Metamorphic Testing on the Continuum of Verification and Validation of Simulation Models

M S Raunak, Megan M Olsen

***Abstract*—Metamorphic testing has been shown to be useful in testing "non-testable" programs in many domains. Modeling & simulation is one such domain, where both verification and validation can be difficult due to lack of oracles. Although the definition of verification and validation vary slightly in modeling and simulation when compared to standard software, we show that metamorphic testing is appropriate in both aspects of ensuring that a simulation model is accurate. In this paper we expand on our five years of prior work on metamorphic testing for simulation validation to show how metamorphic testing can be used for verification as well, and how the previously defined guidelines for validation can be utilized in eliciting metamorphic relations for verification.**

***Index Terms*—Modeling and Simulation, Verification and Validation, Metamorphic Testing.**

I. Introduction

Software systems have become an integral part of every aspect of our lives. From simple computation to complex analysis, from mundane automation to intricate simulation, computational models and their executable implementations are driving many areas of scientific advancement. Ensuring the correctness and reliability of these systems is important, and done via *verification* and *validation*. Verification of software is defined as the set of activities that ensure the consistency between the *specification* of a software and its actual implementation. *Validation* of software, on the other hand, involves the set of activities that establishes how closely software meets its real requirements, i.e., the need of the users [1].

Software testing, a broad and multi-faceted term, is the primary approach used for verification. The goal of testing is to discover software faults or anomalies. Edsger Dijkstra famously pointed out, "Program testing can be used to show the presence of bugs, but never to show their absence!" Nevertheless, testing has been our primary vehicle to increase confidence that software behaves as expected under all inputs and conditions. Testing requires judiciously selecting a subset of a software's usually infinite input and configuration space, executing it with the selected elements (test cases), and then ensuring that it matches the expected output. Exhaustive testing, where all inputs and configurations are tested for correct output, is infeasible for any reasonable software system due to the infinite input and configuration space. Hence, the goal of testing is to determine how to select the test cases and know when enough testing has been done such that testing can stop.

M S Raunak is a Computer Scientist at the National Institute of Standards and Technology. E-mail: raunak@nist.gov

Megan Olsen is an Associate Professor at the Department of Computer Science, Loyola University Maryland, Baltimore, MD 21210. E-mail: mmolsen@loyola.edu

To run tests we also need a test oracle, which is a mechanism through which one knows if software is producing the correct output or behavior. Setting up a test oracle is a necessary and reasonable step for most software systems. However, for some programs developing an oracle is either extremely difficult or too expensive, and are thus often termed 'non-testable' [2]. Metamorphic Testing (MT) has been successful in testing 'non-testable' programs in many domains [3], [4] such as cryptographic algorithm implementations [5]–[7], scientific computations [8], and machine learning algorithms [9], [10]. One important domain that is particularly well-suited for metamorphic testing is Modeling and Simulation (M&S), as simulation models and their executable implementations suffer from the same test oracle problem.

Validation of simulation models differs from the standard software validation definition as it specifies how well an executable simulation model mimics the real world system that it simulates. Often a simulation model is built when the real world cannot be directly experimented with due to its complexity, cost, or safety issues. Models of protein folding, nuclear reaction, and epidemic spread are examples of such scenarios. Once the model is conceptually defined and then implemented, we need to *verify* that it does not have bugs and *validate* that it mimics the real world scenario closely enough for its purpose. Standard validation approaches primarily rely on either an expert on the modeled system to determine whether the conceptual model correctly represents the system, or comparing the output to known results. Unfortunately, either or both of these resources may be difficult to acquire at a sufficient level for any given model, leading to an oracle problem. Our prior work shows that metamorphic testing can fill this gap, and provides guidelines for its application [11].

On the other hand, the meaning of verification for simulation is largely the same as in standard software: making sure that the computer code implementing the model is internally consistent and matches the model's specification. Metamorphic testing has been shown to be useful for verification of simulations in a number of case studies [12]–[14], as it can be most natural to apply it for testing. However, there is no defined process for eliciting the metamorphic relations for verification, or determining if a metamorphic relation is for verification or validation. In this paper, we present a cohesive process for using MT on both verification and validation of simulation models, building on our prior process on using MT for simulation validation [11], [15], [16]. We propose that verification and validation of simulation models need to be considered as a continuum, and demonstrate how MT should be applied within that spectrum.

II. Related Work

Bair and Tolk argued that there is no unified frame of reference in the literature for simulation validation [17]. Nevertheless, researchers and practitioners in the M&S community discuss three different areas of validation: *conceptual validation*, *data validation*, and *operational validation* [18]. From the software engineering verification and validation (V&V) angle, the primary area of concern is the operational validation, which checks how well the model represents the real world behavior that it simulates [15]. The Handbook of Simulation presents over 75 different V&V techniques categorized in four different groups: informal, static, dynamic, and formal [19]. Informal techniques such as review, inspection, visualization, and face validation are some of the most commonly used approaches for validating simulation models. Static techniques are based upon software static verification, and some can be applied for analyzing operational validity of simulation models in a limited way. Dynamic techniques are essentially extensions of software testing techniques, and are used for verification of simulation code. Most of these verification techniques are not directly applicable for model validation, however. Some dynamic techniques that are specifically used for validation purposes, e.g. statistical techniques, rely heavily on the presence of real world data from the system under study (SUS) [18], [20]. Model validation has thus been a well-established challenge in building trustworthy and useful simulation models [21]. A consequence of this challenge is that validation efforts on specific models are often absent or not clearly reported in the literature [22].

Over the last two decades, Metamorphic Testing (MT) has grown as a dominant technique for testing systems that suffer from the ‘oracle problem’ [3], [4]. Researchers have applied it to cryptographic algorithms implementations [5]–[7], memory systems [23], Artificial Intelligent (AI) applications in autonomous systems [9], [13], [24], and driver-less cars [25] amongst many other domains. Of particular relevance to the work presented here, MT has also occupied a very interesting and important place for verification and validation of modeling and simulation. Some early work showed the presence of metamorphic relations and use of metamorphic testing in simulation models with limited applicability and a narrow focus. In one of the earliest works in this domain, Murphy and Raunak showed that MT is useful in finding bugs in simulation software [12]. Lindvall et al. presented a case study of using metamorphic testing on an autonomous robot simulation, which verified the simulation code [13]. He et al. showed the use of MT for verification of a high performance numerical simulation program, and pointed out that even though MT is a highly promising and successful technique for testing simulation, there are still many challenges [14]. Ding and Hu, for example, articulated the need for measuring and enhancing the adequacy of Metamorphic Relations (MRs) using a Monte Carlo simulation program [26]. Olsen and Raunak articulated a process for using MT for simulation model validation after noting that many researchers were only applying MT as verification in simulation, and there was no obvious approach for validation [15], [16]. They presented ideas and examples on how to modify MT to be applicable for validating both agent-based as well as discrete event simulation models. [11] presented a general framework for validating simulation models using MT and showed its applicability through a set of detailed guidelines and case studies, with the extension to verification left as future work.

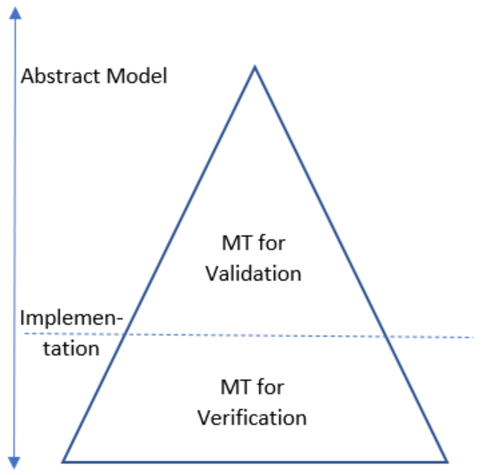

Fig. 1. Applying MT for simulation exists over a V&V Continuum.

In summary, the primary work thus far on metamorphic testing in modeling and simulation has either been examples of verification, or guidelines and case studies for validation. There has not been an overarching approach yet that encompasses both verification and validation. We have noticed that many researchers are unsure how to differentiate MR for verification vs. validation, when applying MT to simulation models. In this paper, we propose that MT can be an effective tool for both verification and validation of simulations, and propose how to identify, apply, and differentiate metamorphic relations for both purposes.

III. Our Approach

To properly define metamorphic testing for simulation, three aspects need to be determined: 1) How to differentiate between verification and validation for Metamorphic Relations (MR); 2) How to define MRs for verification; and 3) How to define MRs for validation. Our previously published guidelines provide a process for eliciting MRs for validation in agent-based and discrete event simulation models [11]. At a high level, there are two types of MRs: those that are defined by a change to a parameter value, and those that are defined by a change to the model. We provide a framework within which MRs for validation should be defined, based on the common *aspects* of these types of models. We also provide guidelines on how to define these relations and test them [11], [15], [16].

In this paper, we propose that MT should be considered as a continuum for increasing trust in simulation models and their implementations (Figure 1). The primary difference between MT used for validation and those used for verification are what information is used to define the output-pair. In the case of validation, we use the abstract model or the real world system to predict how a change in model or parameter affects

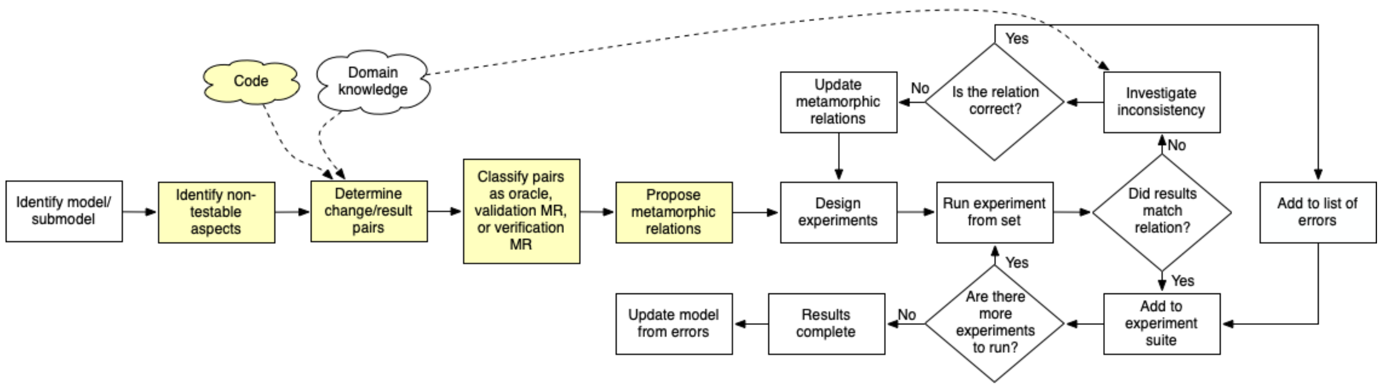
Code
Domain knowledge
Identify model/ submodel
Identify non-testable aspects
Determine change/result pairs
Classify pairs as oracle, validation MR, or verification MR
Propose metamorphic relations
Design experiments
Run experiment from set
Did results match relation?
Add to list of errors
Investigate inconsistency
Is the relation correct?
Update metamorphic relations
Add to experiment suite
Are there more experiments to run?
Results complete
Update model from errors
Yes
No

Fig. 2. The overall process for applying Metamorphic Testing for verification and validation of simulation models. Expanded from Figure 3 in [11], with additions for verification highlighted.

the change in simulation output. In the case of verification, we are using implementation details to make that prediction. The line between verification and validation for simulation is not always clear. However, we propose that if one focuses on what is informing the MR, then it becomes clear whether a MR is for verification or validation.

Verification of simulation code is generally easier to accomplish than model validation. However, not all aspects of the model's implementation may have an available oracle. It is in those cases that MT will be most valuable. The question then remains as to how one develops the verification MRs. The process of eliciting these MRs can also lead to identifying test cases for which an oracle is already present. Our earlier work details the different *aspects* of a simulation model in terms of categories that one needs to consider for discovering the MRs. As an addition to that framework, we propose the following steps to *verify* and *validate* simulation models using MT:

1) Consider what aspects of the model must be verified and validated.
2) Identify aspects from step 1 that cannot be tested using traditional verification or validation techniques. Focus on eliciting MRs from these aspects.
3) For each aspect enumerated, consider how *a change to a parameter* should affect the system. There may be more than one change/result pair for a given parameter.
4) For each aspect enumerated in step 2, consider how *a change to the model* should affect the system, for example, a small change to an algorithm. Again, more than one change/result pair is possible for an aspect.
5) Classify the change/result pairs as one of the following:
    a) If the outputs are exactly known values, it indicates that you have successfully specified an oracle for those test cases. Add them to the standard test suite.
    b) If the outputs aren't exactly known values, yet the type of change in an output can be predicted due to expert domain knowledge of the system being modeled, it is a MR for *validation*.
    c) If the outputs aren't exactly known values, yet the type of change in an output can be predicted due to the code or implementation details, it is a MR for *verification*.

When eliciting MRs, there are two sources of information to aid the process: the code or implementation plan, and knowledge of the system being modeled ("domain knowledge"). In general, the code or implementation will inform the verification MRs and the domain knowledge will inform the validation MRs. However, as one is eliciting metamorphic relations, creation of a relation for validation may cause an intuition of a relation for verification, or vice versa. Thus, the need for step 5.

As one follows step five above, it will become clear that some MRs are almost both verification and validation; thus, the continuum in Figure 1. In general, the validation MRs could be definable by a domain expert who is not involved in model development, whereas the verification MRs are more likely to be defined by a simulation developer. In practice, the simulation developer is likely to be the creator of both sets of relations. We propose that when it is unclear exactly where to draw the line between verification and validation, that the modeler use their best judgement based on their knowledge of the domain and the simulation being tested, and considering our guidelines. The key is to consider what information is informing the predicted change: the system being studied (validation), or the implementation or model design details (verification)? This consideration will lead the modeler to correctly categorize the MRs within the continuum.

At the end of this process, a test suite will exist. Figure 2 shows an overall process from identifying the model to be validated and verified to creating and testing the metamorphic relations. This figure expands on our proposed process for metamorphic testing for simulation validation [11]. For validation in particular there is a risk that an initially defined MR may actually be a misconception on the domain, and thus if a test fails it should be considered whether the error is in the model or the MR. Please see [11] for a full discussion on this type of situation. Although we leave this step in the overall process diagram, we expect that any errors resulting from verification MRs are due to errors in the model only, and thus will indicate bugs to be fixed.

IV. Case Study

To illustrate the idea of MT on the continuum of verification and validation of simulation models, we present a case study

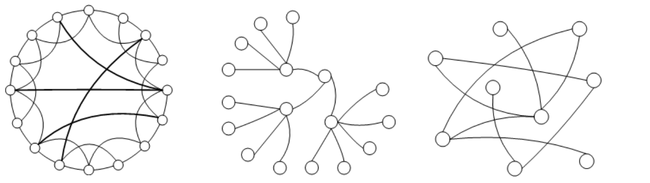

Fig. 3. The three network types modeled: a) Small-World Network, b) Scale-Free Network, c) Random Network.

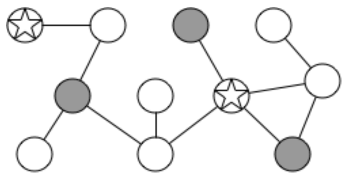

Fig. 4. An example gossip network. A node with a star is an observer, where the gossip initiates. Shaded nodes are distorters who change the bit string.

simulating how gossip can propagate in a human social network. This case study was used in [11], [16] for eliciting MRs for validation, and is based on a published model [27].

The simulation is an agent-based model where nodes represent gossipers that are connected within a network. The topology of the connections between the nodes will define how information can flow (Figure 3). The edges between the nodes can be assigned randomly (random network), using a power law distribution (scale-free network), or such that most nodes are not direct neighbors of each other but have a short distance separating them (small-world network). Gossip is defined as a bit string, and begins at randomly assigned "observers." At each step of the model, any node that previously received gossip will spread their gossip to any neighbor with whom they have a strong enough connection (Figure 4). However, there are predefined distorters within the system that modify some percentage of a bit string message before passing it on to their neighbors. Nodes that received more than one potential belief will make a decision on what gossip to believe using one of three decision rules: mode, which choose the most common message; bitwise mode, which choose the most common bit at each position in the bit string message; and random, which chooses a received message uniformly at random. Once gossip is done spreading within the system, a fitness between 0 and 1 is assigned to each node and the system overall based on how close to the truth each node believed.

In the following sections we will first define the aspects of the model that must be verified and validated, then show metamorphic relations for both verification and validation and discuss their differences.

A. *Defining Aspects of Model to Test*

As described in our approach, the first step is to determine the parameters and model aspects that need testing. The potential parameters are mean node degree, number of agents, number of observers, number of distorters, choice of network type, choice of decision rule, message length, amount of distortion, and threshold of weight for sending to neighbors.

Aspects of the model that should be considered are the creation of each type of network, the creation of the correct number of nodes, the creation of observers, the creation of distorters, process of gossip propagation for an individual node, management of the gossip propagation steps ("waves"), calculation of fitness, heterogeneous decision rule usage, mode decision rule, bitwise mode decision rule, random decision rule, belief updates, and weight updates.

Some aspects of the model already have oracles, such as testing that the networks are properly created, and that updates to fitness are correctly calculated. We can also verify that the correct number of observers and distorters are created in the system, the mean node degree is represented in the created network, the specified network type was created, the message length is correct, the distortion amount is correct, the threshold of the weight is correctly represented, and the decision rules are using the correct processes on individual decisions.

Although this model benefits the most from using MT on the validation side of the continuum, there are still aspects to gossip propagation that can be further verified using MT. In the following subsections we will examine metamorphic relations that are appropriate for each end of the continuum.

B. *Eliciting Metamorphic Relations for Validation*

In our prior work we showed the development of MRs for simulation validation in this particular model [11], [15], [16]. We proposed that MRs for validation should be defined as 1) the parameter or property to which the MR applies; 2) the modification; and 3) the expected change to the output. The final version of the MRs for this model can be seen in the top part of Table I, and full details on the process followed to elicit these relations can be read in the paper [11]. The table shows that metamorphic relations are divided into categories defined for agent-based models: A2 is agent parameters; A3 is agent topology; A4 is interactions between agents; and A6 is individual agent behaviors.

These MRs are defined based on the parameters and model aspects defined in the prior subsection, based on the expectations in the real system being studied (e.g., actual gossip spread). The A2 MRs, the third A3 MR, the A4 MR, and the first three A6 MRs result from changes in parameter values; the rest are examples of model changes. Results from testing these MRs can be seen in [11].

C. *Eliciting Metamorphic Relations for Verification*

Metamorphic relations for simulation verification can be elicited using a similar process, as described in Section III. In this particular model, we benefit the most by using MT to examine the algorithms used to propagate gossip, as well as the parameters and constants that define how the model runs during gossip propagation. To demonstrate, we provide two sets of tests: example MRs for verification that are tested via parameter change ("V1"); and example MRs for verification that are tested by model change ("V2"). Both sets can be seen in Table I. All verification MRs were elicited by examining the code.

TABLE I
METAMORPHIC RELATIONS FOR THE NETWORKED AGENT BASED MODEL (ABM) STUDYING GOSSIP PROPAGATION. VERIFICATION MRS (V1-2) ARE NEWLY PROPOSED; VALIDATION MRS (A2-6) ARE BASED ON TABLE I FROM [11].

Type	Parameter or Property	Type of change	Pseudo-oracle Answer
A2	Number of agents	increase	no change in fitness
	Number of agents	decrease	no change in fitness
	Ratio of observers	increase	overall fitness increases
	Ratio of distorters	increase	overall fitness decreases
A3	Observer placement in scale-free network	placed as hubs	fitness increases
	Distorter placement in scale-free network	placed as hubs	fitness decreases
	Mean node degree	Increases	Fitness increases
A4	Message length	increases	bitwise mode improves over standard mode
A6	Decision rule	Bitwise mode	Best result of 3 decision rules
	Decision rule	Standard mode	Outperforms random except in scale-free, only if all nodes are not distorters
	Decision rule	Standard mode	Identical to random if all nodes are distorters
	Heterogenous decision rule: Standard with bitwise	bitwise percent increases	Fitness increases
	Heterogenous decision rule: Standard with random	standard percent increases	Fitness increases except in scale-free
	Heterogenous decision rule: random with bitwise	bitwise percent increases	Fitness increases
V1	Neighbor weight threshold	increase	wave length decreases
	message length with bitwise mode	significant increase	computation time increases
V2	Mode decision rule	least common kept	overall fitness decrease
	Bitwise Mode decision	least common kept	overall fitness decrease
	Initial Message	random	No change to results

Verification MRs that are tested via changing a parameter or constant value include:

1) If the threshold of neighbor weight, which defines to whom you share your gossip, is increased, then the length of the wave should decrease.
2) If the message length significantly increases with bitwise mode, the computation time increases.

Verification MRs that are tested via model changes include:

1) If the mode decision rule is changed to take the least common belief instead of most common, overall fitness decreases.
2) If the bitwise mode decision rule is changed to take the value with the smallest score instead of largest score, the overall fitness decreases.
3) If the message is initialized randomly instead of all ones, the results should not change.

D. Comparison of Validation and Verification

All MRs for verification are defined based on knowledge of the code and implementation, as opposed to the system being studied. In each case they enable us to have better trust in the implementation of the model, as opposed to whether the model correctly represents gossip spread. As described in [11], the difference between verification and validation can be tricky when eliciting MRs for simulation models. The MRs on message length demonstrate how the same parameter may be both verified and validated with different metamorphic relations. The validation relation states how the decision rules succeed based on message length; although there are implementation details causing that difference, the difference is primarily expected due to the real system being represented by those rules. The validation relation is therefore close to the line between validation and verification, but falls on the validation side. The verification relation defines how computation time increases, which is based on implementation details; if this MR is upheld, then it increases our confidence that all of the involved parts of computing belief are implemented correctly. It does not, however, increase our confidence that it is correctly modeling the real system, as computation time has no analogy to the real world. It is therefore firmly a verification MR.

E. Changes that Result in Oracles

While developing MRs, in some cases an oracle is the answer for how the results should appear after making a change to a parameter or model. The following non-exhaustive list of tests can be performed for verification without the use of metamorphic testing, but can be elicited by following our process. In traditional verification, these can be considered as extreme case testing:

1) If there are no distorters in the network, all nodes' fitness and overall fitness should be 1.
2) If distorters modify belief by zero (i.e., the distortion causes no change to the message) then all nodes' fitness and overall fitness should be 1.
3) If a node is allowed to have no neighbors, its fitness should be zero.

The goal of verifying and validating a model is to ensure that it is trustworthy for asking questions about the system being studied. It is of course possible that some of these oracles could have been developed without the metamorphic testing process, but the system only benefits from this opportunity to ensure a more complete test suite.

V. Conclusion

In this paper we define guidelines for applying metamorphic testing within the continuum of verification and validation of simulation models. Simulation models are particularly difficult to validate, as data from the real system are not always sufficient, while some aspects of the implemented model may be difficult to verify as well. MT can therefore improve both verification and validation in simulation models. Although there are prior studies showing the use of MT for verification of specific models, there was no guidance on how to determine if a MR is verifying or validating a simulation model.

Building on the previously defined guidelines and case studies on applying MT for simulation validation [11], we provide a process for eliciting metamorphic relations for both verification and validation of simulation models, as well as how to differentiate between the two. We believe that this differentiation is particularly difficult when applying MT for simulation, and that these guidelines will assist model developers in correctly using MT across the entire verification and validation continuum. We also show how this process can aid in the development of test oracles for some parts of the system's verification. Using an agent-based model of gossip propagation, we demonstrate the use of these guidelines for applying metamorphic testing to the continuum of verification and validation.

Work on applying MT to simulation models is ongoing. In the future we plan to expand this process to additional simulation model approaches, and continue development of an overall framework that can be used to apply metamorphic testing to verification and validation of all simulation models. We also plan to investigate whether MT for verification can provide test cases for aspects of a simulation model that already have oracles, but for which the oracle test may be more complicated and therefore too time consuming to perform.

Acknowledgements

The authors thank Mohammed Farhan and Caroline Krejci of the University of Texas Arlington for their valuable feedback on the ideas presented in this paper. Any mention of commercial products in this article is for information only; it does not imply a recommendation or an endorsement by the National Institute of Standards and Technology (NIST).

References

[1] M. Pezzè and M. Young, Software Testing and Analysis: Process, Principles and Techniques. John Wiley & Sons Inc., 2008.

[2] E. J. Weyuker, "On testing non-testable programs," Computer Journal, vol. 25, no. 4, pp. 465–470, November 1982.

[3] T. Y. Chen, F.-C. Kuo, H. Liu, P.-L. Poon, D. Towey, T. H. Tse, and Z. Q. Zhou, "Metamorphic testing: A review of challenges and opportunities," ACM Computing Surveys, vol. 51, no. 1, pp. 4:1–4:27, 2018.

[4] S. Segura, G. Fraser, A. Sanchez, and A. Ruiz-Cortes, "A survey on metamorphic testing," IEEE Transactions on Software Engineering, vol. 42, no. 9, pp. 805–824, 2016.

[5] N. Mouha, M. S. Raunak, R. Kuhn, and R. Kacker, "Finding bugs in cryptographic hash function implementations," IEEE Transactions on Reliability, vol. 67, no. 3, pp. 870–884, July 2018.

[6] S. Pugh, M. S. Raunak, D. R. Kuhn, and R. Kacker, "Systematic testing of post-quantum cryptographic implementations using metamorphic testing," in 2019 IEEE/ACM 4th International Workshop on Metamorphic Testing (MET), 2019, pp. 2–8.

[7] S. Pugh, M. S. Raunak, R. Kuhan, and R. Kacker, "Systematic testing of lightweight cryptographic implementations," in 2019 Lightweight Cryptography Workshop. National Institute of Standards and Technology, Nov 2019.

[8] U. Kanewala and J. M. Bieman, "Techniques for testing scientific programs without an oracle," in Proceedings of the 5th International Workshop on Software Engineering for Computational Science and Engineering (SE-CSE '13). IEEE Press, 2013, pp. 48–57.

[9] X. Xie, J. Hob, C. Murphy, G. Kaiser, B. Xue, and T. Y. Chen, "Testing and validating machine learning classifiers by metamorphic testing," Journal of Systems and Software, vol. 84, no. 4, pp. 544–558, 2011.

[10] S. Segura, R. M. Hierons, D. Benavides, and A. Ruiz-Cortesa, "Automated metamorphic testing on the analyses of feature models," Information and Software Technology (IST), vol. 53, pp. 245–258, March 2011.

[11] M. Olsen and M. S. Raunak, "Increasing validity of simulation models through metamorphic testing," IEEE Transactions on Reliability, vol. 68, no. 1, pp. 91–108, 2019.

[12] C. Murphy, M. Raunak, A. King, S. Chen, C. Imbriano, and G. Kaiser, "On effective testing of healthcare simulation software," in Software Engineering in Health Care (SEHC '11), 2011.

[13] M. Lindvall, A. Porter, G. Magnusson, and C. Schulze, "Metamorphic model-based testing of autonomous systems," in Proceedings of the IEEE/ACM 2nd International Workshop on Metamorphic Testing (MET '17), in conjunction with ICSE, 2017.

[14] X. He, X. Wang, J. Shi, and Y. Liu, "Testing high performance numerical simulation programs: Experience, lessons learned, and open issues," in Proceedings of the 29th ACM SIGSOFT International Symposium on Software Testing and Analysis, ser. ISSTA 2020, 2020, p. 502515.

[15] M. Raunak and M. Olsen, "Simulation validation using metamorphic testing (wip)," in Proceedings of the Summer Computer Simulation Conference (SCSC '15), 2015.

[16] M. Olsen and M. Raunak, "Metamorphic validation for agent-based simulation models," in Proceedings of the Summer Computer Simulation Conference (SCSC '16), July 2016.

[17] L. J. Bair and A. Tolk, "Towards a unified theory of validation," in Proceedings of the 2013 Winter Simulation Conference: Simulation: Making Decisions in a Complex World, ser. WSC '13. Piscataway, NJ, USA: IEEE Press, 2013, pp. 1245–1256. [Online]. Available: http://dl.acm.org/citation.cfm?id=2675983.2676141

[18] R. G. Sargent, "Verifying and validating simulation models," in Proceedings of the 2010 Winter Simulation Conference (WSC '10), 2010, pp. 166–183.

[19] J. Banks, Handbook of Simulation: Principles, Methodology, Advances, Applications, and Practice. John Wiley & Sons, 1998.

[20] Sokolowski and Banks, Modeling & Simulation Fundamentals. Wiley, 2010.

[21] S. J. Taylor, A. Khan, K. L. Morse, A. Tolk, L. Yilmaz, and J. Zander, "Grand challenges on the theory of modeling and simulation," in Proceedings of the Symposium on Theory of Modeling & Simulation-DEVS Integrative M&S Symposium. Society for Computer Simulation International, 2013, p. 34.

[22] M. Raunak and M. Olsen, "A survey of validation in health care simulation studies," in Proceedings of the 2014 Winter Simulation Conference (WSC '14), 2014, pp. 4089–4090.

[23] P. C. Caizares, A. Nez, and J. de Lara, "An expert system for checking the correctness of memory systems using simulation and metamorphic testing," Expert Systems with Applications, vol. 132, pp. 44–62, 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0957417419303069

[24] J. M. Zhang, M. Harman, L. Ma, and Y. Liu, "Machine learning testing: Survey, landscapes and horizons," IEEE Transactions on Software Engineering, 2020.

[25] Z. Q. Zhou and L. Sun, "Metamorphic testing of driverless cars," Communications of the ACM, vol. 62, no. 3, p. 6167, Feb 2019.

[26] J. Ding and X.-H. Hu, "Application of metamorphic testing monitored by test adequacy in a monte carlo simulation program," Software Quality Journal, vol. 25, pp. 841–869, Sep 2017.

[27] M. E. Laidre, A. Lamb, S. Shultz, and M. Olsen, "Making sense of information in noisy networks: human communication, gossip, and distortion," Journal of Theoretical Biology, vol. 317, pp. 152–160, 2013.