in reduced number of test cases. Besides, Fujiwara et al. [50] have introduced an approach to generate test cases for web-based applications using class diagram and Object Constraints Language (OCL). UML class diagrams are used to represent the structure of data while OCL is used to model the behavior and limitations of that data. Each generated test case has three levels: a pre-state, a

post-state, and an event start. The Modulo Theorem (SMT) analyzer is used for compatibility with more complex data types.

Moreover, García and Dueñas [51] They provided an approach to automated testing of web-based applications. The proposed approach includes four phases which are generation of test cases, then test derivation of data, then test case implementation, and finally reporting of test case. In addition to the approach inputs are UML models, XML files or Selenium scripts. The output as a report submitted for each test case. Furthermore, Nabuco and Paiva [52] introduced an approach that uses UML diagrams which are sequence diagrams (which provide dynamic navigation of the application under test) and web diagrams (which provide functional requirements such as attributes, operation, shape, and frameset) to generate test cases for web-based applications s. By applying the principles of data mining in SQL, test cases are created. This approach was validated and validated by a case study. Their study concluded that problems with deleted data were overcome and information on test cases improved. Because test cases are generated automatically from sequence diagrams and web diagrams, it has contributed to capturing the dynamic behavior of web-based applications.

6. RESULTS AND DISCUSSION

This section compares, analyzes and evaluates MBT approaches for web-based applications testing based on specific parameters. The comparative parameters are UML diagram, testing type, testing technique, automation tool, approach's limitations. These parameters are important to select a proper MBT approach for web-based applications testing. Based on currently literatures review of testing approaches, the results are obtained and summarized in Table 1.

As most web applications are an important resource for providing online services in many critical areas including commercial ones that require their reliability. Therefore, it must be tested and ensured for its effectiveness and accuracy. But web-based applications testing using traditional testing techniques is a major and complex problem because these techniques are expensive and time consuming. Therefore, researchers have introduced several MBT methods and this approach is feasible for testing web-based applications because it detects flaws while reducing cost and time.

Based on Table 1, test cases approaches in literature used different UML diagrams as use case, class, sequence, and activity. As well as, testing techniques used as black box or white box. Moreover, the approaches used are automated. Comparative evaluation explains the different approaches used in the testing process by showing the drawbacks of each approach. The results show that a better approach to test case generated is required. It may involve combining more UML diagrams as well as combining two or more technologies to improve existing approaches.

7. CONCLUSION

This study, presented the applicability of MBT to web-based applications as Testing Type, Testing Technique, automation tool. In addition, the UML diagrams used in these methods are highlighted. The results of the evaluation comparison showed that the majority of the UML diagrams used in the mentioned methods are Class Diagram, Sequence Diagram, Activity Diagram, Use Case. Moreover, the approach was analyzed to highlight issues and limitations such as complex or weaknesses. Based on the comparative evaluation presented of MBT methods for web-based applications. This study fulfilled its main objective of providing guidance to researchers for selecting an appropriate MBT approach. Besides providing future guidance to researchers and planning work for different web-based application areas such as load testing, security and navigation.

Table 1: Comparative evaluation on proposed approaches of web-based applications testing

No	Reference	UML Model Used	Testing Type	Testing Technique	Tool	Limitations
1	Ricca & Tonella (2001) [46]	Class Diagram	Factional Testing	White Box Testing	✓	This approach is suitable for testing static web pages only, and there is a possibility that the size of the test suite will expand due to the path explosion.
2	Cho & Chong (2005) [47]	Sequence Diagram	Unit Testing And Integration Testing	Black Box Testing	✓	The approach is fairly simple and straightforward however, the tool used is no longer available online.
3	Huang & Chen (2006) [48]	Activity Diagram	Factional Testing	White Box Testing	✓	The approach is considered tedious as the test schematic diagram then needs to be translated into assembled test scripts.
4	Li et al. (2008) [49]	Use Case Diagram and Sequence Diagram	Unit Testing	White Box Testing	✓	The approach is limited to branch testing so requires slicing in the case of large web applications.
5	Fujiwara et al (2011) [50]	Class Diagram And OCL	Unit Testing	Black Box Testing	✓	The approach has proven to be effective, but it requires that the tester be well trained in formal modeling.
6	García & Dueñas (2011) [51]	Use Case, Activity, And Presentation Diagram	Functional Testing	Gray Box Testing	✓	The approach is very suitable for asynchronous web systems. Because he did not use graphs to represent navigation as a set of states and transitions rather than web pages and links.

7	Nabuco & Paiva (2014) [52]	Sequence Diagram and Web Diagram	Functional Testing	Black Box Testing	✓	The approach uses model testing as a tool for building static and specific models based on user interface patterns to build test models and to generate and execute test cases. But to avoid test case explosion, test cases should be filtered based on specific configuration or randomly filtered.

8. REFERENCES

[1] Raut, V., & Patil, P. (2016). Use of Social Media in Education: Positive and Negative impact on the students. International Journal on Recent and Innovation Trends in Computing and Communication, 4(1), 281-285.

[2] Ullah, S. E., Alauddin, T., & Zaman, H. U. (2016, January). Developing an E-commerce website. In 2016 International Conference on Microelectronics, Computing and Communications (MicroCom) (pp. 1-4). IEEE.

[3] Srivastava, N., Kumar, U., & Singh, P. (2021). Software and performance testing tools. Journal of Informatics Electrical and Electronics Engineering (JIEEE), 2(1), 1-12.

[4] Kundu, S. (2012). Web testing: tool, challenges and methods. IJCSI International Journal of Computer Science Issues, 9(2), 1694-0814.

[5] Ammann, P., & Offutt, J. (2016). Introduction to software testing. Cambridge University Press.

[6] Thanakorncharuwit, W., Kamonsantiroj, S., & Pipanmaekaporn, L. (2016, December). Generating test cases from UML activity diagram based on business flow constraints. In Proceedings of the Fifth International Conference on Network, Communication and Computing (pp. 155-160).

[7] Sawant, A. A., Bari, P. H., & Chawan, P. M. (2012). Software testing techniques and strategies. International Journal of Engineering Research and Applications (IJERA), 2(3), 980-986.

[8] Tuteja, M., & Dubey, G. (2012). A research study on importance of testing and quality assurance in software development life cycle (SDLC) models. International Journal of Soft Computing and Engineering (IJSCE), 2(3), 251-257.

[9] Akour, M., Falah, B., & Kaddouri, K. (2016). ADBT Frame work as a testing technique: An improvement in comparison with traditional model based testing. International Journal of Advanced Computer Science and Applications, 7(5).

[10] Garousi, V., Keleş, A. B., Balaman, Y., Güler, Z. Ö., & Arcuri, A. (2021). Model-based testing in practice: An experience report from the web applications domain. Journal of Systems and Software, 180, 111032.

[11] Lebeau, F., Legeard, B., Peureux, F., & Vernotte, A. (2013, March). Model-based vulnerability testing for web applications. In 2013 IEEE Sixth International Conference on Software Testing, Verification and Validation Workshops (pp. 445-452). IEEE.

[12] Mariani, L., Pezze, M., Riganelli, O., & Santoro, M. (2012, April). Autoblacktest: Automatic black-box testing of interactive applications. In 2012 IEEE fifth international conference on software testing, verification and validation (pp. 81-90). IEEE.

[13] Kumar, S. S. (2019). Model Based Object-Oriented Software Testing.

[14] Ahmad, T., Iqbal, J., Ashraf, A., Truscan, D., & Porres, I. (2019). Model-based testing using UML activity diagrams: A systematic mapping study. Computer Science Review, 33, 98-112.

[15] Villalobos-Arias, L., Quesada-López, C., Martinez, A., & Jenkins, M. (2019). Model-based testing areas, tools and challenges: A tertiary study. CLEI Electronic Journal, 22(1), 3-1.

[16] Ma, C., & Provost, J. (2019). Introducing plant features to model-based testing of programmable controllers in automation systems. Control Engineering Practice, 90, 301-310.

[17] Li, W., Gall, F. L., & Spaseski, N. (2017, March). A survey on model-based testing tools for test case generation. In International Conference on Tools and Methods for Program Analysis (pp. 77-89). Springer, Cham.

[18] Utting, M., Pretschner, A., & Legeard, B. (2012). A taxonomy of model-based testing approaches. Software testing, verification and reliability, 22(5), 297-312.

[19] Schieferdecker, I., & Hoffmann, A. (2012). Model-Based Testing. IEEE software, 29(1), 14-18.

[20] Saifan, A., & Dingel, J. (2008). Model-based testing of distributed systems. Technichal report, 548(2008).

[21] Gurbuz, H. G., & Tekinerdogan, B. (2018). Model-based testing for software safety: a systematic mapping study. Software Quality Journal, 26(4), 1327-1372.

[22] Schieferdecker, I., & Hoffmann, A. (2012). Model-Based Testing. IEEE software, 29(1), 14-18.

[23] Gutiérrez, J. J., Escalona, M. J., & Mejías, M. (2015). A model-driven approach for functional test case generation. Journal of Systems and Software, 109, 214-228.

[24] Veanes, M., Campbell, C., Schulte, W., & Tillmann, N. (2005, September). Online testing with model programs. In Proceedings of the 10th European software engineering conference held jointly with 13th ACM SIGSOFT international symposium on Foundations of software engineering (pp. 273-282).

[25] Hielscher, J., Kazhamiakin, R., Metzger, A., & Pistore, M. (2008, December). A framework for proactive self-adaptation of service-based applications based on online testing. In European Conference on a Service-Based Internet (pp. 122-133). Springer, Berlin, Heidelberg.

[26] Bouquet, F., Jaffuel, E., Legeard, B., Peureux, F., & Utting, M. (2005). Requirements traceability in automated

test generation: application to smart card software validation. ACM SIGSOFT Software Engineering Notes, 30(4), 1-7.

[27] Legeard, B. (2010). Model-based testing: Next generation functional software testing. In Dagstuhl Seminar Proceedings. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik.

[28] - Rocha, M., Simão, A., & Sousa, T. (2021). Model-based test case generation from UML sequence diagrams using extended finite state machines. Software Quality Journal, 29(3), 597-627.

[29] Kansomkeat, S., Offutt, J., Abdurazik, A., & Baldini, A. (2008, August). A comparative evaluation of tests generated from different UML diagrams. In 2008 Ninth ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing (pp. 867-872). IEEE.

[30] Pretschner, A., Prenninger, W., Wagner, S., Kühnel, C., Baumgartner, M., Sostawa, B., ... & Stauner, T. (2005, May). One evaluation of model-based testing and its automation. In Proceedings of the 27th international conference on Software engineering (pp. 392-401).

[31] Shirole, M., & Kumar, R. (2013). UML behavioral model based test case generation: a survey. ACM SIGSOFT Software Engineering Notes, 38(4), 1-13.

[32] Rocha, M., Simão, A., & Sousa, T. (2021). Model-based test case generation from UML sequence diagrams using extended finite state machines. Software Quality Journal, 29(3), 597-627.

[33] Conallen, J. (2003). Building Web applications with UML. Addison-Wesley Professional.

[34] Torre, D. (2016, October). Verifying the consistency of UML models. In 2016 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW) (pp. 53-54). IEEE.

[35] Kos, T., Mernik, M., & Kosar, T. (2019). A tool support for model-driven development: An industrial case study from a measurement domain. Applied Sciences, 9(21), 4553.

[36] Jafer, S., Durak, U., Aydemir, H., Ruff, R., & Pawletta, T. (2018). Advances in Software Engineering and Aeronautics. In Advances in Aeronautical Informatics (pp. 87-102). Springer, Cham.

[37] Rapolu, R. K. (2018). Selection of UML models for test case generation: A discussion on techniques to generate test cases.

[38] Minhas, N. M., Masood, S., Petersen, K., & Nadeem, A. (2020). A systematic mapping of test case generation techniques using UML interaction diagrams. Journal of Software: Evolution and Process, 32(6), e2235.

[39] Makedonski, P., Adamis, G., Käärik, M., Kristoffersen, F., Carignani, M., Ulrich, A., & Grabowski, J. (2019). Test descriptions with ETSI TDL. Software Quality Journal, 27(2), 885-917.

[40] Boghdady, P. N., Badr, N. L., Hashem, M., & Tolba, M. F. (2011). A proposed test case generation technique based on activity diagrams. International Journal of Engineering & Technology IJET-IJENS, 11(03), 1-21.

[41] Felderer, M., & Herrmann, A. (2015). Manual test case derivation from UML activity diagrams and state machines: A controlled experiment. Information and Software Technology, 61, 1-15.

[42] Lafi, M., Alrawashed, T., & Hammad, A. M. (2021, July). Automated Test Cases Generation from Requirements Specification. In 2021 International Conference on Information Technology (ICIT) (pp. 852-857). IEEE.

[43] Wang, X., Guo, L., & Miao, H. (2008, December). An Approach to transforming UML model to FSM model for automatic testing. In 2008 International Conference on Computer Science and Software Engineering (Vol. 2, pp. 251-254). IEEE.

[44] Tiwari, R. G., Srivastava, A. P., Bhardwaj, G., & Kumar, V. (2021, April). Exploiting UML diagrams for test case generation: a review. In 2021 2nd international conference on intelligent engineering and management (ICIEM) (pp. 457-460). IEEE.

[45] Yu, B., Ma, L., & Zhang, C. (2015, November). Incremental web application testing using page object. In 2015 Third IEEE Workshop on Hot Topics in Web Systems and Technologies (HotWeb) (pp. 1-6). IEEE.

[46] Ricca, F., & Tonella, P. (2001, May). Analysis and testing of web applications. In Proceedings of the 23rd International Conference on Software Engineering. ICSE 2001 (pp. 25-34). IEEE.

[47] Cho, Y., Lee, W., & Chong, K. (2005, May). The technique of test case design based on the UML sequence diagram for the development of web applications. In International Conference on Computational Science and Its Applications (pp. 1-10). Springer, Berlin, Heidelberg.

[48] Huang, C. H., & Chen, H. Y. (2006, October). A tool to support automated testing for web application scenario. In 2006 IEEE International Conference on Systems, Man and Cybernetics (Vol. 3, pp. 2179-2184). IEEE.

[49] Li, L., Miao, H., & Qian, Z. (2008, December). A UML-based approach to testing web applications. In 2008 International Symposium on Computer Science and Computational Technology (Vol. 2, pp. 397-401). IEEE.

[50] Fujiwara, S., Munakata, K., Maeda, Y., Katayama, A., & Uehara, T. (2011). Test data generation for web application using a UML class diagram with OCL constraints. Innovations in Systems and Software Engineering, 7(4), 275-282.

[51] García, B., & Dueñas, J. C. (2011). Automated Functional Testing based on the Navigation of Web Applications. Workshop on Automated Specification and Verification of Web Systems (WWV 2011), pp. 49-65, 2011.

[52] Nabuco, M., & Paiva, A. C. (2014, June). Model-based test case generation for web applications. In International Conference on Computational Science and Its Applications (pp. 248-262). Springer, Cham.