A Review of Meta-Model Designed for the Model-Based Testing Technique

Harshith Desamsetti[1*], Mounika Mandapuram[2]

[1]Student, Master of Science, Computer Science, Northern Illinois University, **USA**
[2]Cognizant Technology Solutions, Teaneck, New Jersey, **USA**

[*]Corresponding Contact:
Email: harshithdesamsetti9@gmail.com

ABSTRACT

Recently, software engineering research has focused on IT system testing. Due to its practicality, academic and corporate research has presented various model-based testing methodologies. Model-based testing is a popular software testing technique that ensures system performance and evaluation. Automatic test case generation from requirements is intended to replace manual testing. Most of these approaches differ in critical areas like the specification paradigm and the model used to define requirements. Still, they share common principles, problems, and characteristics inherited from model-based testing. This paper proposes a meta-model that represents model-based testing principles and attributes to define the structure and entities of any new approach.

Key words

Software Testing, Meta-Model, Specification Paradigm, Model-based Testing

12/31/2017 **Source of Support:** None, **No Conflict of Interest:** Declared

This article is licensed under a Creative Commons Attribution-NonCommercial 4.0 International License.
Attribution-NonCommercial (CC BY-NC) license lets others remix, tweak, and build upon work non-commercially, and although the new works must also acknowledge & be non-commercial.

STATEMENT OF THE PROBLEM

It is apparent that this activity dramatically affects the final product's quality, usability, development efficiency, and manageability. As IT applications become more complex, validation and verification methods must be used to validate IT systems. Testing is a common way to test and confirm IT system functionality and conformance to basic specifications. It is also the most expensive phase of software development. Several test approaches are utilized to optimize this phase. MBT automatically generates tests using a model that describes system under test (SUT) behaviors (Brunette et al., 2009). This study examined the relationship between formal model-generated test suite quality factors and their generation strategy under genuine industrial situations. This method was meant to speed up the testing of new versions. It can produce a test suite offline or progressively build a test case online while running tests. Mutations of formal

security protocol models generate abstract test cases in this method. Most academic and industry MBT approaches share prerequisites and abstract model testing but differ in numerous fundamental ways. This article presented a meta-model to characterize model-based testing methodologies, illustrate MBT concepts, and validate any given approach (Mandapuram, 2016).

Meta-models abstractly describe models, which are abstractions of systems. Formal model definitions, or meta-models, define the structure and rules for semantic models. Meta-models aid model structure, semantics, and use of reasoning (Desamsetti, 2016). According to MOF, a meta-model defines the structure of any model of it. Meta-modeling creates static model meta-models. It defines concepts and their relationships. Meta-modeling is standard in information systems engineering, especially models and methodologies. Numerous studies and meta-models have been developed in this context, each focusing on a specific topic.

SATIN was used to construct and deliver several adaptation-based applications. They defined modeling language notation using three essential elements: Basic figures from the UML family and layouts like flow, border, decoration, role name, vector graph, and extendable. Polychronous systems use many clocks; therefore, signals are not always available. They also demonstrated how this meta-model can be extended to give designers control-oriented and avionics system design concepts. They presented a meta-model for PAIS event logs and defined an MXML definition to enable it. As the primary language for all meta-modeling structures and implementation definitions, they used XML. Most of the above literature is about subjects other than software testing; however, meta-modeling linked to testing methodologies still needs to be improved. This study proposes a meta-modeling solution for model-based testing (MBT) that includes principles essential to validate any MBT strategy.

However, other sources use model-based testing. Some offer novel MBT-supporting methods (Gebizli & Sozer, 2016). The first was model-based software behavior notation. The authors used the AETGSpec notation from the AETG TM software system to describe data flow solely. Second, he used AETG software to produce input value combinations for the test-generating process. To support the tests, the tools generated infrastructure and expected outputs. Thus, Dalal's approach's strength was its close coupling of tests to requirements and capacity to quickly regenerate tests in reaction to changes. Unfortunately, this solution just addressed data flow and did not consider control flow, and there was no Oracle to store test cases. Only simple systems are affected by this technique. By using case diagrams to express software system requirements, the authors can create test scenarios during the software development life cycle. They also employ class diagrams for static data structures and state-chart diagrams for object state transitions. The proposal was fault-based testing that generated test cases from a system model. The system model was described using UML state charts, class diagrams, and instance diagrams. The ability to prove the absence of flaws was MBMT's main strength. It was one of the most potent test case generation methods for defect detection, producing efficient and effective test cases. They also demonstrated how the model, implementation coverage, and failure detection are connected. This work's main contribution was numbers showing explicit behavior models' testing utility. The concept also raises questions about automation and structural requirements like C/D coverage in model-based testing.

MODEL-BASED-TESTING

In model-based testing, often known as MBT, the criteria outline the actions the system being tested should take in response to various stimuli (Gutlapalli, 2016b). The first step in

requirements management is to collect client wants, desires, and limitations. The next phase in requirements management is to manage and classify these elements as requirements. Determining an abstract model of the system that is being tested is the purpose of the second stage of the MBT. This model was an abstract representation of the system that was being tested. It was defined in a specific formalism following the paradigm of specification. In the subsequent stage, an abstract model was utilized by a test generator. This test generator, via selection criteria, can generate abstract test cases and a traceability matrix, highlighting the connection between the tests and the elements of the model. In the process known as "concretization," the abstract test cases are changed into their concrete counterparts to establish a connection between the abstract model's components and the system's concrete components. Performing this stage typically still involves manual labor and a higher level of knowledge (Gutlapalli, 2016a). In conclusion, the specific test cases can be executed on the system being evaluated to obtain a judgment to determine whether or not the system respects the model that has been created.

SPECIFICATION PARADIGMS

In most cases, testing of information technology systems starts with the perception of requirements definition (Zachariadis et al., 2006). Model-based testing is a testing methodology that uses a specification accompanied by a formal model as a point of reference to produce test cases. It was a method of testing that was predicated on behavior models. The behavioral models are meant to depict the behavior that a system should exhibit in response to various stimulations from the outside world (Mandapuram, 2016). This is because different modeling formalisms have different strengths and weaknesses.

MODEL-BASED TESTING SYSTEMS

Based on a comparison between various MBT approaches and the description of the MBT technique, the following entities could be characterized as:

- Requirements of the system being tested and its surroundings, which could be classified as either functional or non-functional requirements
- Software domain, which defines the range of the method and the circumstances under which it can be utilized
- Test model that is used to describe the software; this model may be a behavior model that describes the program's behavior, or it could be a structure model that describes the structure of the software.
- The matrix of traceability
- Criteria for test selection that will allow the generation of a set of test scenarios that all have common characteristics
- Test cases that are capable of taking either an abstract or a concrete state
- The notation used for modeling
- Algorithm for generation that enables the automatic generation of a set of test cases

DISCUSSION

Since model-based testing is a prominent academic and corporate research topic, many publications and innovative methodologies have been created. However, no unifying

conceptual framework or model shows model-based testing concepts and describes the primary MBT methodologies. Most studies and publications offer novel MBT methods. Graf-Brill and Hermann test asynchronous communication systems using model-based testing. Wang uses model-based testing to validate Gorums library-implemented quorum-based systems using a new MBT technique. However, other research publishes supporting approaches for modeling and test creation, industrial integration, taxonomies, industrial evaluations, surveys, and classification. Zachariadis et al. (2006) proposed a model-based testing taxonomy covering the primary MBT methodologies. This study explains the approaches' traits, similarities, and differences. This article presents a meta-model for model-based testing to present the standard concepts and essential characteristics of the various MBT approaches due to the need for a unifying conceptual framework or model. This study provides new support for researchers proposing new MBT techniques.

CONCLUSION

A general meta-model for the model-based method was proposed in this research. This meta-model will actively support any new model-based testing approach with the necessary knowledge. Software testing solutions like model-based testing have distinct concepts and properties. This study provided a meta-model representing the model-based testing technique's overall concept and properties. This meta-model can help researchers propose new MBT approaches. We are developing a meta-model of the Risk-based-testing technique to propose a new approach that combines MBT and RBT to overcome model-based testing limitations. We conduct case studies to obtain empirical results and compare our approach to existing approaches.

REFERENCES

Brunette, C., Talpin, J. P., Gamatie, A., and Gautier, T. (2009). A meta-model for the design of polychronous systems. *J. Logic Algebraic Program.*, *78*, 233-259.

Desamsetti, H. (2016). Issues with the Cloud Computing Technology. *International Research Journal of Engineering and Technology (IRJET)*, *3*(5), 321-323.

Gebizli, C. S. and Sozer, H. (2016). Automated refinement of models for model-based testing using exploratory testing. *Software Qual. J.*, *25*, 979-1005. https://doi.org/10.1007/s11219-016-9338-2

Gutlapalli, S. S. (2016a). An Examination of Nanotechnology's Role as an Integral Part of Electronics. *ABC Research Alert*, *4*(3), 21–27. https://doi.org/10.18034/ra.v4i3.651

Gutlapalli, S. S. (2016b). Commercial Applications of Blockchain and Distributed Ledger Technology. *Engineering International*, *4*(2), 89–94. https://doi.org/10.18034/ei.v4i2.653

Mandapuram, M. (2016). Applications of Blockchain and Distributed Ledger Technology (DLT) in Commercial Settings. *Asian Accounting and Auditing Advancement*, *7*(1), 50–57. Retrieved from https://4ajournal.com/article/view/76

Zachariadis, S., Mascolo, C., & Emmerich, W. (2006). The SATIN component system-a meta-model for engineering adaptable mobile systems. *IEEE Trans. Software Eng.*, *32*, 910-927. https://doi.org/10.1109/TSE.2006.115

--0--