Metamorphic Testing 20 Years Later: A Hands-on Introduction

Sergio Segura
Department of Computer Languages and Systems
Universidad de Sevilla
Seville, Spain
sergiosegura@us.es

Zhi Quan Zhou
Institute of Cybersecurity and Cryptology
School of Computing and Information Technology
University of Wollongong
Wollongong, NSW 2522, Australia
zhiquan@uow.edu.au

ABSTRACT

Two of the key challenges in software testing are the automated generation of test cases, and the identification of failures by checking test outputs. Both challenges are effectively addressed by metamorphic testing (MT), a software testing technique where failures are not revealed by checking an individual concrete output, but by checking the relations among the inputs and outputs of multiple executions of the software under test. Two decades after its introduction, MT is becoming a fully-fledged testing paradigm with successful applications in multiple domains including, among others, big data engineering, simulation and modeling, compilers, machine learning programs, autonomous cars and drones, and cybersecurity. This technical briefing will provide an introduction to MT from a double perspective. First, we will present the technique and the results of a novel survey outlining its main trends and lessons learned. Then, we will go deeper and present some of the successful applications of the technique, as well as challenges and opportunities on the topic. The briefing will be complemented with practical exercises on testing real web applications and APIs.

CCS CONCEPTS

• **Software and its engineering → Software testing and debugging**;

KEYWORDS

Metamorphic testing, tutorial

ACM Reference Format:
Sergio Segura and Zhi Quan Zhou. 2018. Metamorphic Testing 20 Years Later: A Hands-on Introduction. In *ICSE '18 Companion: 40th International Conference on Software Engineering , May 27-June 3, 2018, Gothenburg, Sweden.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3183440.3183468

1 METAMORPHIC TESTING (MT)

As pointed out by Dijkstra, testing can only show the presence of faults, not their absence. In many situations, successful test cases (those that did not detect any failure) were regarded as useless and, therefore, discarded or retained merely for the purpose of regression testing. In contrast to this perspective, MT observes that successful test cases carry useful, but implicit, information about the software under test (SUT), and that such information can be used to test the SUT further.

To perform MT, we need to first identify some *metamorphic relations* (MRs), which are necessary properties among the inputs and outputs of *multiple* executions of the intended program's functionality. An MR can transform existing (source) test cases into new (follow-up) test cases. If the actual outputs of these test cases violate the MR, the SUT must be faulty. Because the detection of MR violations does not require the existence of a test oracle for the individual test cases, MT is widely recognized as a mechanism that can effectively address the test oracle problem [1].

As an example, consider an algorithm f that calculates the shortest path between two nodes in an undirected graph G. Let p be a program implementing f. For any two nodes a and b in a large G, it may not be easy to verify whether the output of p is indeed a shortest path from a to b. Despite this difficulty, many MRs can be identified. For example, if we swap a and b, the length of the shortest path will remain the same, that is, $|f(G, a, b)| = |f(G, b, a)|$. Using this MR, we can run p twice, first on a source test case (G, a, b) and then on a follow-up test case (G, b, a). Instead of focusing on the correctness of each single output of p (which is difficult to decide due to the lack of an oracle), MT will check whether the relation $|p(G, a, b)| = |p(G, b, a)|$ is satisfied. If a violation is detected, p must be faulty.

For the shortest path problem, many different MRs can be identified to conduct MT. As a real-life example, Fig. 1 shows a bug of Google Maps detected using MT, where the starting and ending points were only two meters apart from each other but Google Maps returned a route of 4.3 miles [2]. This finding might indicate a weakness of Google Maps navigation when the starting or ending point was in a car park. This bug has been reported to Google, and the company is currently investigating its root cause.

In addition to addressing the oracle problem, MT is effective at fault detection, even for very intensively studied programs. For example, MT detected three previously unknown bugs in three out of seven programs in the Siemens suite [5], a benchmark widely used by the research community for the evaluation of various software testing techniques. Given that the Siemens programs are small and had been so extensively tested by different research groups in the previous two decades, the detection of the previously unknown bugs clearly demonstrates that MT complements existing software testing strategies. This is not to say that MT is necessarily superior to the other techniques, but rather testing should be conducted from *diverse* perspectives: The success of MT is due to its test case generation strategy that is based on a perspective different from

ICSE '18 Companion, May 27-June 3, 2018, Gothenburg, Sweden
© 2018 Association for Computing Machinery.
ACM ISBN 978-1-4503-5663-3/18/05…$15.00
https://doi.org/10.1145/3183440.3183468

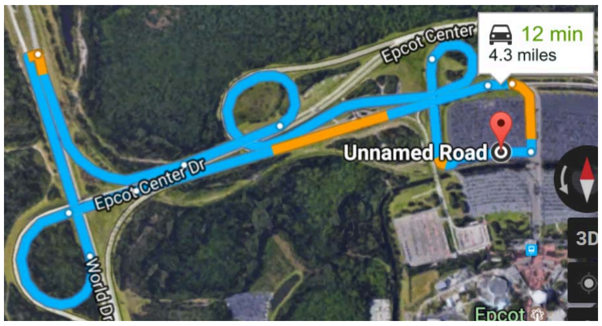

Figure 1: MT detected a real-life bug in Google Maps [2].

those used before (looking at relations among *multiple* executions of the SUT) [5].

MT has been found to be useful not only for verification but also for validation. Furthermore, it has been found to be useful for the assessment of different types of software quality characteristics, such as functional correctness, capacity, operability, user error protection, maturity, effectiveness, and context completeness [12]. MT has thus been developed into a unified framework for software verification, validation, and quality assessment [12]: In verification, MRs are derived by testers based on software specifications; in validation, MRs can be defined by users based on their expectations; and in quality assessment, different MRs can be defined by different stakeholders who are interested in different types of software quality characteristics.

In addition to addressing the oracle problem in testing, MT has also been applied to alleviate similar problems in other areas of computing. This is because other areas, such as debugging, analysis, proving, fault tolerance, and automated program repair, often assume the existence of an oracle, and therefore the integration with MT extends their applicability [3].

2 POTENTIAL INTEREST IN THE TOPIC

MT is simple in concept and straightforward in implementation. There exist strong evidence of a rapidly growing interest in this topic from both the research community and industry.

(1) *Research impact.* The growing interest in MT from the research community is reflected in the nearly 150 publications on the topic in major publication venues (the majority of which were published in recent years). The state of the art, challenges and opportunities of MT have been reviewed in two notable survey papers recently published in the *IEEE Transactions on Software Engineering* [10], and the *ACM Computing Surveys* [3].
(2) *Industrial impact.* MT is also receiving a significant attention from industry, as revealed in published results of successful applications in companies and organizations such as NASA [8] and Adobe [6]. Also, there exists strong evidence of real bugs being detected in real-world systems such as the search engines Google and Bing [12], open-source and commercial code obfuscators as well as the popular GCC and LLVM compilers [4, 7], the machine learning system RapidMiner [9], and the Web APIs of Spotify and YouTube [11].
(3) *Related events.* In 2016, Upulee Kanewala, Laura L. Pullum, Sergio Segura, Dave Towey, and Zhi Quan Zhou co-founded the first ICSE International Workshop on Metamorphic Testing (ICSE MET '16). The third edition of the workshop, organized by Laura L. Pullum, Pak Lok Poon, and Xiaoyuan Xie, is held in conjunction with the 40th International Conference on Software Engineering (2018).
(4) *Recent ACM SIGSOFT Webinar.* The authors were recently invited to give an ACM SIGSOFT Webinar on metamorphic testing [1]. A total of 316 people watched it live, and within one week, 250 people watched it on demand.

ACKNOWLEDGMENTS

This work was supported in part by the Spanish Government CICYT project BELI (project ID: TIN2015-70560-R), the Andalusian Government project COPAS (project ID: P12-TIC-1867), and a linkage grant of the Australian Research Council (project ID: LP160101691).

REFERENCES

[1] E. T. Barr, M. Harman, P. McMinn, M. Shahbaz, and S. Yoo. 2015. The Oracle Problem in Software Testing: A Survey. *IEEE Transactions on Software Engineering* 41, 5 (May 2015), 507–525. https://doi.org/10.1109/TSE.2014.2372785
[2] J. Brown, Z. Q. Zhou, and Y.-W. Chow. 2018. Metamorphic Testing of Navigation Software: A Pilot Study with Google Maps. In *Proceedings of the 51st Annual Hawaii International Conference on System Sciences (HICSS-51)*. 5687–5696. Available: http://hdl.handle.net/10125/50602.
[3] T. Y. Chen, F.-C. Kuo, H. Liu, P. L. Poon, D. Towey, T. H. Tse, and Z. Q. Zhou. 2018. Metamorphic Testing: A Review of Challenges and Opportunities. *ACM Computing Surveys* 51, 1 (2018), 4:1–4:27. https://doi.org/10.1145/3143561
[4] T. Y. Chen, F.-C. Kuo, W. Ma, W. Susilo, D. Towey, J. Voas, and Z. Q. Zhou. 2016. Metamorphic Testing for Cybersecurity. *Computer* 49, 6 (June 2016), 48–55. https://doi.org/10.1109/MC.2016.176
[5] T. Y. Chen, F.-C. Kuo, D. Towey, and Z. Q. Zhou. 2015. A Revisit of Three Studies Related to Random Testing. *Science China Information Sciences* 58, 5 (2015), 052104:1–052104:9. https://doi.org/10.1007/s11432-015-5314-x Springer-Verlag.
[6] D. C. Jarman, Z. Q. Zhou, and T. Y. Chen. 2017. Metamorphic Testing for Adobe Data Analytics Software. In *Proceedings of the IEEE/ACM 2nd International Workshop on Metamorphic Testing (MET '17), in conjunction with the 39th International Conference on Software Engineering (ICSE '17)*. 21–27. https://doi.org/10.1109/MET.2017.1
[7] V. Le, M. Afshari, and Z. Su. 2014. Compiler Validation via Equivalence Modulo Inputs. In *Proceedings of the 35th ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI '14)*. 216–226. https://doi.org/10.1145/2594291.2594334
[8] M. Lindvall, D. Ganesan, R. Ardal, and R. E. Wiegand. 2015. Metamorphic Model-Based Testing Applied on NASA DAT – An Experience Report. In *Proceedings of the IEEE/ACM 37th International Conference on Software Engineering (ICSE '15)*, Vol. 2. 129–138. https://doi.org/10.1109/ICSE.2015.348
[9] C. Murphy, K. Shen, and G. Kaiser. 2009. Using JML Runtime Assertion Checking to Automate Metamorphic Testing in Applications without Test Oracles. In *Proceedings of the 2nd International Conference on Software Testing, Verification and Validation (ICST '09)*. 436–445. https://doi.org/10.1109/ICST.2009.19
[10] S. Segura, G. Fraser, A. Sanchez, and A. Ruiz-Cortés. 2016. A Survey on Metamorphic Testing. *IEEE Transactions on Software Engineering* 42, 9 (Sept 2016), 805–824. https://doi.org/10.1109/TSE.2016.2532875
[11] S. Segura, J. A. Parejo, J. Troya, and A. Ruiz-Cortés. 2017. Metamorphic Testing of RESTful Web APIs. *IEEE Transactions on Software Engineering* (2017). https://doi.org/10.1109/TSE.2017.2764464 In press.
[12] Z. Q. Zhou, S. Xiang, and T. Y. Chen. 2016. Metamorphic Testing for Software Quality Assessment: A Study of Search Engines. *IEEE Transactions on Software Engineering* 42, 3 (March 2016), 264–284. https://doi.org/10.1109/TSE.2015.2478001

[1]https://event.on24.com/wcc/r/1451736/8B5B5925E82FC9807CF83C84834A6F3D