International Journal on Advanced Science Engineering Information Technology

Vol.7 (2017) No. 4-2
ISSN: 2088-5334

Comparative Evaluation of the State-of-art Requirements-based Test Case Generation Approaches

Ahmad Mustafa[#], Wan M.N. Wan-Kadir[#], Noraini Ibrahim[#]

[#]*Software Engineering Department Universiti Teknologi Malaysia, Skudai, Johor Baharu, 81310, Malaysia*
E-mail: wnasir@utm.my

***Abstract*— The overall aim of software testing is to deliver the error-free and high-quality software products to the end users. The testing process ensures that a software is aligned with the user specification and requirements. In software testing process, there are many challenging tasks however test case generation process is considered as the most challenging one. The quality of the generated test cases has a significant impact on efficiency and effectiveness of the testing process. In order to improve the quality of a developed software, the test cases should be able to achieve maximum adequacy in the testing and requirements' coverage. This paper presents a comparative evaluation of the prominent requirement-based test case generation approaches. Five evaluation criteria namely, inputs for test case generation, transformation techniques, coverage criteria, time and tool's support are defined to systematically compare the approaches. The results of the evaluation are used to identify the gap in the current approaches and research opportunities in requirements-based test case's generation.**

***Keywords*— test case generation; requirements; specification-based testing; diagram-based testing; coverage criteria**

I. INTRODUCTION

Software testing is performed to assure the quality of a software product. Therefore, it is considered as one of the important phases of software development. In addition, it provides the system stakeholders with the measures on the degree of how far the system meets the customer's expectation.

Software testing takes 40-70% of the development effort, time, and cost [1]. In another study [2], software testing accounts for 50-75% of total development cost. Although software testing is an expensive phase, it has a significant impact on achieving a high degree of software quality and consumer confidence.

Nowadays, software testing researchers are concentrating on the efficiency of software testing process. Furthermore, automation of the software testing process is considered as a solution to lower the cost, decrease the effort, and increase the time to market of software product [3]. However, automation of testing is not a silver bullet which can provide a solution to all problems. It is considered that software test case automation process is itself an expensive activity and requires development effort, cost and time. Despite expenses, it has a significant impact on development cost, time to market and quality of the product [4].

Adequate validation of functional requirements in test cases is considered as one of the industrial software problems. The foremost effort to cover this difficulty stems that how each source requirement is typically mapped onto a respective test run result. The suitability of mapping a requirement onto a test is typically not formally recognized. Industry expert generally fills this gap in some way with test reviews or inspections [5].

Software requirements based testing is considered as an important phase of the software development process, as it addresses the problem that how to validate the developed software against its requirements. Designing of test cases is the foremost success factor of the testing process, and customers are concerned with test cases successful implementation. Furthermore, requirement based test case generation helps to obtain the adequate traceability information of requirements. It also ensures the complete requirements coverage in the designed test case suite [6].

A perfect set of test cases is one that has a high chance of discovering unknown errors. To uncover all potential errors in a program, detailed testing is required to examine all possible input and logical execution paths, but it is neither possible nor economically feasible. Thus, the actual goal of software testing is to increase the finding errors' probability using a limited number of test cases that perform in less time with less effort [7].

Many metrics have been proposed to evaluate the quality of test case generation, e.g., cost, time, effort and coverage criteria. Many researchers have put their efforts on minimization of time, effort and cost. However, from last decade, researchers put their efforts on automation of test

case generation [8]. The main function of requirement coverage metric is to monitor and report the number of requirements tested and whether these requirements are correctly implemented or not.

Similarly, a test coverage criterion shows that how effective the testing process has been done. As per testing approaches, there are two main types of coverage criteria: requirement coverage and structural coverage [9]. Requirements' coverage which covers all requirements in functional testing. Particularly, it is a measurement of the requirements that are covered in the testing and specifies the performance of functional testing achievement [10]. Likewise, the structural coverage checks how many behaviours of the requirements' specification are covered by the testing process [11].

This study presents a comparative evaluation of different approaches for automated test case generation from requirements, which is a critical part of software testing process and types of coverage criteria that are used in these methods.

In this paper, we classify the existing prominent approaches in requirements-based test case generation. In Section II, we briefly elaborate these approaches, we explain the process of test case generation from requirements, and we explain the evaluation criteria used to check the quality of approaches. Discussion on the evaluation results of approaches is presented in Section III. Conclusion and future research work are presented in Section IV.

II. Material and Methods

Test case generation process is the most important and fundamental testing process. If the test cases are generated earlier, it will reduce the cost, time and effort when actual testing starts.

A. Classification of Test Case Generation Approaches

In general, there are two major approaches in requirements-based test case generation namely specification-based testing and Sketched Diagram-based testing.

1) Specification-based Testing: In specification-based testing, requirements are used to develop the application and define the test cases. The defined test cases verify that all requirements are applied in the application domain. Specification-based methods for testing are the techniques to generate a bulk of test cases from the documentation regarding the specifications of the system such as a formal requirement specification [12], [13]. Specification-based testing depends on requirement's models. It can be further categorized in two requirements analysis approaches namely natural language requirements and formal method's analysis.

In the early phase of system development, requirements are specified in natural language because natural language is universal, flexible and comprehensible. In addition, natural language can be used to describe any circumstances and environment [14].

According to IEEE-830 standard, the term “natural language requirement” is referred as a requirement document that is documented using natural languages such as English, French, Arabic, Malay, and Urdu. “Shall" requirements, use case, user stories, scenarios, and feature lists are examples of natural language requirements [15]. “Shall” requirements are stated with “The system shall ...”, with the possibility to swap “system” with system or actor name for example, “The temperature sensor shall send the temperature of the ducts to the duct monitoring system” [16].

Formal methods are the rigorous mathematical-based approaches that are used for the specification, modeling, and verification of software and hardware systems [17]. Formal methods are associated with three techniques such as formal specification, formal verification, and refinements.

There are some drawbacks of using the formal methods as a specification-based technique. First, it is difficult to estimate the effort cost in conducting formal analysis due to the complexity of the analysis process [18]. Second, the great manual effort is required in generating test cases as compared to automated test case generation process [19], [20]. Third, formal specification methods are not widely accepted for several reasons. For example, a formal model is difficult to be used for communication purpose, especially for non-technical personnel. Furthermore, extensive training is required during the implementation of formal models, which is a time-consuming and expensive process.

2) Sketched Diagram-based Testing: The Sketched Diagram-based testing technique is the type of model-based testing. It is also known as UML-based testing because it generates test cases from the UML diagram. The test cases can be generated in analysis and design phase from the requirement-based model. UML diagrams are a most common way to represent the requirement-based model. The UML-based testing (UBT) is defined as a testing approach that uses UML-based software models or specifications in the design, generation, and execution of the test cases. In the UML-based testing process, UBT particularly designs and generates test cases (with oracles), and evaluates test results based on the relevant UML-based software models and UML-based specifications for testing the SUT [21].

The UML diagrams can be categorized into behavioural and structural diagrams [22].

- Behavioural Diagrams are the type of diagrams that represent behavioural features of a business or system process. These diagrams show that what should happen in a system. Behavioural diagrams include activity, sequence, use case, state chart and four interaction diagrams (communication, interaction, sequence, and timing). The interaction diagrams are used to describe that how objects interact with each other to create the functions within the system.
- Structural diagrams emphasise the elements of the specification which is irrespective of time. Diagrams included under this category are composite, deployment, package, profile, class, object and component diagrams.

B. Requirements based Test Case Generation Process

This sub-section briefly explains the process of two requirements based test case generation approaches namely specification-based test case generation process and Sketched Diagram-based test case generation process. The generation of test value is not covered by this work.

1) Specification-based Test Case Generation Process: Many of the researchers focus on the derivation of test cases from natural language requirements. The finite state machine and formal modelling languages are used to represent the requirement model of the system. The state machine is used to derive natural language scenarios. To write test cases from requirements, the test engineer must clearly understand the requirements' specification. Fig. 1 shows the generic process of test case generation from natural language requirements. It is a summarization of approaches [24]-[33], and a brief description of steps is mention below:

- In specification-based testing, documents are written in natural language / textual requirements.
- To increase the quality of natural language requirement, the approach [23] has performed syntactic and semantic analysis of natural language requirements.
- Different requirement formalization tools are used to develop a formal model from textual requirements, i.e., consistency checking tool analysed that formal model is in the correct state. If the formal model is not in the correct state, then requirements are inconsistent. In a failure, situation requirements are again analysed and corrected. If errors are not corrected, then inconsistency may result in subsequent phases of system development.

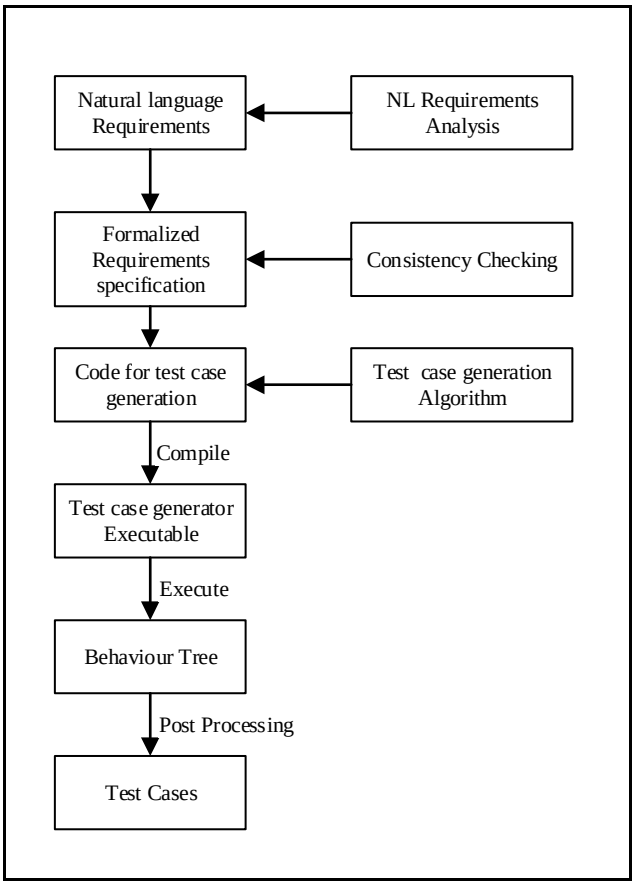

Fig. 1 Test case Generation Process from natural language requirements

- Test case generation tools derived test cases from the consistent formal model. Then test cases are executed on system implementation via the executable tool. The code for test cases is compiled with the generation algorithms.
- Then simulation of test cases is performed using any simulation tool, e.g., MATLAB Simulink Design verifier.
- The approach [24] applied behaviour trees (BTs) as a graphical notation to capture formal requirements. Using behaviour tree is easy to maintain direct traceability between individual functional requirements and their representation in the BT model.
- During post-processing test cases are extracted from behaviour tree by performing a modified depth-first traversal algorithm.

2) Sketched Diagram-based Test Case Generation Process

In Sketched Diagram-based test case generation, at the initial phase of requirements, requirement engineer starts describing the requirements as scenarios. These scenarios are transformed into a requirement model, e.g., activity or use case diagram. These diagrams are used for generation of initial test scenarios using the searching algorithm, e.g., graph search, Breadth-First Search (BFS) [34]-[36]. Fig. 2 shows the sequence of task's implementation for automation of test case generation process [36]-[39].

- Describe system requirements as scenarios.
- Transform requirement into any one UML diagram, e.g., activity, use case or class diagram.
- The XML metadata interchange (XMI) file is used to store the requirement model.
- It transforms requirement model into test model using meta modelling.
- When test model and metamodel have loaded a searching, the algorithm is adopted to generate different test item sequences from the test model.

C. Evaluation Criteria

This sub-section briefly explains the criteria used for evaluation of test case generation approaches.

1) The input to Generate Test Case: It refers to a type of information that is being used to generate test cases at the start of a testing process. It presents the information used for test cases generation process. For example, in case specification-based testing input may be in the form of natural language requirements or formal specification. Similarly, in Sketched Diagram-based testing approach input may be a use case, activity or class diagram.

2) Transformation Techniques: It refers to change that how from requirements to testing requirements are a process and reshaped. This metric is checking which transformation technique is more effective. Transformation techniques used in selected studies are formal checking, metamodelling, and formal methods.

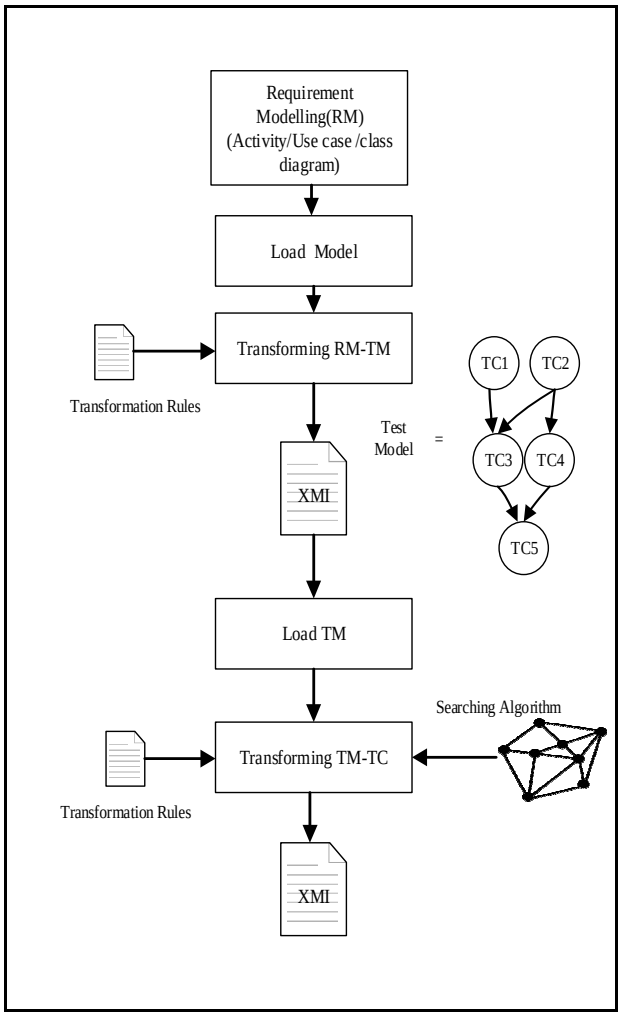

Fig. 2 Test case generation process from requirement's modelling

3) Coverage Criteria: Coverage metric is applied to quantify the quality of a system specification and is commonly applied to assess progress in system validation. For example, in the case of white box testing coverage metric answer question that "have I written enough tests cases?" Similarly, in black box testing, it addresses "Have I written enough properties?" Highest coverage to satisfy the test adequacy criteria that is given as input to the software under test, there are many types of coverage criteria depending on the type of testing.

4) *Time:* Through the testing process, the time can be increased due to unsuitable test cases. These unsuitable tests cases caused resources wasted as well as time. For that reason, there is a need to reduce time.

5) Tool Supports: It refers to use of Computer-Aided Software Engineering (CASE) tools for automation in the process.

III. Results and Discussion

Table 1 shows the result summary of comparison of requirements based test case generation approaches.

The evaluation metric time and CASE tool support responses are recorded in "yes" or "No" only. If any approach is not considering time factor during generation of test cases sign (X) is used, and in case yes sign (✓) is used. Similarly, if CASE tools are applied for automation in test cases generation process then sign (✓) and in case of no automation in test cases generation process sign (X) is used.

As we can see in Table 1, use case and sequence diagrams [35], [40], [42] are used as the input of test case generations. The proposed approach in [40] uses XMI parser for transformation to test cases. Equivalency class partition techniques are applied with the use of an automated tool to achieve maximum time to market and to minimize the cost.

In a similar manner, the approach in [36] has used activity diagrams as the input of test case generation whereas model transformation approach is used to convert the artefacts of activity diagrams into test cases. On the other hand, this study has no focus on time and cost. Moreover, no automation tool is used during the study.

Most of the studies [24], [25], [27]-[29], [32], [33] use natural language requirements as the input for their test case generation. At beginning phase of the system development, requirements are documented in natural language while the natural language's requirements may be ambiguous, incomplete and inconsistent. Meanwhile, manual inspection of natural language issues can be hard to minimize.

The study [28] proclaimed a method to generate test cases using natural language. The data-flow reactive system (DFRS) is used for transformation formal model. The (DFRS) automatically obtained artifacts from natural-language requirements that describe functional, reactive and temporal properties. Despite the generation of test cases from natural language, this study does not provide any mechanism of test case reduction.

Similarly, the requirement centred analysis and testing framework are used for the formalization of natural language requirements. After analysing the requirements' issues, test cases are generated from formalised requirements and executed on the implementation model [29]. However, this study is not addressing the issue of test case reduction. Moreover, no cost-effective solution is provided to minimize the cost of the testing process.

Likewise, the approach [27] proposed a method for generation of test cases from natural language (NL) requirements using an automated tool. This C&L tool translates automatically natural language requirement's descriptions into behavioural models for automated testing. Studies [27], [41] used path coverage, although it is a reliable metric though it is not applicable to large systems. Like previous studies, this study is not proposing any strategy of test case reductions.

In like manner, the study [25], proposed a method of test case's generation from natural language SRS documents. Text mining and symbolic execution techniques are used for generation of test cases. However, the study does not clearly state the transformation strategy and no cost-effective solution by focusing on test case reduction.

TABLE 1
COMPARISON OF TEST CASE GENERATION APPROACHES

	Approach	Input	Transformation Techniques	Coverage criteria	Time	Tools Support
Specification-based Approaches	Lindsay et al. [24]	Natural language	model checking	Path coverage	✕	✓
	Elghondakly et. al. [25]	Natural language	✕	Requirement coverage	✕	✓
	Venkatesh et al. [26]	Formal specification	Formal Model	Requirement coverage	✕	✕
	Sarmiento et al. [27]	Natural language	Model transformation	Path coverage	✕	✓
	Carvalho et al [28]	Natural language	Formal model	✕	✕	✓
	Aichernig et al. [29]	Natural language	model checking	Requirement coverage	✕	✓
	Carrera et al [30]	Natural language	Metamodeling	Functional coverage	✓	✓
	Gao et al. [31]	category partition	Formal model	Equivalency partition	✕	✓
	Yue et al. [32]	Natural language	Model transformation	Structural coverage	✕	✓
	Carvalho et al. [23]	Natural language	Model Transformation	Requirement coverage	✕	✓
Sketched Diagram-based Approaches	Chatterjee et al. [33]	Use case	Formal model	Requirement coverage	✕	✓
	Barbosa et al. [34]	Sequence & use case diagrams	XMI parser	equivalency partition	✕	✓
	Gutiérrez et al. [35]	category-partition method	Metamodeling	Path coverage	✕	✓
	Zhang et al. [36]	Use case diagram	Metamodeling	Structural coverage	✕	✓
	Ibrahim et al. [37]	Sequence & use case diagrams	✕	✕	✕	✓
	Gantait et al. [38]	Activity diagram	Model transformation	Path coverage	✓	✕
	Gutiérrez et al. [39]	category-partition method	Metamodeling	Functional coverage	✕	✓
	Granda et al. [40]	Requirement model	Model transformation	Path coverage	✕	✓
	Straszak et al. [41]	Use case scenarios	Metamodeling	Requirement coverage	✕	✓
	Wang et al. [42]	Use case	Metamodeling	Requirement coverage	✕	✓

In addition to above studies, the following study [30] proposed a methodology for test case generation using an open source framework Behavioural Agent Simple Testing (BEAST). It uses Behaviour Driven Development (BDD) techniques and Multi-Agent Systems (MASs) for generation of test cases process. Yet, this study can be only applied in agile development methodology. Moreover, the study is covering only functional requirements no emphasis on non-functional requirement coverage.

The study [30] presented a method to generate abstract level test suites from requirement's models using model-driven testing paradigm. Navigational Development Techniques are incorporated in functional system test cases. Moreover, the metamodeling is applied for verification of functional requirements in testing. However, at the same time focus on test case reduction.

Furthermore, the study [35] proposed a systematic approach for automated derivation of manually executable test cases from use case's model. The use cases and test cases are derived from restricted Natural Language with a tool support. Moreover, the presented approach helps in describing diverse test coverage criteria on requirements. However, the approaches are using structural coverage, which focuses on structural features, i.e., structural coverage and branch features, and it is not addressing requirement coverage.

In the same way, the proposed study [39] generates acceptance level test cases from use cases using the model-driven paradigm. Metamodeling is used for transforming the Requirements Specification Language (RSL) into test's cases. Same like above studies, this study is not providing any solution to minimise the cost and time during the testing process.

Likewise, research [26] proposed a method for test case's generation from a formal specification of the system. It implements the Expressive Decision Table (EDT) algorithm for requirement notation to reduce translation efforts of formal specification. After all, it is a fact that formal specifications are difficult to use for communication propose especially for non-technical personnel. Moreover, extensive

training is required for implementation of formal models, which is a time-consuming and expensive process.

Like the formal specification, the study [24] proposed the generation of test cases using a symbolic model checker. It ensures that test cases are correct and complete. The test case generated from the behaviour tree requirements model is traced back to the original requirements and with correctness and completeness guaranteed by the model checker. The approach is validated using case studies of an Automated Teller Machine, an air-traffic control system. However, the study is using model checking approach, which is mainly appropriate to test the control intensive application and it is less suitable for the data-intensive application. Moreover, it has the capacity to verify system model only, and it does not verify the actual system (product/prototype).

Consequently, it is concluded that most of the studies of requirements based test case generation [24]-[29], [31]-[35], [37], [38], [40]-[43] have no focused on requirement coverage. Moreover, there is no cost-effective solution of test case generation in the literature. No evidence found which is first focusing on natural language requirement's issues, for example, ambiguity, incompleteness, and inconsistency and subsequent generation of test cases.

IV. Conclusion

Software testing aims at assuring that the developed system conforms to the stated requirements and reducing errors arose during system operation. This paper presented our comparative evaluation of requirements based test case generation methods. The comparative evaluation was performed based on five evaluation criteria namely input of test case generation, transformation techniques, coverage criteria, time and tools support. The comparative evaluation results show that there is no single approach fulfil all evaluation criteria. Based on the evaluation, it is found that the specification-based approach is the more mature and effective approach for the generation of test cases. It has the capacity to effectively capture the behaviour within the system and maximum requirement coverage. Furthermore, it implements a rigorous mathematical model for verification and validation of the system.

However, the UML-based approach is very complex and contains a comprehensive set of numerous modelling diagrams and notations for general-purpose system modelling. UML activity and use case diagrams are often used to model and validate system requirements. Sequence and state machine diagrams to capture the behavior of the system and derive unit tests for the system. The class diagram is used to model classes and static structure of the system. It can be used for all testing levels.

UML sequence and state machine diagram are unable to capture non-functional requirements. Likewise, these techniques are not effective for verification and acceptance of the large software system.

In future, we will present a model of test case generation from requirement using Natural Processing Language (NLP) techniques. This model will be helpful in the effective formalization of requirements and generation of test cases. Moreover, this dynamic model will help the practitioner in the successful completion of a software project with a quality product, minimizing cost and time to market.

Acknowledgment

The authors would like to express their deepest gratitude to Research Management Center (RMC), Universiti Teknologi Malaysia (UTM) and Ministry of Higher Education Malaysia (MOHE) for their financial support under Research University Grant Scheme (Vot number Q.J130000.2516.11H71).

References

[1] N. Kosindrdecha and J. Daengdej, "A Test Case Generation Process and Technique," *Journal of Software Engineering 4(4): 265-287, 2010*, 2010.

[2] C. P. Lam, *Computational Intelligence for Functional Testing*: IGI Global, 2010.

[3] E. Alégroth, R. Feldt, and P. Kolström, "Maintenance of automated test suites in industry: An empirical study on Visual GUI Testing," *Information and Software Technology*, vol. 73, pp. 66-80, 2016 2016.

[4] D. Kumar and K. K. Mishra, "The Impacts of Test Automation on Software's Cost, Quality and Time to Market," *Procedia Computer Science*, vol. 79, pp. 8-15, 2016/01/01 2016.

[5] S. Baranov, V. Kotlyarov, and T. Weigert, "Verifiable Coverage Criteria for Automated Testing," in *SDL Forum*, 2011, pp. 79-89.

[6] Y. I. Salem and r. Hassan, "Requirement-based test case generation and prioritization," in *2010 International Computer Engineering Conference (ICENCO)*, Giza, 2011, pp. 152-157.

[7] M. S Geethadevasena and M. L Valarmathi, "Search based Software Testing Technique for Structural Test Case Generation," *International Journal of Applied Information Systems*, vol. 1, pp. 20-25, 2012.

[8] S. M. Mohi-Aldeen, S. Deris, and R. Mohamad, "Comparative Evaluation of Automatic Test Case Generation Methods," in *Proceedings of Technology, Education, and Science International Conference (TESIC) 2013: Developing Innovative Technology towards Better Human Life*, 2013, p. 66.

[9] M. Utting and B. Legeard, *Practical model-based testing: a tools approach*: Morgan Kaufmann, 2010.

[10] Y. Sun, G. Memmi, and S. Vignes, "A Model-Based Testing Process for Enhancing Structural Coverage in Functional Testing," *Complex Systems Design & Management Asia 2016*, pp. 171-180, 2016.

[11] I. Ober and I. Ober, *SDL 2011: Integrating System and Software Modeling: 15th International SDL Forum Toulouse, France, July 5-7, 2011. Revised Papers* vol. 7083: Springer, 2011.

[12] P. A. P. Salas and B. K. Aichernig, "Automatic Test Case Generation for OCL: a Mutation Approach," *UNU-IIST Report*, 2005.

[13] M. A. Almeida, J. de Melo Bezerra, and C. M. Hirata, "Automatic generation of test cases for critical systems based on MC/DC criteria," in *2013 IEEE/AIAA 32nd Digital Avionics Systems Conference (DASC)*, ed: IEEE, 2013, pp. 7C5-1-7C5-10.

[14] K. Pohl, *Requirements engineering fundamentals: a study guide for the certified professional for requirements engineering exam-foundation level-IREB compliant*: Rocky Nook, Inc., 2016.

[15] K. Pohl, *Requirements engineering: fundamentals, principles, and techniques*: Springer Publishing Company, Incorporated, 2010.

[16] IEEE Computer Society, "Ieee recommended practice for software requirements specifications," Institute of Electrical and Electronics Engineers 0738103322, 1998.

[17] T. Pandey and S. Srivastava, "Comparative analysis of formal specification languages Z, VDM, B," *International Journal of Current Engineering and Technology E-ISSN*, pp. 2277-4106, 2015.

[18] C.-H. Liu, D. C. Kung, P. Hsia, and C.-T. Hsu, "An object-based data flow testing approach for web applications," *International Journal of Software Engineering and Knowledge Engineering*, vol. 11, pp. 157-179, 2001.

[19] R. Nilsson, J. Offutt, and J. Mellin, "Test case generation for mutation-based testing of timeliness," *Electronic Notes in Theoretical Computer Science*, vol. 164, pp. 97-114, 2006.

[20] X. Jia and H. Liu, "Rigorous and automatic testing of web applications," in *Proceedings of the 6th IASTED International Conference on Software Engineering and Applications (SEA 2002)*, 2002, pp. 280-285.

[21] W. Zheng, "Model-Based Software Component Testing," PhD, School of Electrical, Electronic and Computer Engineering, The University of Western Australia 2012.
[22] S. S. Alhir, *Learning Uml*: O'Reilly Media, Inc., July 2003.
[23] G. Carvalho, D. Falcão, F. Barros, A. Sampaio, A. Mota, L. Motta, *et al.*, "NAT2TESTSCR: Test case generation from natural language requirements based on SCR specifications," *Science of Computer Programming*, vol. 95, pp. 275-297, 2014.
[24] P. A. Lindsay, S. Kromodimoeljo, P. A. Strooper, and M. Almorsy, "Automation of Test Case Generation from Behavior Tree Requirements Models," in *2015 24TH AUSTRALASIAN SOFTWARE ENGINEERING CONFERENCE (ASWEC 2015)*, ed. 345 E 47TH ST, NEW YORK, NY 10017 USA: IEEE, 2015, pp. 118-127.
[25] R. Elghondakly, S. Moussa, and N. Badr, "Waterfall and agile requirements-based model for automated test cases generation," in *2015 IEEE Seventh International Conference on Intelligent Computing and Information Systems (ICICIS)*, ed: IEEE, 2015, pp. 607-612.
[26] R. Venkatesh, U. Shrotri, A. Zare, and S. Agrawal, "Cost-effective functional testing of reactive software," in *Evaluation of Novel Approaches to Software Engineering (ENASE), 2015 International Conference on*, 2015, pp. 67-77.
[27] E. Sarmiento, J. C. S. d. P. Leite, and E. Almentero, "C&L: Generating Model Based Test Cases from Natural Language Requirements Descriptions," in *2014 IEEE 1st International Workshop on Requirements Engineering and Testing (RET)*, ed: IEEE, 2014, pp. 32-38.
[28] G. Carvalho, A. Cavalcanti, and A. Sampaio, "Modelling timed reactive systems from natural-language requirements," *Formal Aspects of Computing*, vol. 28, pp. 725-765, 2016.
[29] B. K. Aichernig, K. Hormaier, F. Lorber, D. Nickovic, R. Schlick, D. Simoneau, *et al.*, "Integration of Requirements Engineering and Test-Case Generation via OSLC," in *2014 14th International Conference on Quality Software*, ed: IEEE, 2014, pp. 117-126.
[30] Á. Carrera, C. A. Iglesias, and M. Garijo, "Beast methodology: An agile testing methodology for multi-agent systems based on behaviour driven development," *Information Systems Frontiers*, vol. 16, pp. 169-182, 2014.
[31] R. Gao, J. S. Eo, W. E. Wong, X. Gao, and S.-Y. Lee, "An empirical study of requirements-based test generation on an automobile control system," in *Proceedings of the 29th Annual ACM Symposium on Applied Computing - SAC '14*, ed. New York, New York, USA: ACM Press, 2014, pp. 1094-1099.
[32] T. Yue, S. Ali, and M. Zhang, "RTCM: a natural language based, automated, and practical test case generation framework," in *Proceedings of the 2015 International Symposium on Software Testing and Analysis - ISSTA 2015*, ed. New York, New York, USA: ACM Press, 2015, pp. 397-408.
[33] R. Chatterjee and K. Johari, "A prolific approach for automated generation of test cases from informal requirements," *ACM SIGSOFT Software Engineering Notes*, vol. 35, pp. 1-11, 2010.
[34] D. L. Barbosa, H. S. Lima, P. D. L. Machado, J. C. A. Figueiredo, M. A. Jucá, and W. L. Andrade, "Automating functional testing of components from UML specifications," *International Journal of Software Engineering and Knowledge Engineering*, vol. 17, pp. 339-358, 2007.
[35] J. Gutiérrez, G. Aragón, M. Mejías, F. J. D. Mayo, and C. M. R. Cutilla, "Automatic test case generation from functional requirements in NDT," in *International Conference on Web Engineering*, 2012, pp. 176-185.
[36] M. Zhang, T. Yue, S. Ali, H. Zhang, and J. Wu, "A Systematic Approach to Automatically Derive Test Cases from Use Cases Specified in Restricted Natural Languages," in *Lncs* vol. 8769, ed: Springer, 2014, pp. 142-157.
[37] R. Ibrahim, M. Z. Saringat, N. Ibrahim, and N. Ismail, "An automatic tool for generating test cases from the system's requirements," *CIT 2007: 7th IEEE International Conference on Computer and Information Technology*, pp. 861-866, 2007.
[38] A. Gantait, "Test Case Generation and Prioritization from UML Models," in *2011 Second International Conference on Emerging Applications of Information Technology*, ed: IEEE, 2011, pp. 345-350.
[39] J. Gutiérrez, M. Escalona, and M. Mejías, "A Model-Driven approach for functional test case generation," *Journal of Systems and Software*, vol. 109, pp. 214-228, 2015.
[40] M. F. Granda, N. Condori-Fernandez, T. E. J. J. Vos, and O. Pastor, "Towards the automated generation of abstract test cases from requirements models," in *2014 IEEE 1st International Workshop on Requirements Engineering and Testing (RET)*, ed: IEEE, 2014, pp. 39-46.
[41] T. Straszak and M. Śmiałek, "Model-driven acceptance test automation based on use cases," *Computer Science and Information Systems*, vol. 12, pp. 707-728, 2015.
[42] C. Wang, F. Pastore, A. Goknil, L. Briand, and Z. Iqbal, "Automatic generation of system test cases from use case specifications," in *Proceedings of the 2015 International Symposium on Software Testing and Analysis - ISSTA 2015*, ed. New York, New York, USA: ACM Press, 2015, pp. 385-396.