Looking for an MR? Try METWiki today

Xiaoyuan Xie*, Jiahao Li, Chen Wang
State Key Lab of Software Engineering,
Computer School
Wuhan University, China
{xxie, jiahao_li}@whu.edu.cn
whuwac@gmail.com

Tsong Yueh Chen
State Key Lab of Software Engineering,
Computer School, Wuhan University, China
&Department of Computer Science and
Software Engineering
Swinburne University of Technology, Australia
tychen@swin.edu.au

ABSTRACT

Metamorphic Testing (MT) has been demonstrated to successfully alleviate oracle problems in many areas, including machine learning, compilers, bioinformatics, etc. However, given a new MT task, it is still very challenging to identify enough Metamorphic Relations (MRs) which are key components of MT. Aiming at this problem, we revisited previous MT applications and realized that they could form a very valuable database. Currently there is a lack of efficient link to get testers access these historical data. Therefore, in this paper, we propose to build METWiki, an MR repository, for collection and retrieval of these MRs. By providing exploration and search facilities, testers can find their desired MRs for reuse, reference, or inference. We also illustrate three sample usages of METWiki, which show important illuminations on how MRs can be obtained in practice.

CCS Concepts

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

Keywords

metamorphic testing, metamorphic relation, MR repository, METWiki

1. INTRODUCTION

It has been almost two decades ever since Metamorphic Testing (MT) was proposed by Chen et al. [1]. MT is known as an effective alleviation to oracle problem [2,5]. Its key idea is to introduce multiple executions of different inputs for one test. Instead of checking the correctness of the output for each individual execution, MT examines the relations among these inputs and their outputs. Similar to traditional software testing, violating the relation indicates bugs in the implementation. This procedure successfully avoids having the requisite of expected output for each execution which is unavailable for programs without test oracle. Thus, it works well on those non-testable programs due to the oracle problem.

*Corresponding author.

MET'16, May 14-22, 2016, Austin, TX, USA

DOI: http://dx.doi.org/10.1145/2896971.2896976

In the past decades, MT has been widely adopted in diverse domains, such as machine learning platform [15, 16], bioinformatics applications [3, 9, 10], compilers [8, 13], SAT solvers [11], task schedulers [17], lexical analyzers [17] and etc. These studies include applications of MT for non-testable program testing, where real-life faults were found [15]. Apart from alleviating the oracle problem, MT is also successful in addressing a serious deficiency of real-life test cases for industrial projects. For example, in one of our recent projects with Company "VisyBoxes&More"[1], due to the commercial confidentiality, only 4 to 5 real-life sample input files can be released for demonstration purpose, which are far from forming an adequate test suite. However, randomly generated test cases cannot properly simulate the real-life scenarios. To solve this problem, MT were adopted for the multiplication of substantial test cases, meanwhile without any need of manual oracle generation.

The basic idea of MT is simple. It is light-weight and easy to automate. However, there is no free lunch in the world. In order to benefit from MT, people must first identify a number of Metamorphic Relations (MRs), which have been deemed as a very challenging task since it generally involves human intelligence and domain knowledge. According to our communication with various people (such as researchers in software engineering and other areas, engineers from industry, graduate/undergraduate students, etc.), we have found very obvious disparity among them: Some people can easily grasp the idea of MT and quickly come up with a number of correct MRs; while others are stuck with deriving MRs. Interestingly, this phenomenon is not always related to the subject's intelligence or expertise. Therefore, there is an emerging concern that generation of MRs can be one bottleneck that limits the development of MT. But unfortunately, to the best of our knowledge, there are only a few such studies [4, 6, 7, 18].

We agree that generating **new** MRs may not be a trivial activity. Therefore, we propose to pursue the answer from historical data that might be reused to some extent. Recently, there is a very comprehensive literature review on MT, showing that there have been over 110 applications of MT from diverse domains [12]. This convinces us that extracting and gathering these MRs are meaningful, and it will be a waste if we don't properly utilize them. Inspired by the open bug repositories, in this paper we propose to build METWiki, an MR repository to achieve the above goal.

2. MOTIVATION

2.1 Sample scenarios

Before introducing METWiki, let us consider the following scenarios.

[1]http://ssil.com.au/projects/synchronise/

Scenario A: Suppose that a tester T_1 wants to use MT to test a program which implements quick sorting algorithm. But he/she is a novice in MT. Assume that some experienced testers have adopted various MRs to test a program of insert sorting.

Scenario B: Suppose that another tester T_2 wants to use MT to test tan function. However, he/she cannot precisely recall the properties of this trigonometric function except $tan(x)=\frac{sin(x)}{cos(x)}$. And he/she does not have much experience in writing an MR. Assume that there has already existed a list of MRs for sin and cos functions derived by other people whom T_2 dose not know.

Then, one question is what is a smarter way that T_1 and T_2 should follow? Puzzle themselves to work out MRs independently, or get some hints from previous MRs of similar applications? Obviously, the latter idea sounds better. But, what can T_1 and T_2 do if they have totally no idea about the existence of those previous MRs?

2.2 Hints learned from the samples

As illustrated in Figure 1, the problems of T_1 and T_2 are due to the lack of connection with the already identified MRs.

Figure 1: Unconnected information

Actually, when people write MRs, he/she does not necessarily have to write them from scratch. Generally speaking, there could be three cases as follows.

(1) Case 1: The new SUT (Software Under Testing) can utilize exactly the same MRs from previous studies;

(2) Case 2: The new SUT can partly adopt some previous MRs for other software with only a few modifications;

(3) Case 3: Of course, the tester can create MRs for this new SUT completely from scratch.

As a consequence, an intuitive solution is to build a connection between a new MT request and historical information, which collects the existing data and exposes them to people in need. Because we target at end-users who simply want to quickly identify enough MRs to perform MT on a particular application, then academic papers may not be proper data sources. First, the studies are sporadic in many papers from various disciplines and publication venues. Collecting these papers is very time-consuming, and sometimes even inaccessible for non-academic people. Secondly, reading and quickly extracting the desired MR information from all relevant papers is never a trivial task, especially for people without much academic background. Besides, not all MR data were reported in the paper, some were in the experimental results [18].

3. METWIKI: AN MR REPOSITORY

The outline of METWiki is shown in Figure 2. In METWiki, a repository of MRs is created, upon which exploration and search facilities are provided.

3.1 Exploration of MRs

A user-friendly repository should provide quick exploration facilities. In METWiki, we propose to adopt the github-style showcases to display various application domains vividly, as shown in

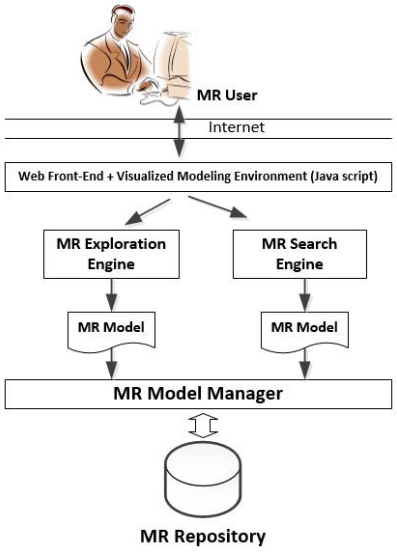


Figure 2: Outline of METWiki

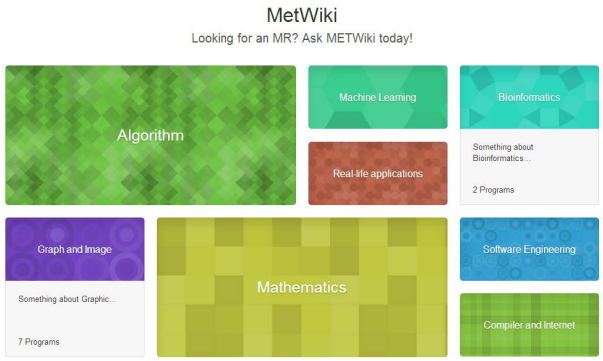


Figure 3: Showcases for application domains

Figure 3. And the size of a card reflects the corresponding amount of applications.

By clicking a particular card, user can then browse all applications belonging to this domain, as shown in Figure 4.

Figure 4 shows parts of the programs with some brief description. Each program is associated with keywords, domain names, types of input and output. By clicking the title, people can view detailed information, as shown in Figure 5. On this page, the top area describes the program, and followed by the extracted MR information from the original references, which are also given for extended reading.

3.2 Search in the repository

To quickly locate a piece of interested information, query should also be provided. In METWiki, we will allow user to search for MRs by domains, keywords, as well as input and output types.

4. HOW CAN METWIKI HELP: SOME ILLUMINATIONS

While the prototype of METWiki is almost done, we now mainly focus on the MR collection. During this process, we mine these collected data and have obtained the following observations and insights.

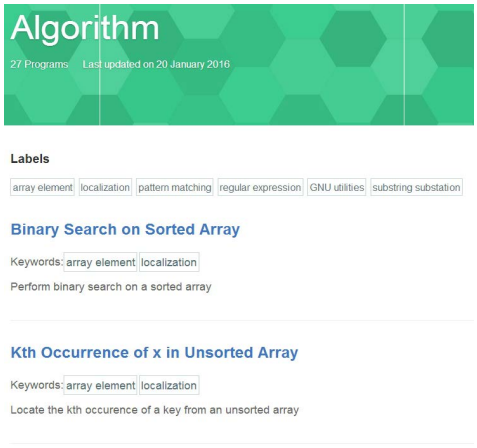


Figure 4: Exploring all programs in the selected domain

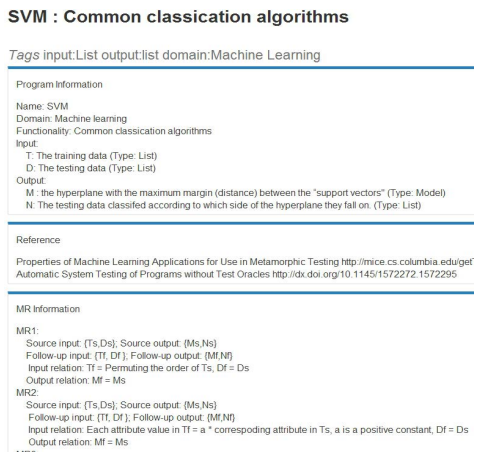


Figure 5: View details of the selected program

(1) Most of the applications are computational or algorithmic programs (by referring to the top two largest domains, there are 35 applications in Mathematics and 27 in Algorithm, which take over 50% of all current records). On one hand, this indicates much more room for exploring new applications. On the other hand, we observed that some areas with sophisticated domain knowledge may benefit from basic areas like Mathematics and Algorithm. It is commonly known that programs in these basic areas seldom appear individually. Instead, they are usually encapsulated and presented in types like libraries, services, API, etc. Many software are built by incorporating these encapsulated modules (as illustrated in Figure 6). As a consequence, MRs of these modules may also be available for the complete software. For example, in SeqMap, a bioinforamtics program whose major function is pattern matching, some MRs can be derived by referring to MRs of "pattern matching" algorithms. Therefore, though the domain knowledge and business logic in some new applications are complex, generating MRs may not be difficult.

This observation reveals one usage of METWiki: By building a repository, it provides some foundation elements to create more complex MRs. Given a new program, if the tester knows some of its basic components, he can search by "keywords" to get some MRs for these components, which could be further tailored and assembled to produce the final MRs. The procedure is illustrated in Figure 7(a).

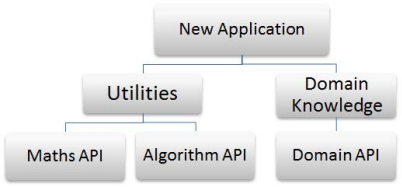
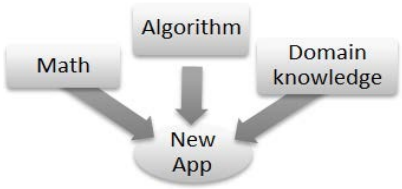
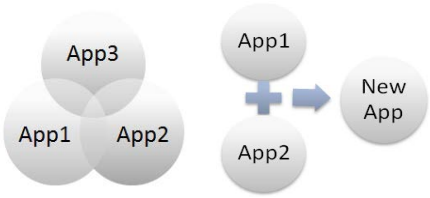


Figure 6: Hierarchy of modern software components



(a) *Usage 1*



(b) *Usage 2* (c) *Usage 3*

Figure 7: Sample usages of METWiki

(2) Applications from the same domain, especially which serve the same purposes, are very likely to share the common MRs. Let us consider several examples. In [14, 15], four MRs have been proved to be necessary properties for two different classification algorithms, kNN and NBC. And Google, LiveSearch and Yahoo, which are three famous search engines, can be tested via the same MR [19]. This observation is also held for various mathematic libraries. In [18], four mathematic libraries (namely, Apache, JDK, GSL, MATLAB) were investigated and obviously these libraries provide a large amount of algorithms with common functionalities but in different languages and different ways to invoke.

This observation reveals another usage of METWiki: Via searching by domain or keyword, a tester can view similar applications, from which common MRs can be reused. By revisiting Scenario A in Section 2.1, T_1 can be assisted via searching "sorting". As a reminder, these MRs should be reused with caution, by carefully checking its necessity in the new application. We reckon that this is an acceptable cost since human involvement is always inevitable in MR generation. As compared with manual verification on some machine-generated MRs [18], this checking shall be easier since these historical MRs normally have better comprehensibility than the machine-generated ones. This can be illustrated by Figure 7(b).

(3) Some MRs can be deduced by referring to others in the same application domains. Let us recall the sample Scenario B in Section 2.1. With METWiki, T_2 can simply execute a query of "trigonometric function" or directly look up sin and cos. Suppose he has obtained historical MRs: $sin(x)=-sin(-x)$

and $cos(x)= cos(-x)$. Because T_2 knows $tan(x)=\frac{sin(x)}{cos(x)}$, then, it is very easy for him to get MR $tan(x)=-tan(-x)$. In other words, METWiki provides "seed MRs" for mathematic deduction. Figure 7(c) illustrates this usage.

In summary, the usages from METWiki could be multi-fold. Obviously, they are not limited to the above mentioned ones. More aspects will be investigated in our future study.

5. CONCLUSION

In retrospect, MT has been successfully applied on over 110 programs across diverse areas. However, getting enough MRs to conduct MT on a new program is still a big challenge. Having the fact that these previous studies have provided quite a few historical data, we believe that these data are valuable and can be reused, referred, or mined for new information. It is a waste if they are not known or accessible to the MT users. Therefore, we propose to build a repository that extracts and gathers these information from their original references. A prototype of this repository, METWiki, is introduced. Furthermore, by mining the historical MRs, we have explored many usages of METWiki to facilitate the MR identification. In our future work, a full version of METWiki will be completed and released, where the database will be filled gradually and kept up-to-date.

Acknowledgment

This work is partially supported by the National Natural Science Foundation of China (61572375), and Australian Research Council Linkage Grant (ARC LP100200208). We also appreciate the valuable information from the comprehensive literature review by Segura et al. [12].

6. REFERENCES

[1] T. Y. Chen, S. C. Cheung, and S. M. Yiu. Metamorphic testing: a new approach for generating next test cases. Technical Report HKUST-CS98-01, Department of Computer Science, Hong Kong University, 1998.

[2] T. Y. Chen, J. Feng, and T. H. Tse. Metamorphic testing of programs on partial differential equations: a case study. In *Proceedings of the 26th International Computer Software and Applications Conference*, pages 327–333, Oxford, England, August 2002.

[3] T. Y. Chen, J. W. K. Ho, H. Liu, and X. Xie. An innovative approach for testing bioinformatics programs using metamorphic testing. *BMC bioinformatics*, 10(1):24–35, 2009.

[4] T. Y. Chen, P.-L. Poon, and X. Xie. Metric: Metamorphic relation identification based on the category-choice framework. *Journal of Systems and Software*, page in press, 2015.

[5] T. Y. Chen, T. H. Tse, and Z. Q. Zhou. Semi-proving: An integrated method for program proving, testing, and debugging. *IEEE Transactions on Software Engineering*, 37(1):109–125, 2011.

[6] U. Kanewala and J. Bieman. Using machine learning techniques to detect metamorphic relations for programs without test oracles. In *Proceedings of the 24th IEEE International Symposium on Software Reliability Engineering*, pages 1–10, Southern California, USA, Nov 2013.

[7] U. Kanewala, J. M. Bieman, and A. Ben-Hur. Predicting metamorphic relations for testing scientific software: a machine learning approach using graph kernels. *Software Testing, Verification and Reliability*, page in press, 2015.

[8] V. Le, M. Afshari, and Z. Su. Compiler validation via equivalence modulo inputs. In *Proceedings of the 35th ACM SIGPLAN Conference on Programming Language Design and Implementation*, PLDI '14, pages 216–226, New York, NY, USA, 2014. ACM.

[9] L. Pullum and O. Ozmen. Early results from metamorphic testing of epidemiological models. In *Proceedings of 2012 ASE/IEEE International Conference on BioMedical Computing*, pages 62–67, Dec 2012.

[10] A. Ramanathan, C. Steed, and L. Pullum. Verification of compartmental epidemiological models using metamorphic testing, model checking and visual analytics. In *Proceedings of 2012 ASE/IEEE International Conference on BioMedical Computing*, pages 68–73, Dec 2012.

[11] S. Segura, A. DurÃąn, A. B. SÃąnchez, D. L. Berre, E. Lonca, and A. Ruiz-CortÃl's. Automated metamorphic testing of variability analysis tools. *Software Testing, Verification and Reliability*, 25(2):138–163, 2015.

[12] S. Segura, G. Fraser, A. B. Sanchez, and A. Ruiz-Cortes. Metamorphic testing: A literature review. Technical Report ISA-15-TR-02, Applied Software Engineering Research Group, University of Seville, Spain, 2015.

[13] Q. Tao, W. Wu, C. Zhao, and W. Shen. An automatic testing approach for compiler based on metamorphic testing technique. In *Proceedings of the 17th Asia Pacific Software Engineering Conference*, pages 270–279, Sydney, Australia, Nov 2010.

[14] X. Xie, J. W. K. Ho, C. Murphy, G. Kaiser, B. W. Xu, and T. Y. Chen. Application of metamorphic testing to supervised classifiers. In *Proceedings of the 9th International Conference on Quality Software*, pages 135–144, Jeju, Korea, August 2009.

[15] X. Xie, J. W. K. Ho, C. Murphy, G. Kaiser, B. W. Xu, and T. Y. Chen. Testing and validating machine learning classifiers by metamorphic testing. *Journal of Systems and Software*, 84(4):544–558, 2011.

[16] X. Xie, J. Tu, T. Y. Chen, and B. Xu. Bottom-up integration testing with the technique of metamorphic testing. In *Proceedings of the 14th International Conference on Quality Software*, pages 73–78, Oct 2014.

[17] X. Xie, W. E. Wong, T. Y. Chen, and B. Xu. Metamorphic slice: An application in spectrum-based fault localization. *Information and Software Technology*, 55(5):866 – 879, 2013.

[18] J. Zhang, J. Chen, D. Hao, Y. Xiong, B. Xie, L. Zhang, and H. Mei. Search-based inference of polynomial metamorphic relations. In *Proceedings of the 29th ACM/IEEE International Conference on Automated Software Engineering*, pages 701–712, Vasteras, Sweden, 2014.

[19] Z. Q. Zhou, S. Zhang, M. Hagenbuchner, T. H. Tse, F.-C. Kuo, and T. Y. Chen. Automated functional testing of online search services. *Software Testing, Verification and Reliability*, 22(4):221–243, 2012.