OPEN ACCESS Journal of Software Engineering

ISSN 1819-4311
DOI: 10.3923/jse.2017.

Review Article
A Meta-Model for Model-Based Testing Technique: A Review

[1]Atifi Meriem and [2]Marzak Abdelaziz

[1]Faculty of Sciences Ben M'sik, University Hassan II, Laboratory of Information Technology and Modeling, 33 Street Youssef Mazdaghi, FLR 2, Casablanca, Morocco
[2]Faculty of Sciences Ben M'sik, University Hassan II, Laboratory of Information Technology and Modeling, Street Driss El Harti, B.P 7955, Sidi Othmane, Casablanca, Morocco

Abstract

In software testing field, model-based testing technique is a prominent validation technique that assures the high performance and evaluation of the systems. It is proposed to replace the manual way by generating automatically test cases from requirements. From the fact to its high practical relevance, several approaches that support model-based testing technique have been proposed in academic and industrial research. Although, most of these approaches differ on several important points like the specification paradigm, the model used to design requirements but share common principal, common challenges and common characteristic that are inherited from the fundamental concepts of model-based testing technique. In this context, proposed through this study a meta-model that represents model-based testing concepts and characteristics in order to define the structure and entities that must take any new model-based testing approach.

Key words: Model-based testing, software testing, meta-model, IT systems, specification paradigm

Received: Accepted: Published:

Citation: Atifi Meriem and Marzak Abdelaziz, 2017. A meta-model for model-based testing technique: A review. J. Software Eng., CC: CC-CC.

Corresponding Author: Atifi Meriem, Faculty of Sciences Ben M'sik, University Hassan II, Laboratory of Information Technology and Modeling, 33 Street Youssef Mazdaghi, FLR 2, Casablanca, Morocco Tel: +212 0 6 56 44 75 00

Competing Interest: The authors have declared that no competing interest exists.

Data Availability: All relevant data are within the paper and its supporting information files.

INTRODUCTION

Recently the testing of IT systems has emerged as a major area of research in software engineering. It has become clear that the result of this activity has tremendous impact on the quality and usefulness of the ultimate product and on the efficiency and manageability of its development. Due to the increase of IT applications complexity, it becomes increasingly necessary to implement appropriate validation and verification techniques to validate IT systems. Testing is one of the most widespread techniques used to verify and validate the proper functioning of IT system and its conformity with the initial specifications. It is also considered as the most expensive phase in the software development life cycle (SDLC). Several test techniques or methods are, thus, used in order to optimize this phase. Model-based testing (MBT) is one of these techniques, which aims to automatically generating tests from a model, describing certain behaviours of the system under test (SUT). In this context, several studies and approaches have been proposed within the scope of MBT, including a study that was done by Blom *et al.*[1]. This study was provided in order to investigate the dependency between different quality factors of test suites, generated from a formal model and the strategy or the approach used for their generation under realistic industrial conditions. The main findings of this study are model-based testing discovered faults also in previously well tested parts of the application and using (model) coverage criteria to generate test suites allow a good trade-off between fault-detection capability and test suite size, for moderate-to-large size test suites, which outperforms that of randomly generated test suites. In the same scope, Lindvall *et al.*[2,3] have proposed a new approach that combines metamorphic testing principles and model-based testing technique to perform testing of NASA's Data Access Toolkit (DAT) system, more especially to test a subsequent release of DAT when there is a new version. The main purpose of this approach was to facilitate rapid testing when a new version is delivered. Another approach considered in Graf-Brill and Hermanns[4] study for testing asynchronous communicating systems by using model-based testing. It is a practical approach that main to deriving test cases directly from an input-output transitions system, it can be applied to generate a test suite offline or to construct a test case online, incrementally during test execution. Also, Heam *et al.*[5] have presented through their study a new approach that focus on security protocols testing. This approach relies on the use of mutations of formal models of security protocols to generate abstract test cases. Most of the MBT approaches proposed in academic or industrial research have two common aspects, requirements and generating tests from abstract model and they differ on several important points. In this context, it proposed through this article a meta-model that allows the characterization of different model-based testing approaches and that illustrated the common concepts of MBT approaches in addition to allowing validating any proposed approach.

Meta-models and model-based testing in literature: A model is an abstraction of a system and a meta-model is an abstract description of a model. Meta-model is a formal definition of a model, it define the constructs and rules needed to create semantic models. A meta-model facilitates the reasoning on model structure, model semantics and model use. According to meta-object facility standard (MOF), a meta-model defines the structure to have by any model of this meta-model. In other words, any model must respect the structure defined by its meta-model[6]. The meta-modeling is the activity that produces, among other things, meta-models that reflect the static structure of the models. It consists on specifying the concepts and the links between these concepts. Meta-modeling is widely used in information systems engineering and particularly in models and methods engineering. Its played a central role in all model-driven approaches like MDA, MDD and MDSE[7]. In the literature, several research works have been carried out in this context and several meta-models have been proposed, each of which was concerned with specific fields. In Zachariadis *et al.*[8] study, a meta-model for engineering adaptable mobile systems was proposed. It was a local component meta-model for mobile adaptive systems, implemented in the SATIN (system adaptation targeting integrated networks) middleware system. The SATIN was used to create and offer a number of applications that show different adaptation aspects. He *et al.*[9] have proposed a meta-model in order to define a graphical notation for graphical modeling language. They have used three essential elements to define the notation of a modeling language: Basic figures which are collected from UML family and layouts as flow layout, border layout, decoration layout, role name layout, vector graph layout and extendable layout. Brunette *et al.*[10] have defined a meta-model to design polychronous systems. As the name suggests, polychronous systems use multiple clocks, which means that signals do not need to be present in all instants. In this context, Brunette and Talpin[10] have developed a meta-model and an open-source

design environment for the synchronous language SIGNAL in the GME (generic modeling environment) and eclipse frameworks. They have also shown how this meta-model can be easily extended to provide designers with adequate concepts for the design of both control oriented and avionics systems. Van Dongen *et al.*[11] in their work, focused on the use of meta-models in the context of process aware information systems (PAIS). They have proposed a meta-model for event logs generated by PAIS and then, they have defined a formal XML definition (MXML) for event logs, to support the proposed meta-model. In the same context, Gholizadeh and Azgomi[12] have proposed a meta-modeling approach to define a multi-formalism modeling framework for Petri nets. They have defined the meta-modeling structure and its implementation using XML (extensible markup language) as the base language for the all definition. Although most of the literatures represented above are dedicated to subjects other than software testing, a fundamental lack in meta-modeling related to testing approaches was seen. For this reason, this paper proposed a meta-modeling solution for model-based testing (MBT) technique that contains a set of concepts needed to validate any MBT approaches.

On the other hand, various works in the literature use the term model-based testing. Some of them present new approaches that support the MBT technique. For example, Dalal *et al.*[13] have proposed an MBT approach that consists on using a data model to generate tests. It depends on three key elements. The first one was the notation used to describe the software behaviour as a model. The authors used for this aim the AETGSpec notation which was part of the AETG TM software system and was useful to describe the data-flow only. The second element was the test generation algorithm in which he used the AETG software system to generate combinations of input values. Finally, the tools aimed to generate the supporting infrastructure for the tests (including the expected outputs). Hence, the major strength of Dalal's approach was the tight coupling of between the tests to and requirements and also the ability to regenerate tests rapidly in response to changes. The negative aspects of this approach were, however, the fact that only data flow was covered and the control flow was not considered in addition to the lack of oracle to store the test cases. Moreover, only systems of with low complexity are concerned by this approach. Also Deng *et al.*[14] have proposed an MBT approach for testing and maintenance based on semantic software development model. In this approach, the authors use the diagrams during the software development life cycle to generate test scenarios, for example, they use the case diagrams used to describe the software system requirements. They also use the class diagrams describing the static data structure, in addition to the state-chart diagrams used to describing the state transitions of objects. For maintenance, Dong *et al.*[14] use the software maintenance model which consists of on a set of fix, bug and modification objects that link the information generated from modifications. Krenn *et al.*[15] have proposed a new testing approach named model-based mutation testing (MBMT). The proposal was a particular instance of fault based testing and it uses a model of the system under test to generate test cases. The system model was described using UML diagrams, namely, UML state charts, class diagrams and instance diagrams. The major strength of MBMT methodology was the possibility to demonstrate the absence of certain faults. It was considered as one of the most powerful test case generation approaches, that allows excellent fault detection capability, the generated test cases are both effective in finding faults and efficient when executed. Pretschner *et al.*[16] in their work, have compared the model-based test suites derived automatically or manually with the handcrafted test suites that were directly derived from the requirements in term of quality and then they have shown that the first test suites detect significantly more requirements and programming errors than the seconds. Moreover, they showed the method on which the model, the implementation coverage and failure detection are related. The major contribution of this work was the provision of numbers indicating the usefulness of explicit behaviour models in testing. The proposal also stimulates the discussion on the usefulness of automation and the use of structural criteria as like C/D coverage in model-based testing.

Model-based-testing and models

Model-based testing process: The model-based testing solution consists on producing test cases from SUT (system under test) model and/or its environment model by following a process composed of five phases[17] (Fig. 1):

- Requirements management
- Modelling of an abstract test model dedicated to test of the system
- Generation of abstract test cases from test model
- Concretization of abstract test cases to concrete test cases that can be executed on the system under test
- Execution of concrete test cases on the system and the constitution of their verdict

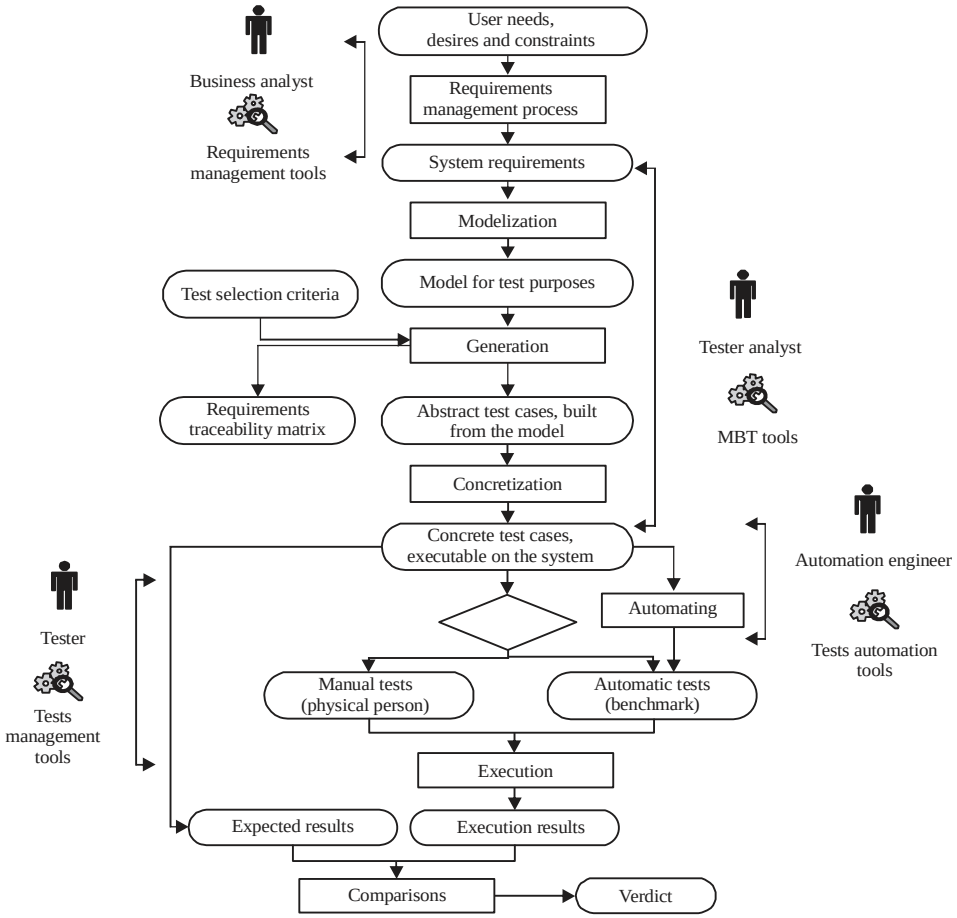

Fig. 1: Model-based testing general process[17]

In model based testing (MBT), the requirements specify the expected behaviour of system under test in response to particular conditions. Requirements management step consists on collecting customer needs, desires and constraints and the management and classification of these later as requirements. The second step of the MBT serves to define an abstract model of the system under test. This model was an abstract view of the system under test, defined in a particular formalism according to the specification paradigm[18-20]. In the next step, the abstract model was used by a test generator which, by means of selection criteria, allows generating the abstract tests cases and traceability matrix that illustrated the link between the tests and the model's elements. In the concretization step, the abstract tests cases are transformed to concrete tests cases to link the elements of the abstract model with the concrete elements of the system. This step was still generally manual and requires more expertise. Finally, the concrete tests cases can be run on the system under test to get the verdict in order to decide if the system respects/or not the model that have been made.

Test case in model-based testing process: In the MBT process, a test case can take several states by when passing from the requirement management, modelling, generation and then concretization and execution (Fig. 2).

Specification paradigms in model-based testing process: Generally, the testing of IT systems begins with the perception of requirements specification, the result of the requirements phase in the system life cycle. Model-based testing is a testing technique that uses a specification described by a formal model as a reference to generate tests cases. It was a test technique based on behavior models. The behavioral models represent the expected behavior of a system with respect to external stimulations. However, when modeling the systems, the choice of modeling formalism depends on the aspect or the paradigm of the system to be modeled. Thus, some formalism favor a representation oriented towards the system and their

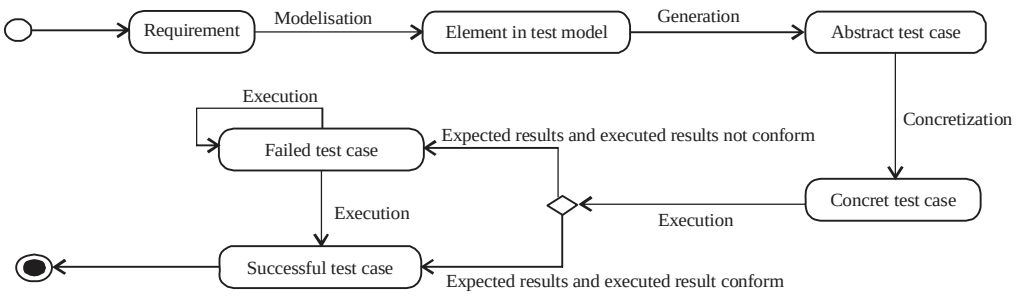

Fig. 2: State chart diagram showing the different states of test case in MBT process

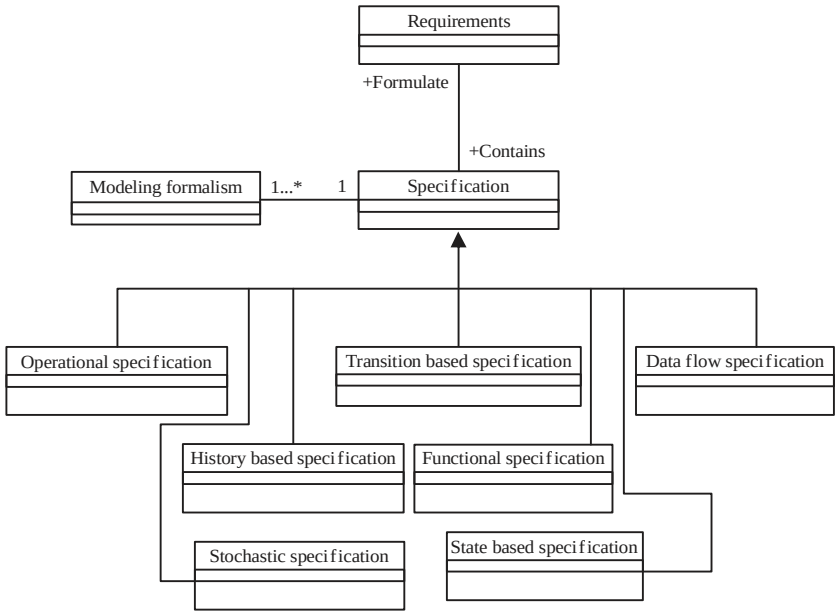

Fig. 3: Specification paradigms

evolution, while others favor a representation oriented events that makes possible the evolvement of the system. The starting point of this analysis was the Lamsweerde's classification[19] of formal specifications into different paradigms and Utting's taxonomy[20] of model-based testing approaches. According to their results, there were several specification paradigms and for each specification paradigm there are several modeling formalisms to represent it (Fig. 3).

Operational specification: Operational specification was the aspect or the paradigm that represents behaviour of a system as a collection of executable processes running in parallel. Process algebras[21] such as CSS, CSP or ASP and Petri nets[22] are notations used to model operational specifications.

Transition based specification: Transition specification is the paradigm that represents the system states and the transitions between different states. State-charts[23], UML (Unified Modeling Language) State Machines and RSML (requirements state machine language)[24] are graphic notations used to model transition based specifications.

Data flow specification: Data flow specification is the paradigm that represents the system data, it concentrates on the data rather than the control flow. Lustre and the block diagrams of Matlab Simulink, was often used to model continuous systems.

History based specification: History based specification is the paradigm that specifies the system by describing its behaviours per unit of time. The system was modeled from

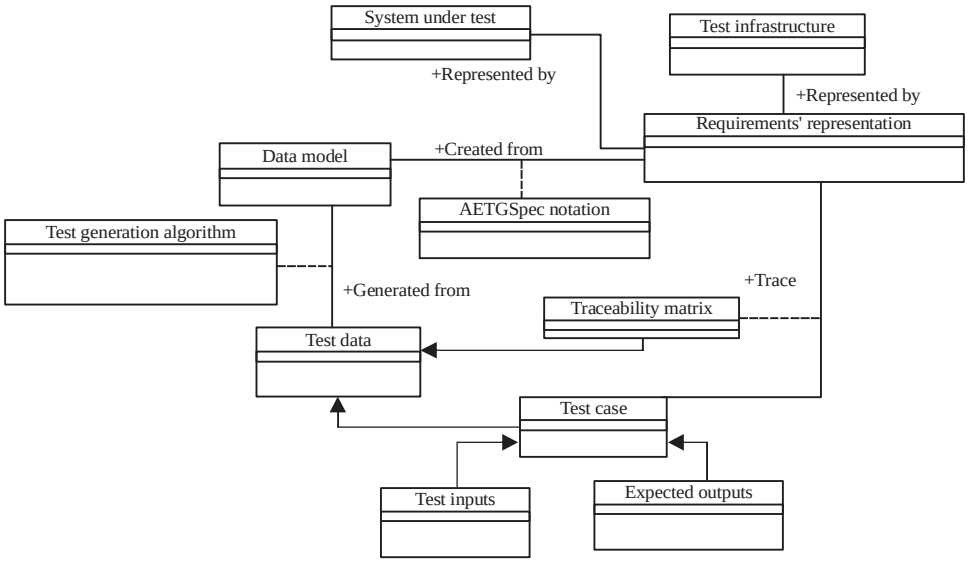

Fig. 4: Model represents Dalal's approach

temporal logical expressions on the different states of the system, past, present and/or future. In such specification, time can be linear or branching. Time structures can be discrete, dense or continuous. The properties may refer to time points, time intervals or both[19].

Functional specification: Functional specification is the paradigm that specifies a system as a structured collection of mathematical functions and that models it from these functions.

Stochastic specification: Stochastic specification is the paradigm that represents events and input data of system as probabilistic model. Markov's chains, was a notation used to model such a specification.

Model for Dalal's approach: In Dalal's approach[13], there were three key elements for model-based testing: The test data model, the test generation algorithm and the tools that generate supporting infrastructure for the tests. Test data model is created based on requirements, representation and by using a specification notation called AETGSpec, which is part of the AETGTM software system. Test data as test inputs, expected outputs and traceability matrix are generated from test data model by using a test generation algorithm. Dalal's approach uses the AETG software system to generate combinations of input values. A model that represents the model based testing approach proposed by Dalal *et al.*[13] was showed in (Fig. 4).

Model for Deng's approach: Deng has used in his approach[14] a model called semantic software development model (SSDM) that combines all information generated during the software development life cycle. It covers all phases of the software development process viz: requirements, design, implementation, testing and maintenance. The software requirements and design are described by functional and behavioural models. For software implementation, Deng *et al.*[14] has used models to design the source codes and their measurements with a set of objects. The software measurements model computes the software metrics for each source code file. For software testing, Deng *et al.*[14] has used the test scenarios, test cases and testing processes. At the end, Deng *et al.*[14] has used a set of fix, bug and modification objects which record the information generated from modifications of the software maintenance model. To generate test cases, Deng *et al.*[14] has used two categories of strategies viz: Test cases generation strategies namely: All-branch testing, boundary testing and faulty testing. In addition to the regression test selection strategies that improve the efficiency of software maintenance. A model representing the model based testing approach proposed by Deng *et al.*[14] was given in (Fig. 5).

Model for Aichernig's approach: Aichernig, in his promising model-based mutation testing (MBMT) approach uses three

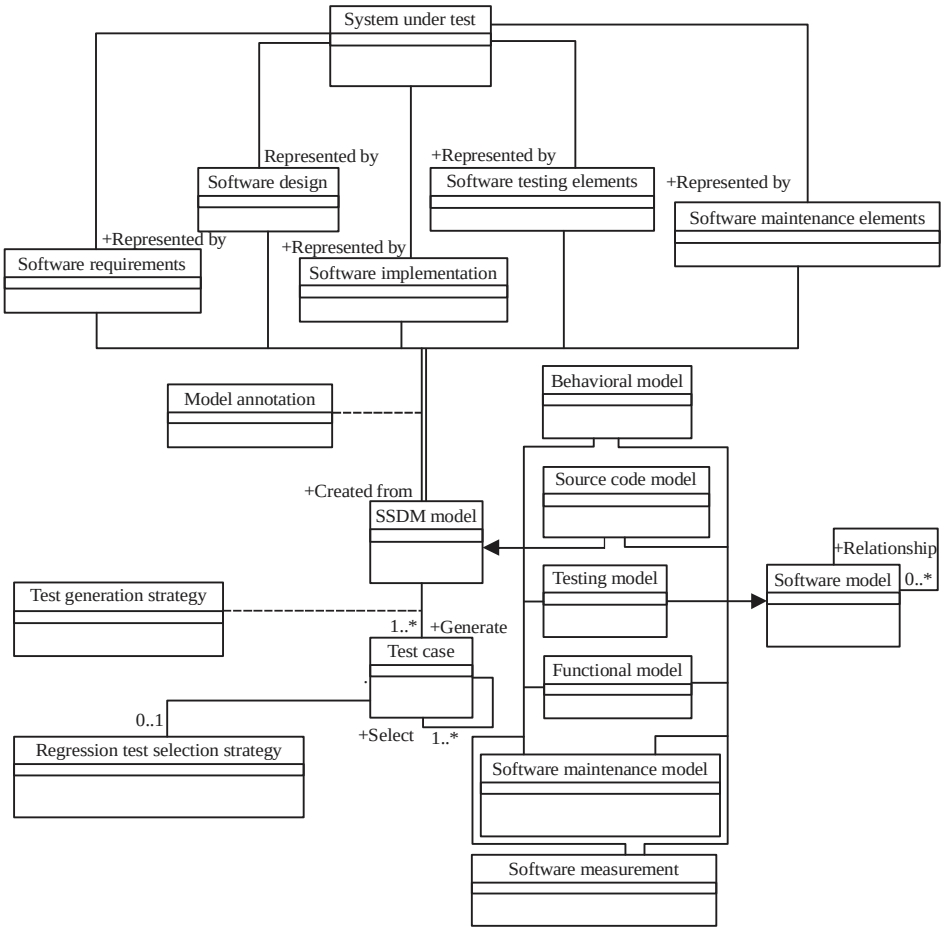

Fig. 5: Model represents Deng's approach

different test case generation strategies: Random strategy, mutation strategy and combined random and mutation strategy[15]. A model representing the model based testing approach proposed by Aichernig (Fig. 6).

Comparison between models structure of MBT approaches: By comparing the above approaches and after analyzing the structures of the proposed models for Dalal's, Deng's and Aichernig's approaches, its can drawed the following conclusions:

- All MBT approaches are designed to test software systems
- All MBT approaches use a model to generate test cases
- Each MBT approach is characterized by the model that represents the system, testing type, testing level, software domain, the generation technique of test cases and the execution technique
- Entry point or the input of each MBT approach is the requirements that represent the system under test or its environment
- Each approach has a specific test generation strategy

Proposed model and meta-model for model-based testing technique: Based on the comparison between some MBT approaches and MBT technique description, following entities could be defined as:

- System under test
- Requirements of the system under test and its environment that may be functional requirements or non-functional requirements
- Software domain that define the scope of the approach and when it can be used
- Test model used to describe the software, it may be a behavior model that describe the software behavior or a structure model that describe software structure

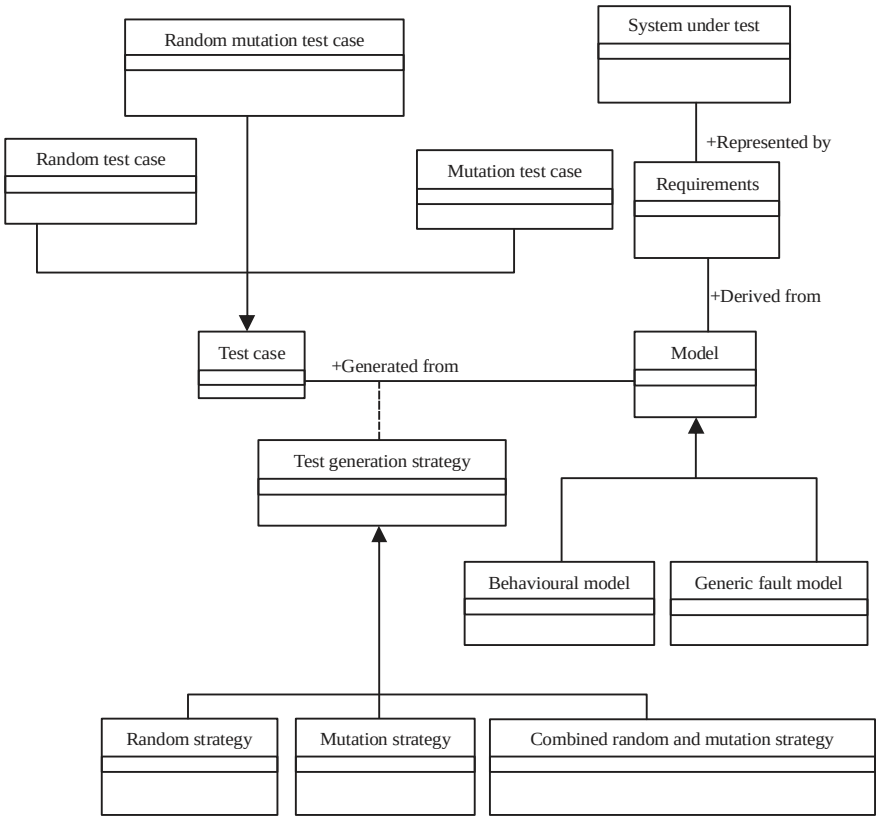

Fig. 6: Model represents Aichernig's approach

- Test cases that can take two states abstract or concrete
- Traceability matrix
- Test selection criteria that allows to generate a set of test cases sharing common properties
- Generation algorithm that allows to automatically generating a series of test cases
- Modeling notation

A model representing the model based testing technique is represented in Fig. 7 and meta-model representing the model based testing technique represented in Fig. 8.

DISCUSSION

From the fact that model-based testing has emerged as a major research area in academic and industrial, a large number of publications and new approaches are produced in this field. But there is currently a lack of a unifying conceptual framework or model that illustrates the common concepts of model-based testing and provides the essential characteristics of the various MBT approaches. Most of the previous studies or publications give new MBT approaches. For example Graf-Brill and Hermanns[4] approach that use model-based testing to test asynchronous communicating systems. Also, Wang approach[25] that use also model based testing to validate quorum-based systems implemented using the Gorums library through a new MBT approach. On the other hand, some studies give the publications covering supporting techniques for modelling and test generation[26], integration into industrial practice[27], taxonomies[20], industrial evaluations[1], surveys and classification. For example Utting *et al.*[20] have provided a taxonomy of model based testing in which the principal aspects of MBT approaches are covered. This study helps to understanding the characteristics, similarities and differences of those approaches. Due to the lack of a unifying conceptual framework or model of model based testing the main purpose of this article is to provide a meta-model for model based testing technique in order to present the common concepts of model-based testing and provides the essential characteristics of the various MBT approaches. This study gives a new support that can be beneficial and will help researchers who want to propose new MBT approaches.

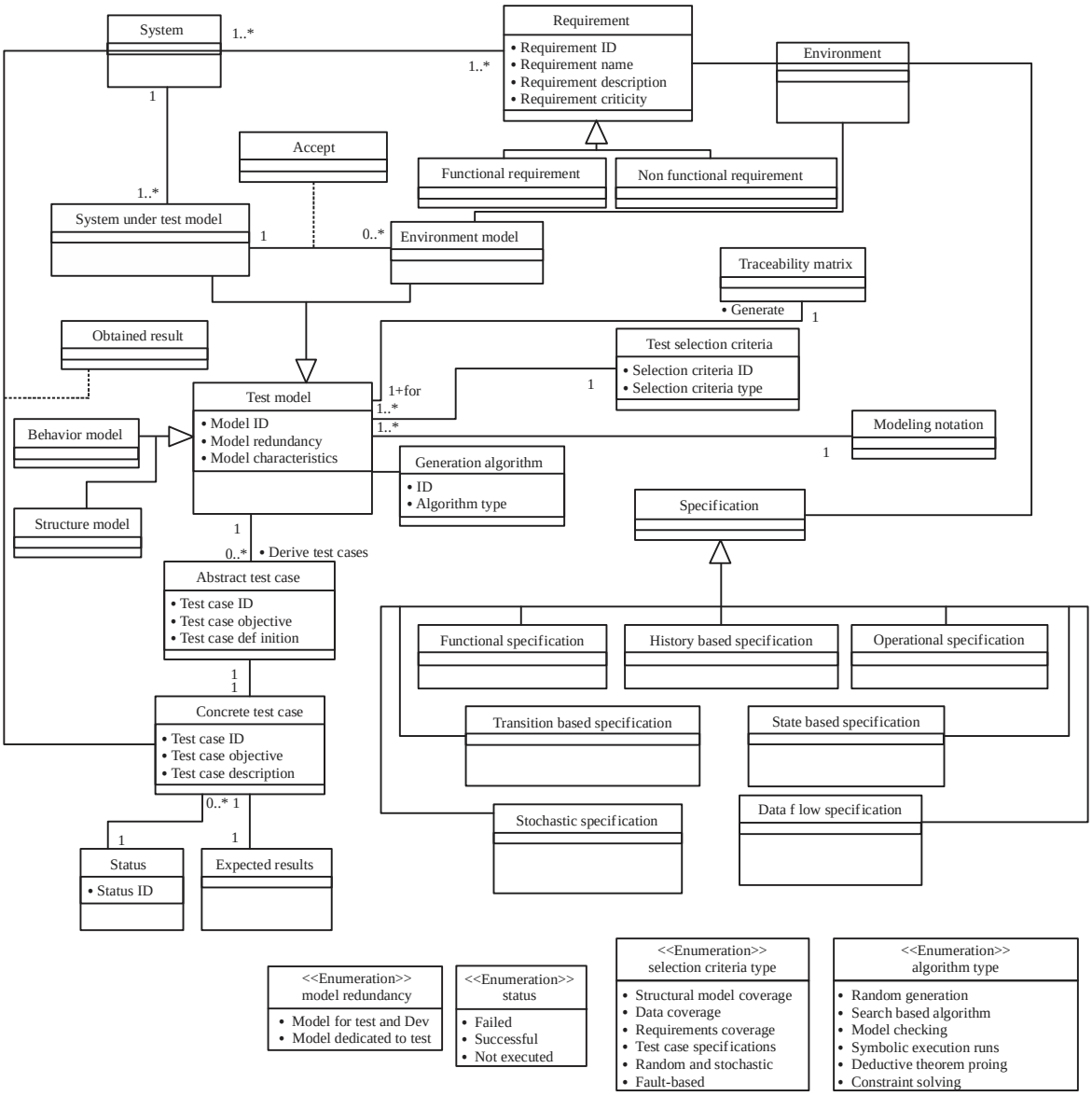

Fig. 7: Model represents model-based-testing technique

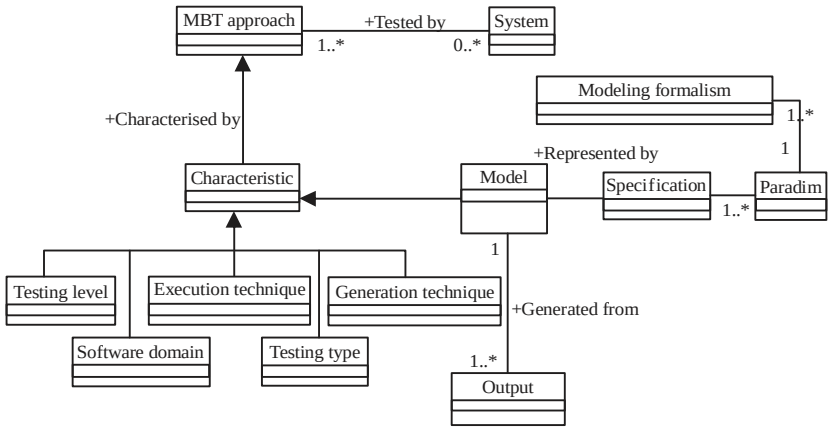

Fig. 8: Meta-model for model-based-testing technique

CONCLUSION

In this paper, proposed a generic meta-model for model-based technique. This meta-model will provide an active support in potentially useful knowledge during proposing any new approach based on the model-based testing technique. In software testing technologies such as model-based testing approaches, each approach has specific concept and characteristics. Through this work proposed a meta-model that represents the general concept and the characteristics of the model-based testing technique, this meta-model can be beneficial and will help researchers who want to propose new MBT approaches. We are currently working on the proposal of a meta-model that represents the Risk-based-testing technique in order to propose a new approach that combines MBT (model-based testing) and RBT (risk-based testing) to overcome some model based testing limitations and make some case studies by applying the novel testing approach to obtain empirical results and compare our approach over existing approaches.

SIGNIFICANCE STATEMENT

This study provides a unifying conceptual model and meta-model for model based testing technique in order to present the common concepts of model-based testing and provides the essential characteristics of the various MBT approaches. This study gives a new support that can be beneficial and will help researchers who want to propose new MBT approaches.

ACKNOWLEDGMENT

I would like to express here the very thanks to my dissertation advisor, Prof. Dr. Marzak Abdelaziz, University Hassan II, who provided me the opportunity to do such a research in his laboratory.

REFERENCES

1. Blom, J., B. Jonsson and S.O. Nystrom, 2016. Industrial evaluation of test suite generation strategies for model-based testing. Proceedings of the IEEE 9th International Conference on Software Testing, Verification and Validation Workshops, April 11-15, 2016, Chicago, IL, USA., pp: 209-218.
2. Lindvall, M., D. Ganesan, R. Ardal and R.E. Wiegand, 2015. Metamorphic model-based testing applied on NASA DAT: An experience report. Proceedings of the 37th International Conference on Software Engineering-Volume 2, May 16-24, 2015, Florence, Italy, pp: 129-138.
3. Lindvall, M., D. Ganesan, S. Bjorgvinsson, K. Jonsson, H.S. Logason, F. Dietrich and R.E. Wiegand, 2016. Agile metamorphic model-based testing. Proceedings of the 1st International Workshop on Metamorphic Testing, May 14-22, 2016, Austin, Texas, pp: 26-32.
4. Graf-Brill, A. and H. Hermanns, 2017. Model-Based Testing for Asynchronous Systems. In: Critical Systems: Formal Methods and Automated Verification, Petrucci, L., C. Seceleanu and A. Cavalcanti (Eds.). Springer, Cham, ISBN: 978-3-319-67113-0, pp: 66-82.
5. Heam, P.C., F. Dadeau, R. Kheddam, G. Maatoug and M. Rusinowitch, 2016. A model-based testing approach for security protocols. Proceedings of the IEEE International Conference on Computational Science and Engineering and IEEE International Conference on Embedded and Ubiquitous Computing and 15th International Symposium on Distributed Computing and Applications for Business Engineering, August 24-26, 2016, Paris, France, pp: 553-556.
6. Object Management Group, 2006. Meta Object Facility (MOF) core specification, version 2.0. Object Management Group, Massachusetts, USA.
7. Miller, J. and J. Mukerji, 2003. MDA guide, Version 1.0.1. Technical Report, Object Management Group (OMG), Massachusetts, USA.
8. Zachariadis, S., C. Mascolo and W. Emmerich, 2006. The SATIN component system-a metamodel for engineering adaptable mobile systems. IEEE Trans. Software Eng., 32: 910-927.
9. He, X., Z. Ma, W. Shao and G. Li, 2007. A metamodel for the notation of graphical modeling languages. Proceedings of the 31st Annual International Computer Software and Applications Conference, July 24-27, 2007, Beijing, China, pp: 219-224.
10. Brunette, C., J.P. Talpin, A. Gamatie and T. Gautier, 2009. A metamodel for the design of polychronous systems. J. Logic Algebraic Program., 78: 233-259.
11. Van Dongen, B.F. and W.M. van der Aalst, 2005. A meta model for process mining data. Proceedings of the 17th Conference on Advanced Information Systems Engineering, June 13-17, 2005, FEUP, Porto, Portugal.
12. Gholizadeh, H.M. and M.A. Azgomi, 2010. A meta-model based approach for definition of a multi-formalism modeling framework. Int. J. Comput. Theory Eng., 2: 87-95.
13. Dalal, S.R., A. Jain, N. Karunanithi, J.M. Leaton and C.M. Lott *et al*., 1999. Model based testing in practice. Proceedings of the International Conference on Software Engineering, May, 1999, ACM and Bellcore, pp: 285-294.
14. Deng, D., P.Y. Sheu and T. Wang, 2004. Model-based testing and maintenance. Proceedings of the IEEE 6th International Symposium on Multimedia Software Engineering, December 13-15, 2004, Miami, FL, USA., pp: 278-285.

15. Krenn, W., R. Schlick, S. Tiran, B. Aichernig, E. Jobstl and H. Brandl, 2015. Momut: UML model-based mutation testing for UML. Proceedings of the IEEE 8th International Conference on Software Testing, Verification and Validation, April 13-17, 2015, Graz, Austria, pp: 1-8.
16. Pretschner, A., W. Prenninger, S. Wagner, C. Kuhnel, M. Baumgartner, B. Sostawa and T. Stauner, 2005. One evaluation of model-based testing and its automation. Proceedings of the 27th International Conference on Software Engineering, May 15-21, 2005, St. Louis, MO, USA., pp: 392-401.
17. Atifi, M., A. Mamouni and A. Marzak, 2017. A comparative study of software testing techniques. Proceedings of the International Conference on Networked Systems, May 17-19, 2017, Marrakech, Morocco, pp: 373-390.
18. Utting, M. and B. Legeard, 2010. Practical Model-Based Testing: A Tools Approach. Morgan Kaufmann, USA., ISBN: 9780080466484, Pages: 456.
19. Van Lamsweerde, A., 2000. Formal specification: A roadmap. Proceedings of the Conference on the Future of Software Engineering, June 4-11, 2000, Limerick, Ireland, pp: 147-159.
20. Utting, M., A. Pretschner and B. Legeard, 2012. A taxonomy of model-based testing approaches. Software Test. Verificat. Reliab., 22: 297-312.
21. Beaton, J. and W. Weijiland, 2000. Process algebra. Cambridge Tracks in Theoretical Computer Science, Volume 18, CUP.
22. Racloz, P., 1996. Introduction Aux Reseaux de Petri. In: Genie Logiciel: Principes, Methodes et Techniques, Strohmeier, A.a nd D. Buchs (Eds.)., Polytechnic and University Presses, Romandes, pp: 207-240.
23. Harel, D., 1987. Statecharts: A visual formalism for complex systems. Sci. Comput. Program., 8: 231-274.
24. Leveson, N.G., M.P.E. Heimdahl, H. Hildreth and J.D. Reese, 1994. Requirements specification for process-control systems. IEEE Trans. Software Eng., 20: 684-707.
25. Wang, R., L.M. Kristensen, H. Meling and V. Stolz, 2017. Application of model-based testing on a quorum-based distributed storage. Proceedings of the International Workshop on Petri Nets and Software Engineering, June 26-27, 2017, Zaragoza, Spain, pp: 177-196.
26. Gebizli, C.S. and H. Sozer, 2017. Automated refinement of models for model-based testing using exploratory testing. Software Qual. J., 25: 979-1005.
27. Peleska, J., 2013. Industrial-strength model-based testing-state of the art and current challenges. Proceedings of the 8th Workshop on Model-Based Testing, March 17, 2013, Rome, Italy, pp: 3-28.