METRIC: METamorphic Relation Identification based on the Category-choice framework☆

Tsong Yueh Chen[a], Pak-Lok Poon[b], Xiaoyuan Xie[a]

[a]*Department of Computer Science and Software Engineering, Swinburne University of Technology, Hawthorn 3122, Australia*
[b]*School of Accounting and Finance, Faculty of Business, The Hong Kong Polytechnic University, Hung Hom, Kowloon, Hong Kong*

Abstract

Metamorphic testing is a promising technique for testing software systems, even in situations where no test oracle exists, and has been successfully applied to various application domains and paradigms. An important and essential task in metamorphic testing is the identification of metamorphic relations, which, due to the absence of a systematic and specification-based methodology, has often been done in an ad hoc manner — something which has hindered the applicability and effectiveness of metamorphic testing. To address this, a systematic methodology for identifying metamorphic relations based on the category-choice framework, called METRIC, is introduced in this paper. A tool implementing this methodology has been developed and examined in an experiment to determine the viability and effectiveness of METRIC, with the experiment results confirming that METRIC is both effective and efficient at identifying metamorphic relations.

Keywords: Software testing, metamorphic testing, metamorphic relation, test oracle, oracle problem

1. Introduction

Many testing techniques exist for detecting software faults, with examples including random testing (Hamlet, 2002), coverage testing (Lyu et al., 1994), combinatorial testing (Nie and Leung, 2011), domain testing (White and Cohen, 1980), equivalence partitioning (Myers and Sandler, 2004), the **CHO**i**C**e re**LAT**ion fram**E**work (CHOC'LATE) (Chen et al., 2003a), and the **C**lassification-**T**ree **M**ethodology (CTM) (Grochtmann and Grimm, 1993). However, application of most of these techniques requires some mechanism which can verify the correctness of the system output (Chen et al., 2003b), something known as a *test oracle* (or simply an *oracle*), in the absence of which, these testing techniques cannot be used.

When testing a software system, the *oracle problem* is said to occur when either an oracle does not exist for the tester to verify the correctness of the computed output; or an oracle does exist but

☆The work described in this paper was partially supported by a Discovery Grant of the Australian Research Council (project no. ARC DP120104773). Human subjects ethics approval was granted by The Hong Kong Polytechnic University (HSEARS20140221001) for all participants involved in the experiment described in this paper.

Email addresses: tychen@swin.edu.au (Tsong Yueh Chen), afplpoon@polyu.edu.hk (Pak-Lok Poon), xxie@swin.edu.au (Xiaoyuan Xie)

cannot be used, perhaps due to feasibility or practicality issues. The oracle problem often occurs in software testing, rendering many testing techniques inapplicable (Harman et al., 2013; Hierons, 2012).

As an attempt to alleviate the oracle problem, Chen et al. (1998, 2003b) developed a technique known as metamorphic testing (MT), which involves multiple executions based on some necessary properties, formally known as *metamorphic relations* (*MRs*), of the software under test. This approach has been receiving increasing attention in the software testing community (Gotlieb and Botella, 2003; Harman et al., 2013). MT has been successfully applied in a number of application domains and paradigms, including healthcare (Murphy et al., 2011), bioinformatics (Chen et al., 2009), feature models (Segura et al., 2011), machine learning (Xie et al., 2011), spreadsheets (Poon et al., 2014), middleware (Chan et al., 2006), and the Web (Chan et al., 2007; Zhou et al., 2012). Furthermore, real-life faults have been detected by MT in some open source and comprehensively tested programs, including: the popular Siemens suites (Xie et al., 2013); a famous Machine Learning Framework (Weka) (Xie et al., 2011); three feature model analysis systems (Segura et al., 2011); and two compilers (GCC-4.4.3 and UCC-1.6) (Tao et al., 2010). In fact, it was somewhat surprising that MT could detect these real-life faults, because these programs had been extensively used and tested. Detection of these faults emphasizes that MT is a new and promising test case selection strategy that complements the existing approaches.

MT has also been used to extend the applicability of some testing and analysis methods which require an oracle. Morell's fault-based technique (Morell, 1990) assumes the presence of an oracle, but when integrated with MT, this need for an oracle is removed (Chen et al., 2003b). Spectrum-based fault localization (Jones and Harrold, 2005) is a very popular statistical approach that, until its recent integration with MT (Xie et al., 2013), also required the presence of an oracle. Xie et al. (2013) also developed a new type of program slices, metamorphic slices, which extend the applicability of program slices in some existing applications. Furthermore, MT has also been integrated with some current fault tolerance techniques to enhance their applicability to situations without oracles (Liu et al., 2014b).

In addition to integration with other testing and analysis methods, MT fundamentals and foundations have also been developing. With symbolic execution, MT has been extended from testing whether MRs are satisfied for selected inputs as test cases, to being able to prove whether MRs are satisfied for all possible inputs and to aiding debugging through identification of the constraints on inputs that violate MRs (Chen et al., 2002, 2011). Using the framework of constraint logic programming, Gotlieb and Botella (2003) have developed a technique for a specific class of programs which can find test cases leading to the violation of an MR, or prove that an MR is satisfied. Furthermore, a recently conducted study has examined how effectively MT can be used as a substitute for the test oracle (Liu et al., 2014a). These studies have enriched the theoretical and practical foundations of MT.

In spite of the successes in applying MT to multiple application domains and paradigms, the innovative integration of MT with other testing and analysis methods, and the development of the theoretical foundations for MT, concerns remain regarding its limitations in terms of MR generation. This is understandable, given that the set of identified MRs is the essence of MT (Mayer and Guderlei, 2006), and that a common opinion has been that identification of these MRs appears to be a manual and difficult task (Groce et al., 2014; Mishra and Kaiser, 2012): "[E]ven experts may find it difficult to discover metamorphic properties [relations]" (Groce et al., 2014, p. 320). Although MR identification has represented one of the most challenging and core

problems in MT, very limited work has been done on this problem, with current MR identification methods (Kanewala and Bieman, 2013; Liu et al., 2012) all requiring the availability of some MRs for identification of further ones. In other words, there does not yet exist a systematic, effective, and generally applicable identification methodology that does not require the pre-existence of some MRs. The absence of such a methodology undoubtedly affects the applicability and effectiveness of MT. A fundamental question therefore is: ***How can MRs be systematically and effectively identified?***

Because MRs represent specific properties of the developed software, they actually form part of the software requirements (either explicitly or implicitly), which will then be included in the software specification. Thus, intuitively, a specification document should contain important and useful information that can be used for identifying the MRs. Accordingly, a specification-based methodology has been developed: **MET**amorphic **R**elation **I**dentification based on the **C**ategory-choice framework (METRIC). METRIC belongs within the specification-based category-choice framework, which includes the concepts of categories, choices, and complete test frames. Examples of this framework include CHOC'LATE and CTM, and their variants (Chen et al., 2003a, 2012; Grochtmann and Grimm, 1993; Hierons et al., 2003; Singh et al., 1997). METRIC enables the tester to systematically identify MRs from complete test frames generated by the category-choice framework.

The rest of this paper is organized as follows: Sections 2 and 3 introduce the concepts of MT/MR and the category-choice framework, respectively, both of which are essential for understanding METRIC. Section 4 explains the motivations of the study and discusses details of METRIC. Section 5 describes the main features of an associated generator tool developed to support METRIC. Sections 6 and 7 describe the setting and results of an experiment to demonstrate the viability and effectiveness of METRIC. Section 8 further elaborates this research, discussing limitations, and highlighting potential future work. Section 9 discusses some related work done by other researchers, and finally, Section 10 summarizes and concludes the paper.

2. Metamorphic testing (MT) and metamorphic relation (MR)

To outline the main concepts of MT and MR, assume a program P, and its input domain D. The size of D is often very large, or even infinite, making exhaustive testing infeasible. In such a situation, a test suite $U \subsetneq D$ may be constructed. Suppose $U = \{t_1, t_2, \ldots, t_n\}$, where t_i is a test case for any $i = 1, 2, \ldots, n$, and $P(t_i)$ represents the result of executing P with t_i. If an oracle exists, and can be feasibly applied, the correctness of $P(t_i)$ can be verified.

MT is an innovative approach to alleviating the oracle problem in testing (Chen et al., 1998, 2003b): Instead of relying on the existence and applicability of an oracle, MT uses a set of MRs (corresponding to some specific properties of the software) to generate test cases and verify test results. The following example illustrates the concepts and the process of MT.

Example 1 (SINE Function) Consider an implemented SINE function. Except for some special values such as 90° ($\sin(90^\circ) = 1$), the exact sine values for many angles remain unknown, and thus an oracle does not exist for most of the values computed by SINE. Some properties, however, do exist for the sine function, including that for any angle x (in degrees), $\sin(x) = \sin(x + 360^\circ)$. This property then forms an MR in MT: "If $y = x + 360^\circ$, then $\sin(x) = \sin(y)$". Using this MR, the implemented SINE should be executed twice: Firstly with any angle x as a *source* test

case; and then with the angle y, such that $y = x + 360^\circ$, as a *follow-up* test case. The rationale is that, although it may be very difficult to check the correctness of $\mathsf{SINE}(x)$ in one single execution (unless x is a special angle such as 90°), executing SINE again with another angle $y = x + 360^\circ$ and (automatically) checking whether $\mathsf{SINE}(y) = \mathsf{SINE}(x)$ can be easily done.

For example, suppose SINE is executed with a test case $x = 69^\circ$, giving an output of 0.9336 (i.e., $\mathsf{SINE}(69^\circ) = 0.9336$). In MT, with respect to the above MR corresponding to the property $\sin(x) = \sin(x + 360^\circ)$, SINE should next be executed with another test case $y = 69^\circ + 360^\circ = 429^\circ$. The output of the second execution should be compared with that of the first (after allowing for rounding error): Does $\mathsf{SINE}(429^\circ) = 0.9336$? If the equality does not hold, then it can be concluded that SINE is faulty. ■

Example 1 shows that, rather than focusing on the correctness of output from *one single* execution, MT checks whether a relevant MR holds among *multiple* executions. It is this characteristic that makes MT very useful in testing, even when the oracle problem occurs. Formal definitions for a Metamorphic Relation (MR) and for Metamorphic Testing (MT) are as follows:

Definition 1 (Metamorphic Relation) *Let f be a target algorithm and P be its corresponding implementation. A* ***metamorphic relation* (*MR*)** *is a necessary property of f over a series of two or more inputs ($x_1, x_2, \ldots, x_n$, where $n \geqslant 2$) and their corresponding outputs ($f(x_1), f(x_2), \ldots, f(x_n)$), that is, a relation $\mathbb{R}\big(x_1, x_2, \ldots, x_n, f(x_1), f(x_2), \ldots, f(x_n)\big)$ involving multiple inputs and their corresponding outputs.*

Without loss of generality, assume there exist k ($1 \leqslant k < n$) inputs such that, for each x_j ($(k+1) \leqslant j \leqslant n$), $x_j = f_j\big(x_1, x_2, \ldots, x_k, f(x_1), f(x_2), \ldots, f(x_k)\big)$. For $1 \leqslant i \leqslant k$, x_i is called a ***source test case****; and for $(k+1) \leqslant j \leqslant n$, x_j is called a* ***follow-up test case****. In other words, if all x_i ($1 \leqslant i \leqslant k$) are specified, then x_j ($(k+1) \leqslant j \leqslant n$) can be determined from the source test cases and their corresponding outputs.*

Definition 2 (Metamorphic Testing) *Let P be an implementation of a target algorithm f. Consider an MR: $\mathbb{R}\big(x_1, x_2, \ldots, x_n, f(x_1), f(x_2), \ldots, f(x_n)\big)$.* ***Metamorphic testing*** *of this MR for P involves the following steps: (1) Given a series of source test cases $\langle x_1, x_2, \ldots, x_k\rangle$ and their respective outputs $\langle P(x_1), P(x_2), \ldots, P(x_k)\rangle$, generate a series of follow-up test cases $\langle x_{k+1}, x_{k+2}, \ldots, x_n\rangle$ according to the function $x_j = f_j\big(x_1, x_2, \ldots, x_k, f(x_1), f(x_2), \ldots, f(x_k)\big)$, but with $f(x_i)$ being replaced by $P(x_i)$, where $1 \leqslant i \leqslant k$ and $(k+1) \leqslant j \leqslant n$. (2) Check the relation $\mathbb{R}\big(x_1, x_2, \ldots, x_n, f(x_1), f(x_2), \ldots, f(x_n)\big)$, but with $f(x_i)$ being replaced by $P(x_i)$, where $1 \leqslant i \leqslant n$. If $\mathbb{R}$ is not satisfied, then metamorphic testing of this MR reveals that P is faulty.*

In MT, for any given test case x_j ($1 \leqslant j \leqslant n$), there is no need to investigate whether $P(x_j) = f(x_j)$ by means of a test oracle. This therefore alleviates the oracle problem in testing.

3. Category-choice framework

As mentioned in Section 1, METRIC is built within the category-choice framework, which involves categories, choices, and complete test frames. Before presenting the details of METRIC, CHOC'LATE (a category-choice framework) is first used to illustrate the concepts of categories, choices, and complete test frames.

CHOC'LATE provides a systematic framework for generating test cases from specifications using categories, choices, and complete test frames (Chen et al., 2003a, 2012). By partitioning the input domain according to choices, CHOC'LATE generates a set of complete test frames based on

Table 1: Hourly parking rates

Actual Hours of Parking	Hourly Parking Rates					
	Weekday			Saturday & Sunday		
	Motorcycle	Car: 2-door Coupe	Car: Others	Motorcycle	Car: 2-door Coupe	Car: Others
$(0.0, 2.0]$	\$4.00	\$4.50	\$5.00	\$5.00	\$6.00	\$7.00
$(2.0, 4.0]$	\$5.00	\$5.50	\$6.00	\$6.50	\$7.50	\$8.50
$(4.0, 24.0]$	\$6.00	\$6.50	\$7.00	\$8.00	\$9.00	\$10.00

Table 2: Categories and choices for FEE

Categories	Associated Choices
Type of Vehicle	Type of Vehicle$_{\text{motorcycle}}$, Type of Vehicle$_{\text{car}}$
Type of Car	Type of Car$_{\text{2-door coupe}}$, Type of Car$_{\text{others}}$
Day of Week	Day of Week$_{\text{weekday}}$, Day of Week$_{\text{Saturday or Sunday}}$
Discount Coupon	Discount Coupon$_{\text{yes}}$, Discount Coupon$_{\text{no}}$
Estimated Hours of Parking	Estimated Hours of Parking $_{(0.0,\ 2.0]}$, Estimated Hours of Parking $_{(2.0,\ 4.0]}$, Estimated Hours of Parking $_{(4.0,\ 24.0]}$
Actual Hours of Parking	Actual Hours of Parking $_{(0.0,\ 2.0]}$, Actual Hours of Parking $_{(2.0,\ 4.0]}$, Actual Hours of Parking $_{(4.0,\ 24.0]}$

the constraints among choices, formally known as *choice relations*. The *complete test frames* thus generated, which are valid combinations of choices, then form the basis for test case generation. If an instance is selected from each choice in a complete test frame, a test case is formed. In this paper, the notations $\mathbb{B}$ and B are used to denote a "generic" complete test frame and a "specific" complete test frame, respectively.

Example 2 (Parking Fee System) Consider a parking fee calculation system FEE, which accepts the parking details of each vehicle, including type of vehicle, type of car, day of week, discount coupon, and hours of parking. FEE first rounds up the parking duration to the next full hour, and then calculates the parking fee for a vehicle according to the hourly rates in Table 1. If a discount coupon is presented, a 50% discount off the parking fee will be given.

To facilitate better parking management, at the time of parking, customers can optionally provide an estimation of parking duration in terms of three different time ranges, namely $(0.0, 2.0]$, $(2.0, 4.0]$, and $(4.0, 24.0]$. Suppose a customer provides an estimation. If the estimated hours of parking and the actual hours of parking fall into the same time range, then the customer will receive a 40% discount. If, however, the estimated hours and the actual hours are in different time ranges, a 20% markup will be added. A customer can choose to either use a discount coupon, or provide an estimation of parking duration, but not both. Obviously, a customer may also choose to neither provide an estimation, nor use a discount coupon. Note that no vehicles are allowed to park across two consecutive days on a continuous basis. Suppose, for instance, that a vehicle parks at 23:30 on a Monday, then it must vacate the car park not later than 23:59 on that same day.

Step 1. Categories and choices are identified from the FEE specification. A *category* is a major property or characteristic of an environment condition, or a parameter of the system that affects its execution behavior. The possible values associated with each category are partitioned into disjoint subsets known as *choices*, with the assumption that all values in the same choice are similar either in their effect on the system's behavior, or in the type of output that they produce. Given a category P, the notation P_x is used to denote a choice of P. Table 2 shows the possible categories and their associated choices for FEE.

Step 2. Constraints are identified for each pair of choices, from which valid combinations

are selected as complete test frames. For FEE, given the categories and choices defined in Table 2, exactly 90 complete test frames will be generated, an example of which is: $B_1 = \{\text{Type of Vehicle}_{\text{motorcycle}}, \text{Day of Week}_{\text{weekday}}, \text{Discount Coupon}_{\text{no}}, \text{Actual Hours of Parking}_{(0.0,\ 2.0]}\}$.

Step 3. Test cases can be formed from each complete test frame $\mathbb{B}$ generated in step 2, by selecting and combining an instance from each choice in $\mathbb{B}$. For example, taking B_1 in step 2, a test case $tc = \{\text{Type of Vehicle} = \text{motorcycle}, \text{Day of Week} = \text{Monday}, \text{Discount Coupon} = \text{no}, \text{Actual Hours of Parking} = 1.4\}$ can be formed. ■

4. METRIC methodology

4.1. A motivating example

When testing the SINE function in Example 1, MRs can be easily identified because properties of this function, such as "$\sin(x) = \sin(x + 360^\circ)$", are well known in mathematics. However, knowledge of specific properties of many applications, especially proprietary software, may not always be explicitly available, thus making ad hoc MR identification for these applications very difficult (Groce et al., 2014). This problem has represented an obstacle to the applicability and effectiveness of MT.

Previous work and experience with MT have revealed that many software practitioners adopt a similar approach for manual MR identification, as illustrated in the following example.

Example 3 (Online Hotel Availability Inquiry System) Consider a Web-based hotel availability inquiry system INQ, which outputs a list of hotels satisfying certain search criteria. Each inquiry includes input parameters such as city name, the maximum distance of the hotel from the center of the city, check-in date, check-out date, the minimum hotel category, and room type. For instance, a possible input I is: "Melbourne/10 km/Jan 3, 2014/Jan 8, 2014/5-star/single". Suppose, given I, INQ returns the output $O = \{\text{Hotel-A, Hotel-B}\}$. Although verifying whether or not Hotel-A and Hotel-B are correctly identified by INQ may be easily done (by calling both hotels, for example), because of the potentially large number of hotels involved, it could be very difficult to check whether any other eligible hotels were omitted by INQ. In other words, it is not practically feasible to check the correctness of O, resulting in an occurrence of the oracle problem for testing INQ.

When identifying an MR, a common approach is to consider how to *adjust* a given input in order to get a *predictable change* in output. For example, if the minimum hotel category is changed from "5-star" to "4-star", but all other input parameters remain unchanged, then the expected output should still include Hotel-A and Hotel-B, but possibly with some other eligible hotels. Thus, an MR can be generalized as follows: "If the minimum hotel category is lowered, the output for the old input will be a subset of that for the new input". ■

The identification pattern described in Example 3 can be summarized as the following "bottom-up" stepwise approach:

1. With respect to the implemented algorithm, select a concrete input I as a source test case.
2. Determine how to change I to another concrete input I' (a follow-up test case), such that a relation R can be defined among I, I', and their corresponding outputs. This step may be iterated until R is found (in which case the tester goes to step 4), or, after several failed attempts, the tester judges that R does not exist with I as the source test case (in which case the tester goes to step 3).

3. The identification exercise is repeated by going back to step 1, involving the selection of another concrete input as a new source test case.

4. If the *R* defined in step 2 can be generalized to a class of source test cases, then an MR is identified. Otherwise, repeat the entire process by starting from step 1, involving the selection of another concrete input as a new source test case.

4.2. Methodology

The category-choice framework (including categories, choices, and complete test frames) was used to develop a new methodology to automate, as much as possible, the manual MR identification pattern in Section 4.1. Reasons for this include:

a. As explained in Section 1, MRs should have already been explicitly or implicitly included as part of the specification, from which it should be possible to extract the relevant information for MR identification. Because the category-choice framework is specification-based, it is a good candidate for the proposed methodology.

b. In MT, each MR represents the effect on the output when an input is altered in some controlled way, therefore MRs focus on the input domain of the system. The category-choice framework also focuses on the input domain by dividing it into several disjoint partitions (known as complete test frames), from which test cases can be constructed.

c. The category-choice framework aims at generating a set of complete test frames from which concrete test cases can be constructed. Each complete test frame is a valid combination of choices, and corresponds to an input (test) scenario rather than a single concrete input (a test case). Complete test frames are therefore more abstract than concrete test cases, and as such, an MR derived from a complete test frame is applicable to *multiple* source test cases and their corresponding follow-up test cases. This makes complete test frames a natural basis for MR identification: A relation derived from concrete test cases corresponds to an *R*, and one derived from complete test frames corresponds to an MR (see step 4 of Section 4.1). Example 5 (later in this section) will illustrate the advantages of using complete test frames to identify MRs from which multiple source and follow-up test cases can be generated.

The METRIC (**MET**amorphic **R**elation **I**dentification based on the **C**ategory-choice framework) methodology focuses on MRs such that each MR involves two distinct inputs, with the associated follow-up input depending only on the corresponding source input *I* (not on the output of *I*). Consequently, only two distinct complete test frames need be considered by the tester to generate an MR, leading to the concept of a candidate pair, which serves as a cornerstone of METRIC.

In general, METRIC involves the following steps:

Step 1. Select two relevant and distinct complete test frames as a *candidate pair* for user's consideration.

Step 2. Enable the user to effectively and efficiently determine whether or not the selected candidate pair is useful for MR identification, and if it is, then allow the user to provide the corresponding MR description.

Step 3. Restart from step 1, and repeat until all candidate pairs are exhausted, or the predefined number of MRs to be generated is reached.

For the remainder of this paper, candidate pairs which are useful for MR identification are called *usable*, and those which are not are *unusable*.

In a typical MR identification process, testers have to *simultaneously* consider both the input and output aspects of the system (Section 4.1), which has been reported by software researchers and practitioners as a difficult task of MR identification. In view of this problem, METRIC provides testers with one pair ($\mathbb{B}_1$ and $\mathbb{B}_2$, which correspond to the inputs only) of complete test frames at a time, thereby enabling them to concentrate on evaluating whether a relation exists between the outputs associated with $\mathbb{B}_1$ and $\mathbb{B}_2$. If such a relation exists, then an MR can be formed.

By definition, any pair ($\mathbb{B}_1$ and $\mathbb{B}_2$) of distinct complete test frames must contain some different choices. Either (a) $\mathbb{B}_1$ and $\mathbb{B}_2$ contain different choices of the same category, or (b) one or more categories associated with $\mathbb{B}_1$ are not associated with $\mathbb{B}_2$ (or vice versa). The different choices (due to reason (a)) and extra choices (due to reason (b)) are referred to as *differentiating choices*, and the categories associated with these differentiating choices are referred to as *differentiating categories*. A pair of differentiating choices associated with the same category but which appears separately in $\mathbb{B}_1$ and $\mathbb{B}_2$ is called a *differentiating choice pair*.

A candidate pair ($\mathbb{B}_1$ and $\mathbb{B}_2$) can be classified into one of three types: a **choice-only pair** (**choice pair**); a **category-only pair** (**category pair**); or a **category-choice pair**. For a choice-only pair, the difference in choices between $\mathbb{B}_1$ and $\mathbb{B}_2$ is due only to reason (a); for a category-only pair, the difference is due only to reason (b); and a category-choice pair is a pair which is neither choice-only nor category-only. A choice, category, or category-choice pair ($\mathbb{B}_1$ and $\mathbb{B}_2$) can be written as a choice(k), category(j), or category-choice(j,k) pair, respectively, where j is the number of differentiating categories that are associated with either $\mathbb{B}_1$ or $\mathbb{B}_2$ (but not both), and k is the number of differentiating categories with different choices in both $\mathbb{B}_1$ and $\mathbb{B}_2$.

Example 4 (Types of Candidate Pairs) Referring again to B_1 in Example 2 (reproduced below), consider the following complete test frames for FEE:

$B_1 = \{\text{Type of Vehicle}_{\text{motorcycle}}, \text{Day of Week}_{\text{weekday}}, \text{Discount Coupon}_{\text{no}},$
$\text{Actual Hours of Parking}_{(0.0,\,2.0]}\}$;
$B_2 = \{\text{Type of Vehicle}_{\text{motorcycle}}, \text{Day of Week}_{\text{weekday}}, \text{Discount Coupon}_{\text{no}},$
$\text{Actual Hours of Parking}_{(2.0,\,4.0]}\}$;
$B_3 = \{\text{Type of Vehicle}_{\text{motorcycle}}, \text{Day of Week}_{\text{weekday}}, \text{Discount Coupon}_{\text{no}},$
$\text{Estimated Hours of Parking}_{(2.0,\,4.0]}, \text{Actual Hours of Parking}_{(2.0,\,4.0]}\}$; and
$B_4 = \{\text{Type of Vehicle}_{\text{motorcycle}}, \text{Day of Week}_{\text{weekday}}, \text{Discount Coupon}_{\text{no}},$
$\text{Actual Hours of Parking}_{(4.0,\,24.0]}\}$.

Consider first B_1 and B_2, which are identical other than having different choices in the category "Actual Hours of Parking" ("Actual Hours of Parking$_{(0.0,\,2.0]}$" $\in B_1$ and "Actual Hours of Parking$_{(2.0,\,4.0]}$" $\in B_2$). Thus, "Actual Hours of Parking" is the only differentiating category; and "Actual Hours of Parking$_{(0.0,\,2.0]}$" and "Actual Hours of Parking$_{(2.0,\,4.0]}$" are the differentiating choice pair. Therefore, B_1 and B_2 form a choice(1) pair.

Next, consider B_2 and B_3. Because the choice "Estimated Hours of Parking$_{(2.0,\,4.0]}$" is contained in B_3 but not in B_2, the category "Estimated Hours of Parking" is associated with B_3 but not with B_2. Thus, B_2 and B_3 form a category(1) pair, where "Estimated Hours of Parking" is the differentiating category, and "Estimated Hours of Parking$_{(2.0,\,4.0]}$" is the differentiating choice.

Finally, consider B_3 and B_4, which form a category-choice$(1,1)$ pair, where "Estimated Hours of Parking" and "Actual Hours of Parking" are the differentiating categories; and "Estimated Hours

of Parking $_{(2.0,\,4.0]}$", "Actual Hours of Parking $_{(2.0,\,4.0]}$", and "Actual Hours of Parking $_{(4.0,\,24.0]}$" are the differentiating choices. Furthermore, "Actual Hours of Parking $_{(2.0,\,4.0]}$" and "Actual Hours of Parking $_{(4.0,\,24.0]}$" are the differentiating choice pair. ■

Example 5 (Identification of MRs from Candidate Pairs) Consider B_1 and B_2 in Example 4, which are identical except in one differentiating choice pair, namely "Actual Hours of Parking $_{(0.0,\,2.0]}$" and "Actual Hours of Parking $_{(2.0,\,4.0]}$". Given the information that a longer time range of actual parking duration (related to the system input) will incur a higher parking fee (corresponding to the system output) for the same type of vehicle and day of week, if the customer does not possess a discount coupon and does not provide an estimation of parking duration, then, when comparing B_1 and B_2, the parking fee for B_2 must be higher than that for B_1. Therefore, B_1 and B_2 are usable for defining the following MR:

> (MR_1) Assume that a motorcycle parks on a weekday, and the customer neither possesses a discount coupon nor provides an estimation of parking duration upfront. If the time range of actual parking duration increases from $(0.0,\ 2.0]$ to $(2.0,\ 4.0]$, the parking fee will increase.

Next, consider B_2 and B_3 in Example 4, which differ from each other only in the differentiating choice "Estimated Hours of Parking $_{(2.0,\,4.0]}$". Because the scenario corresponding to B_3 will receive a 40% discount (an estimation of parking duration was provided upfront and it falls into the same time range as the actual hours of parking), the final parking fee for B_3 must be lower than that for B_2. In this regard, B_2 and B_3 are usable, and lead to the following MR:

> (MR_2) Assume that a motorcycle parks on a weekday with an actual parking duration of $(2.0,\ 4.0]$ and the customer does not possess a discount coupon. The provision of an estimation of parking duration of $(2.0,\ 4.0]$ upfront will result in a lower parking fee than not providing such an estimation.

Finally, consider B_3 and B_4 in Example 4. The parking fee for B_3 must be lower than B_4, because: (a) B_3 has a correct range of estimation of parking duration, resulting in a 40% discount off the parking fee; and (b) B_3 has a shorter actual parking duration than B_4, resulting in a lower hourly rate and a lower parking fee (even before applying the discount). Thus, B_3 and B_4 are usable for defining the following MR:

> (MR_3) Assume that a motorcycle parks on a weekday and the customer neither possesses a discount coupon nor provides an estimation of parking duration upfront. If the time range of actual parking duration changes from $(4.0,\ 24.0]$ to $(2.0,\ 4.0]$, and an estimation of parking duration of $(2.0,\ 4.0]$ is provided, the parking fee will be lowered.

Note that not all candidate pairs are usable. Consider B_3 in Example 4 and $B_5 = \{\text{Type of Vehicle}_{\text{car}}, \text{Type of Car}_{\text{others}}, \text{Day of Week}_{\text{weekday}}, \text{Discount Coupon}_{\text{no}}, \text{Actual Hours of Parking}_{(0.0,\,2.0]}\}$. They form a category-choice$(2, 2)$ pair, but no relationship exists between their corresponding outputs (the computed parking fees). The following three test cases illustrate this situation, where tc_1 is generated from B_3, and tc_2 and tc_3 are generated from B_5.

$tc_1 = \{$Type of Vehicle = motorcycle, Day of Week = Monday, Discount Coupon = no, Estimated Hours of Parking = 3.0, Actual Hours of Parking = 2.5$\}$;
$tc_2 = \{$Type of Vehicle = car, Type of Car = 4-door sedan, Day of Week = Monday, Discount Coupon = no, Actual Hours of Parking = 0.5$\}$; and
$tc_3 = \{$Type of Vehicle = car, Type of Car = 4-door sedan, Day of Week = Monday, Discount Coupon = no, Actual Hours of Parking = 2.0$\}$.

According to the hourly parking rates in Table 1, the parking fees for tc_1, tc_2, and tc_3 are \$9.00 ($= \$5.00/\text{hr} \times 3.0\,\text{hrs} \times 60\%$), \$5.00 ($= \$5.00/\text{hr} \times 1.0\,\text{hr}$), and \$10.00 ($= \$5.00/\text{hr} \times 2.0\,\text{hrs}$), respectively. These parking fees are calculated based on the following points, as stated earlier in the paper: (a) parking durations will be rounded up to the next full hour; and (b) a 40% discount will be given if the estimated hours of parking and the actual hours of parking fall into the same time range. Because the parking fee for tc_1 ($= \$9.00$) is higher than tc_2 ($= \$5.00$), but lower than tc_3 ($= \$10.00$), there is no relationship between the computed parking fees for B_3 and B_5, and therefore B_3 and B_5 are unusable for defining any MR. ■

For any candidate pair $\mathbb{B}_1$ and $\mathbb{B}_2$, their differentiating categories indicate the difference between their concrete source and follow-up test cases, and their non-differentiating categories indicate the commonality between them. Thus, to facilitate MR identification, when a candidate pair has some common choices, the *same* value should be used for each common choice. For instance, in test cases tc_1, tc_2, and tc_3 of Example 5, "Monday" is selected as the value from the choice "Day of Week$_{\text{weekday}}$" in B_3 and B_5.

An advantage of using complete test frames for MR identification (reason (c) in Section 4.2) is illustrated by MR_1 in Example 5, which allows the tester to generate a number of source and follow-up test cases for more comprehensive testing. Example 5 also clearly and convincingly shows that candidate pairs provide a natural and central basis upon which possible MRs can be systematically and effectively identified.

5. An associated generator tool

In order to support and automate (as far as possible) the three steps involved in METRIC (Section 4.2), a Java-based tool called **MR-GEN**erator (MR-GEN) has been developed. MR-GEN does the following: (a) takes a spreadsheet file containing a set of complete test frames (*TF*), which is generated by a tool such as the one described in Chen et al. (2003a); (b) allows the user to specify the type of candidate pairs for evaluation; (c) displays selected candidate pairs for the user to determine if they are usable; (d) records the user's MR description for usable candidate pairs; and (e) outputs an HTML file containing the identified MRs. Other facilitating features of MR-GEN include, for example, postponing evaluation of certain candidate pairs, and allowing the user to modify the currently displayed candidate pairs to control the selection of other candidate pairs. These steps and features are next illustrated using the FEE system from Example 2.

Using the categories and choices in Table 2, a set of 90 complete test frames for FEE, denoted TF_{FEE}, were generated and stored in a spreadsheet file, and then input to MR-GEN. MR-GEN prompts the user to specify the type of candidate pairs and other associated details as shown in Fig. 1. Often, resource constraints impose a limit on the number of test cases that can be executed, thus, MR-GEN provides a parameter $\overline{M}$, to specify the *maximum* number of MRs to be identified from the *TF*. MR-GEN also allows the user to specify a set of critical categories $\{C_1, C_2, \ldots, C_n\}$ (where C_i is a category, $1 \leq i \leq n$) such that each C_i must be associated with at least one of the two complete test frames in any selected candidate pair (Fig. 2). A *critical* category is one judged by the user to play a significant role in the functionality of the software under test. As a further option, the user may specify whether each selected candidate pair must have different choices in a particular C_i (Fig. 2).

After the user has specified the above parameters and critical categories, MR-GEN displays the relevant candidate pairs, one at a time, to be judged usable or not (the upper half of Fig. 3). A

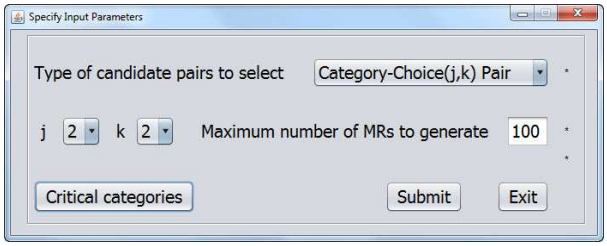

Figure 1: Input screen for specifying parameters.

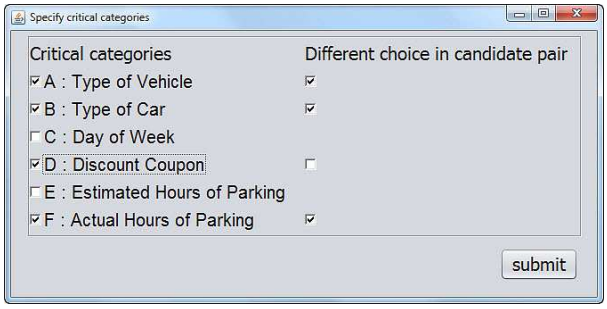

Figure 2: Selection screen for critical categories.

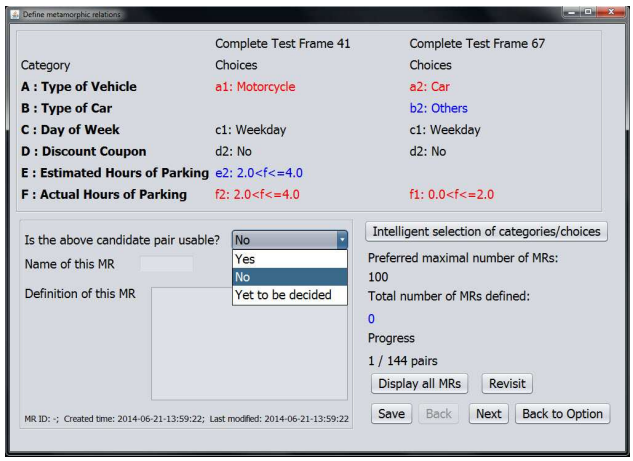

Figure 3: Screen for evaluating candidate pairs and providing MR descriptions.

combo box allows the user to indicate whether or not the pair is usable for identifying an MR. If usable, the MR details are recorded. If the user is not certain whether or not the candidate pair is usable, the pair can be labeled as "yet-to-be-decided", and the pair can be revisited later. After evaluating other candidate pairs, the user may accumulate sufficient knowledge to judge whether or not a "yet-to-be-decided" pair is usable.

Because MR-GEN presents candidate pairs based on the parameters entered in Figs. 1 and 2, the user only needs to determine if a relation exists between the corresponding *outputs*. The user does not need to select complete test frames (corresponding to the *input* aspects) for consideration, and thus a significant portion of the MR identification burden is alleviated (see the discussion on the problem of simultaneous consideration of both the input and output aspects of the system in Section 4.2). Furthermore, to facilitate comparison, MR-GEN presents the complete test frames, side-by-side, for each candidate pair, highlighting differentiating choices in different colors (Fig. 3). This arrangement is obviously more useful than relying only on the user's own ability to select some related inputs from a large pool for possible MR identification.

After identifying some MRs with MR-GEN, the user may develop a better acquaintance with the software under test, and may know what changes to the current candidate pair's choices would lead to a relation between outputs of a new pair. To cater for this, MR-GEN provides an "on-demand selection" option to modify the current candidate pairs, with a view to obtaining a new, but usable candidate pair (Fig. 4). Consider an example: Although B_3 and B_5 in Example 5 are not usable, the user may know that by replacing the choice "Actual Hours of Parking $_{(0.0,\,2.0]}$" in

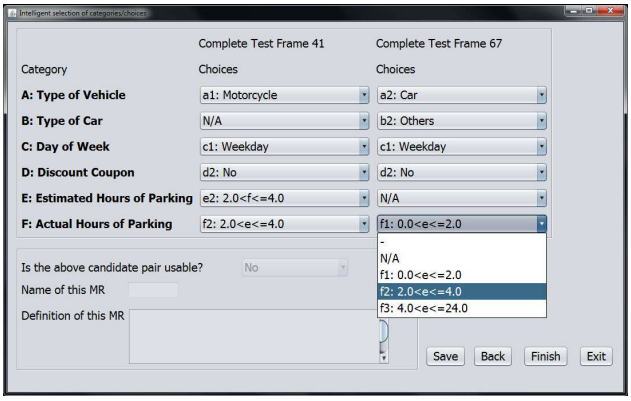

Figure 4: Screen for on-demand selection of subsequent candidate pairs.

B_5 with another choice "Actual Hours of Parking$_{(2.0,\ 4.0]}$" to form a new complete test frame B_6, a relationship exists between the outputs associated with B_3 and B_6 — the parking fee for B_6 must be higher than B_3 — and therefore B_3 and B_6 are usable. Note that the initial candidate pair which serves as the basis for on-demand selection can be usable or unusable.

6. Experimental setting

6.1. Research questions, specifications, and participants

Although METRIC is intuitive, its viability and effectiveness should be verified from the tester's perspective, leading to the following primary research question: **(*RQ1*) *Can METRIC and its associated tool MR-GEN be practically and feasibly used for MR identification?*** This research question can be decomposed into three sub-research questions: **(*RQ1.1*) *Is METRIC easy to use for users with only some primitive knowledge about MT?* (*RQ1.2*) *Does previous experience with MT have a significant impact on the user's MR identification when using METRIC?* (*RQ1.3*) *Does increased familiarity with METRIC and MR-GEN result in a significant improvement in the user's MR identification?***

Intuitively, METRIC should ease the difficulties encountered by testers in MR identification. Furthermore, users, especially those familiar with the specifications and MR-GEN, can use the tool's on-demand feature to select particular candidate pairs for later evaluation, leading to the second research question: **(*RQ2*) *Does use of MR-GEN's on-demand selection feature result in a significant improvement in MR identification?***

Two commercial specifications were used in the experiment: S_{CAR}, which is related to a company car and expense claim system (CAR) for a multinational trading firm (Chen et al., 2012); and S_{MOS} which is related to a meal ordering system (MOS) for an international company providing catering services for many different airlines (Chen et al., 2003a). The experiment involved 19 participants (denoted $\mathbb{P}_1$, $\mathbb{P}_2$, ..., $\mathbb{P}_{19}$), who were postgraduate IT students or who had earned a PhD degree in IT. The participants were classified into two broad groups: The *inexperienced* (*I-group*), who had no knowledge of MT before the experiment (ten participants, $\mathbb{P}_1$ to $\mathbb{P}_{10}$); and the *experienced* (*E-group*) who did have prior MT knowledge and experience (nine participants, $\mathbb{P}_{11}$ to $\mathbb{P}_{19}$). Furthermore, the I-group and the E-group were randomly and evenly divided into two (I-group1 and I-group2) and three (E-group1, E-group2, and E-group3) subgroups, respectively.

Table 3: MR identification arrangements for different parts of the experiment

Participant Groups	Participants	Part I: Using MR-GEN *without* On-demand Selection		Part II: Using MR-GEN *with* On-demand Selection	
		Ia	Ib	IIa	IIb
		For 40 MRs	Within 90 min	For 40 MRs	Within 90 min
		Specification(s) Used			
I-group1	$\mathbb{P}_1$ to $\mathbb{P}_5$	S_{CAR}	S_{CAR}	–	–
I-group2	$\mathbb{P}_6$ to $\mathbb{P}_{10}$	S_{MOS}	S_{MOS}	–	–
E-group1	$\mathbb{P}_{11}$ to $\mathbb{P}_{13}$	S_{CAR}	S_{CAR}	–	–
E-group2	$\mathbb{P}_{14}$ to $\mathbb{P}_{16}$	S_{MOS}	S_{MOS}	–	–
E-group3	$\mathbb{P}_{17}$ to $\mathbb{P}_{19}$	–	–	S_{CAR} & S_{MOS}	S_{CAR} & S_{MOS}

6.2. Experimental preparation and components

Three tutorials were organized to prepare the participants for the experiment. **Tutorial 1 (Specifications)** introduced the S_{CAR} and S_{MOS} specifications to all participants. **Tutorial 2 (MT)** was provided to the inexperienced participants only. It introduced the concept of MT and included two hands-on exercises to reinforce understanding. **Tutorial 3 (Category-Choice Framework and MR-GEN)** was provided to all participants and had two purposes, the first of which was to introduce the concepts of categories, choices, and complete test frames. This was done by revisiting S_{CAR} and S_{MOS}, and examining some complete test frames for CAR and MOS, generated by a tool developed for CHOC'LATE (Chen et al., 2003a). The second purpose of this tutorial was to teach the functionality and operation of MR-GEN. Tutorials 1 and 2 each lasted one hour, and Tutorial 3 lasted for 90 minutes.

The experiment was divided into two parts as follows:

Part I (MR identification supported by MR-GEN without on-demand selection) was conducted after Tutorial 3, and involved all groups except E-group3. This part was further divided into two phases, with a one-hour break between them. In the first phase (Part Ia), each participant independently identified 40 MRs from either S_{CAR} or S_{MOS}, without any time constraint: Because this was their first time to use MR-GEN, the participants would not yet be very familiar with the tool, and therefore by not setting a time limit, the participants were expected to focus more on *correctness* than on speed of performing the task. After this phase, the participants should have become used to the tool, and were therefore given a time limit of 90 minutes in the second phase (Part Ib), in which they continued to identify as many MRs as possible, with the same specifications used by them as in Part Ia, but with different candidate pairs.

Part II (MR identification supported by MR-GEN with on-demand selection) took place after Tutorial 3, and involved only E-group3. This part has two rounds, the first using S_{CAR}, and the second using S_{MOS}. Similar to Part I, each round was divided into two phases with a one-hour break between them, with each participant identifying MRs for the relevant specification, firstly to get 40 MRs with no time constraint (Part IIa), and then to get as many MRs as possible within a period of 90 minutes (Part IIb).

Table 3 summarizes the details of these arrangements.

6.3. Specification of input parameters

The input/selection screens (Figs. 1 and 2) allow users to specify the candidate pairs' types and other associated details, the maximum number of MRs to be identified, and the critical categories. Provision of such parameters allows the users to customize the selection process according to their own needs and preferences, but may also render impossible a fair examination of the different

Table 4: Measures for participants' performance

Notation	Meaning
g_{MR}	number of **g**enuine **MR**s
e_{u}	number of candidate pairs **e**valuated as **u**sable [*]
e_{n}	number of candidate pairs **e**valuated as **n**ot usable (that is, unusable)
c_{n}	number of candidate pairs **c**orrectly evaluated as **n**ot usable (that is, unusable)
R_{g}	$R_{\mathsf{g}} = \frac{g_{\mathsf{MR}}}{e_{\mathsf{u}}}$, which serves as the ***soundness*** measure of MR identification [#]
R_{n}	$R_{\mathsf{n}} = \frac{c_{\mathsf{n}}}{e_{\mathsf{n}}}$ if $e_{\mathsf{n}} > 0$, or $R_{\mathsf{n}} = 1$ if $e_{\mathsf{n}} = 0$; R_{n} serves as the ***completeness*** measure of MR identification [#]
T	**T**ime taken (in minutes) for identifying a certain number of MRs [†]
T_{g}	average **T**ime taken (in minutes) for identifying a **g**enuine MR [‡]

[*] In Parts Ia and IIa, $e_{\mathsf{u}} = 40$. In Parts Ib and IIb, e_{u} was not fixed.
[#] Higher values of R_{g} and R_{n} mean better results.
[†] In Parts Ia and IIa, T was not fixed. In Parts Ib and IIb, $T = 90$ min.
[‡] In Parts Ia and IIa, $T_{\mathsf{g}} = \frac{T}{g_{\mathsf{MR}}}$. In Parts Ib and IIb, $T_{\mathsf{g}} = \frac{90\,\mathrm{min}}{g_{\mathsf{MR}}}$. The smaller the value of T_{g}, the better the result.

components in the research questions (Section 6.1). The parameter specification feature was therefore disabled for the experiment, and the participants were instead provided with a standard setup, where candidate pairs of any type were randomly selected by MR-GEN for evaluation. This setup made it possible to filter out the effects due to a particular sequence of candidate pairs, and thereby measure the viability and effectiveness of METRIC. [1]

6.4. Measurement

MR-GEN timestamps and logs all major events, such as the determination of whether or not candidate pairs are usable, making it possible to examine the time taken and correctness of the evaluations after the experiment.

If an MR identified by the participants satisfied Definition 1, then it was called a *genuine* MR, otherwise it was *non-genuine*. The researchers examined all experiment logs to categorize the participant-identified MRs as genuine or non-genuine. From here until the end of Section 7, the term "metamorphic relations (MRs)" refers to those identified by the participants, which may or may not be genuine. The experiment determined the soundness and completeness of participants' evaluations. ***Soundness*** (R_{g}) was calculated as the ratio of genuine MRs identified by the participants to the candidate pairs evaluated as usable by the participants using MR-GEN. In this way, R_{g} is a measure of the true-positive rate. ***Completeness*** (R_{n}) was calculated as the proportion of the *actually* unusable candidate pairs among the candidate pairs evaluated as unusable by the participants, and is therefore a measure of the true-negative rate. Obviously, higher values of R_{g} and R_{n} mean better results. In MT, soundness is more important than completeness because: (i) every MR used in the construction of source and follow-up test cases must be genuine, otherwise, interpretation of the testing results may be meaningless; and (ii) if the user has already identified some genuine MRs, MT can still be conducted even if some *actually* usable candidate pairs have been *incorrectly* identified as unusable.

Table 4 shows some measures from which important statistics for each participant were collected. Other measures will be introduced in the next section.

[1] The random order of candidate pairs displayed for user evaluation was not followed in Part II, in which the impact of users' choice of candidate pairs using the on-demand selection feature was examined.

Table 5: Performance of MR identification in Part Ia (inexperienced versus experienced participants)

S_{CAR}										
Statistics	**I-group1**						**E-group1**			
	$\mathbb{P}_1$	$\mathbb{P}_2$	$\mathbb{P}_3$	$\mathbb{P}_4$	$\mathbb{P}_5$	**Mean**	$\mathbb{P}_{11}$	$\mathbb{P}_{12}$	$\mathbb{P}_{13}$	**Mean**
R_{g} ($= g_{\mathsf{MR}}/e_{\mathsf{u}}$)	0.98 (39/40)	0.95 (38/40)	0.98 (39/40)	0.98 (39/40)	1.00 (40/40)	**0.98**	1.00 (40/40)	0.95 (38/40)	0.95 (38/40)	**0.97**
R_{n} ($= c_{\mathsf{n}}/e_{\mathsf{n}}$)	0.92 (23/25)	1.00 (20/20)	0.86 (12/14)	0.85 (22/26)	1.00 (21/21)	**0.93**	0.95 (21/22)	0.92 (22/24)	1.00 (11/11)	**0.96**
T_{g} (min)	1.92	1.97	1.79	0.90	1.88	**1.69**	1.88	2.37	1.97	**2.07**
S_{MOS}										
Statistics	**I-group2**						**E-group2**			
	$\mathbb{P}_6$	$\mathbb{P}_7$	$\mathbb{P}_8$	$\mathbb{P}_9$	$\mathbb{P}_{10}$	**Mean**	$\mathbb{P}_{14}$	$\mathbb{P}_{15}$	$\mathbb{P}_{16}$	**Mean**
R_{g} ($= g_{\mathsf{MR}}/e_{\mathsf{u}}$)	0.95 (38/40)	1.00 (40/40)	0.98 (39/40)	1.00 (40/40)	0.98 (39/40)	**0.98**	1.00 (40/40)	0.80 (32/40)	1.00 (40/40)	**0.93**
R_{n}*	1.00	1.00	1.00	1.00	1.00	**1.00**	1.00	1.00	1.00	**1.00**
T_{g} (min)	3.82	2.75	2.31	1.75	2.82	**2.69**	2.50	3.28	1.63	**2.47**

* For participants $\mathbb{P}_6$ to $\mathbb{P}_{10}$ and $\mathbb{P}_{14}$ to $\mathbb{P}_{16}$, $e_{\mathsf{n}} = 0$. According to the scheme for determining the value of R_{n} in Table 4, if $e_{\mathsf{n}} = 0$, then $R_{\mathsf{n}} = 1$.

7. Experimental results and analyses

7.1. MR identification feasibility of METRIC and MR-GEN (RQ1)

7.1.1. Practicality and feasibility of METRIC for users with primitive MT background (RQ1.1)

RQ1.1 was addressed by examining the performance of inexperienced participants in Part Ia of the experiment, in which they identified 40 MRs without time constraint, supported by MR-GEN (with the on-demand selection feature disabled).

In Table 5, the columns with the headings "I-group1" and "I-group2" show the relevant statistics. Looking first at S_{CAR}. For I-group1, although the five inexperienced participants $\mathbb{P}_1$ to $\mathbb{P}_5$ received only one hour of MT training in Tutorial 2 and another 90 minutes training on MR-GEN in Tutorial 3, on average, they were able to identify MRs with a very high soundness measure (R_{g}: mean $= 0.98$, range $= [0.95, 1.00]$). Regarding the completeness measure, on average, the result was also encouraging (R_{n}: mean $= 0.93$, range $= [0.85, 1.00]$). Similar results were observed for S_{MOS}. These promising results clearly confirm that using METRIC and its associated tool MR-GEN are practically feasible from the perspective of those users with primitive MT knowledge.

7.1.2. Impact of user's MT experience on MR identification when using METRIC (RQ1.2)

RQ1.2 was addressed by comparing the performance of experienced and inexperienced participants in Part Ia of the experiment. Refer to Table 5 again. For S_{CAR}, the soundness and completeness measures for I-group1 are quite comparable to the E-group1 statistics (R_{g}: mean $=$ 0.97, range $= [0.95, 1.00]$; R_{n}: mean $= 0.96$, range $= [0.92, 1.00]$) Similar comparable results were observed for S_{MOS}. Furthermore, considering S_{CAR} and S_{MOS} together, the individual values of R_{g} and R_{n} were quite similar across all participants. In other words, the impact of previous MT experience on the soundness and completeness aspects of MR identification (using METRIC) appears to be negligible.

Consider next how quickly genuine MRs were identified, in terms of T_{g}. Here, mixed results were obtained: For S_{CAR}, the mean value of T_{g} was lower in I-group1 than in E-group1; but for S_{MOS}, the opposite was observed. If, however, both specifications are considered together, the mean value of T_{g} is 2.19 min ($\frac{1.69+2.69}{2}$) for inexperienced participants, and 2.27 min ($\frac{2.07+2.47}{2}$) for

experienced participants. It can be seen that the two mean values are about the same. [2]

At first, it may appear somewhat surprising that, in some cases, the R_g and T_g values for inexperienced participants were slightly better than those for the experienced participants. On reflection, however, some reasons for this can be identified as follows. The determination of "experienced participants" (in E-group1 and E-group2) was based on having previous experience in ad hoc MR identification in some other application domains, which was assumed to be somehow advantageous. The apparent impact of this "advantage", however, was absent in the experiment because of three reasons. Firstly, although MT has a sound concept, its technical content is simple to understand, and offering Tutorial 2 to the inexperienced participants largely reduced or eliminated the MT knowledge gap. Secondly, since METRIC is highly intuitive, simple (but elegant), and supported by its generator tool MR-GEN, both experienced and inexperienced participants were able to easily apply the methodology, which reduced the possible performance gap caused by previous MR identification experience. Thirdly, although the experienced participants had MR identification experience in other application domains, such experience might not have been directly applicable to S_{CAR} and S_{MOS}. On the contrary, when compared with the inexperienced participants, the "historical baggage" of previously using an ad hoc approach might have made the experienced participants less able to change their mindset to use METRIC appropriately shortly after learning the methodology.

7.1.3. MR identification improvement after some use of METRIC and MR-GEN (RQ1.3)

In I-group1, I-group2, E-group1, and E-group2, participants used MR-GEN to identify MRs for the same assigned specifications in Parts Ia and Ib. This arrangement made it possible to analyze the changes in MR identification after using MR-GEN for a certain period of time, which was done by, for each relevant participant, analyzing and comparing the statistics R_g, R_n, and T_g in Parts Ia and Ib. From Parts Ia to Ib, the percentage *decrease* in the time taken to identify a genuine MR (denoted R_{T_g}) was also calculated, using the formula $R_{T_g} = \frac{T_g \text{ in Part Ia} - T_g \text{ in Part Ib}}{T_g \text{ in Part Ia}}$. Obviously, a positive value of R_{T_g} indicates an increase in *speed* for identifying genuine MRs.

Table 6 shows the relevant statistics for Part I. Following the observation in Section 7.1.2 that the impact of MT experience on METRIC performance appears minimal, the participants' data are presented without classification into the experienced or inexperienced groups.

As shown in Table 6, with the exception of $\mathbb{P}_{15}$, all participants had high R_{T_g} values, with mean values of 45.4% and 37.4% for S_{CAR} and S_{MOS}, respectively. In other words, for almost all participants, much less time was required to identify a genuine MR after having used MR-GEN for a while. Note that, in addition to increased familiarity with MR-GEN, another factor may also have influenced the improvement in MR identification. After identifying 40 MRs in Part Ia for the assigned specifications, the participants had become more familiar with the specifications, which may have improved their performance in Part Ib. Nevertheless, because time savings for identification of a genuine MR (R_{T_g}) were so substantial, it is unlikely that such savings were solely contributed by increased familiarity with the specifications, without any influence from increased familiarity with METRIC and MR-GEN. These observations, together with the earlier

[2] Analyses were not conducted on R_g, R_n, or T_g in Part Ib of the experiment. This is because, after identifying 40 MRs using MR-GEN in Part Ia, different participants might have different proficiency with the tool and, hence, might make the analyses related to RQ1.2 invalid. Note that the learning aspect of MR-GEN by the participants will be addressed by RQ1.3.

Table 6: Change in performance of MR identification in Part I

Participants	Part Ia			Part Ib			From Part Ia to Ib
	R_{g}	$R_{\mathsf{n}}{}^{*}$	T_{g} (min)	R_{g}	$R_{\mathsf{n}}{}^{*}$	T_{g} (min)	$R_{T_{\mathsf{g}}}$ (%)
S_{CAR}							
$\mathbb{P}_1$	0.98	0.92	1.92	1.00	0.91	1.05	45.3
$\mathbb{P}_2$	0.95	1.00	1.97	1.00	1.00	0.86	56.3
$\mathbb{P}_3$	0.98	0.86	1.79	0.99	0.93	1.11	38.0
$\mathbb{P}_4$	0.98	0.85	0.90	0.98	1.00	0.45	50.0
$\mathbb{P}_5$	1.00	1.00	1.88	0.99	1.00	0.60	68.1
$\mathbb{P}_{11}$	1.00	0.95	1.88	1.00	1.00	1.13	39.9
$\mathbb{P}_{12}$	0.95	0.92	2.37	1.00	0.93	2.00	15.6
$\mathbb{P}_{13}$	0.95	1.00	1.97	0.95	0.93	0.99	49.7
Mean	**0.97**	**0.94**	**1.84**	**0.99**	**0.96**	**1.02**	**45.4**
S_{MOS}							
$\mathbb{P}_6$	0.95	1.00	3.82	0.96	1.00	1.88	50.8
$\mathbb{P}_7$	1.00	1.00	2.75	0.99	1.00	1.11	59.6
$\mathbb{P}_8$	0.98	1.00	2.31	1.00	1.00	1.29	44.2
$\mathbb{P}_9$	1.00	1.00	1.75	0.99	1.00	1.22	30.3
$\mathbb{P}_{10}$	0.98	1.00	2.82	0.96	1.00	1.05	62.8
$\mathbb{P}_{14}$	1.00	1.00	2.50	0.96	1.00	2.00	20.0
$\mathbb{P}_{15}$	0.80	1.00	3.28	0.90	1.00	3.21	2.1
$\mathbb{P}_{16}$	1.00	1.00	1.63	0.91	1.00	1.15	29.4
Mean	**0.96**	**1.00**	**2.61**	**0.96**	**1.00**	**1.61**	**37.4**

* For participants $\mathbb{P}_6$ to $\mathbb{P}_{10}$ and $\mathbb{P}_{14}$ to $\mathbb{P}_{16}$, $e_{\mathsf{n}} = 0$. According to the scheme for determining the value of R_{n} in Table 4, if $e_{\mathsf{n}} = 0$, then $R_{\mathsf{n}} = 1$.

finding related to RQ1.2, reinforce the conclusion that the performance using METRIC depends more on familiarity with the methodology and knowledge of the application domains, rather than on previous experience with MT.

In addition to the above observations, Table 6 shows a slight improvement in soundness and completeness for S_{CAR} from Part Ia (mean $R_{\mathsf{g}} = 0.97$; mean $R_{\mathsf{n}} = 0.94$) to Part Ib (mean $R_{\mathsf{g}} = 0.99$; mean $R_{\mathsf{n}} = 0.96$); but no change for S_{MOS}. This phenomenon of little or no improvement of R_{g} and R_{n} is understandable because their values in Part Ia were already very close to the possible maximum of 1.0.

A summary of the results is as follows. Firstly, even for inexperienced participants (with no previous knowledge about MT prior to the experiment) who have been trained with MT knowledge for only an hour and with the functionality of MR-GEN for 90 minutes, their performance of MR identification by using METRIC (with the support of MR-GEN) was fairly impressive and encouraging, in terms of the soundness and completeness measures (RQ1.1). Secondly, when using MR-GEN, both inexperienced and experienced participants had comparable, very good performance, indicating that METRIC is effective for users with different backgrounds and knowledge of MT (RQ1.2). Thirdly, after using MR-GEN for some time, almost all participants had significantly better MR identification (RQ1.3). Therefore, a clear answer to RQ1 is that METRIC and MR-GEN are practical and feasible for MT identification.

7.2. Impact of on-demand selection on the performance of METRIC (RQ2)

Changes in participants' MR identification when provided with MR-GEN's on-demand selection feature were also investigated, with the data for S_{CAR} shown in Table 7. [3]

[3] The S_{MOS} data were similar to those for S_{CAR} in Table 7. Due to page limitation, the S_{MOS} data and related discussion are not included in this paper. Also, some statistics collected in Part I related to S_{CAR}, which are shown in Table 6, are included in Table 7 for easy comparison.

Table 7: With and without on-demand selection in Parts I and II related to S_{CAR}

***Without* On-demand Selection**				***With* On-demand Selection**			
Participants	R_{g}	R_{n}	T_{g} **(min)**	**Participants**	R_{g}	R_{n}	T_{g} **(min)**
Part Ia (for 40 MRs)				**Part IIa (for 40 MRs)**			
$\mathbb{P}_{11}$ (in E-group1)	1.00	0.95	1.88	$\mathbb{P}_{17}$ (in E-group3)	0.95	0.91	1.58
$\mathbb{P}_{12}$ (in E-group1)	0.95	0.92	2.37	$\mathbb{P}_{18}$ (in E-group3)	0.95	0.75	1.58
$\mathbb{P}_{13}$ (in E-group1)	0.95	1.00	1.97	$\mathbb{P}_{19}$ (in E-group3)	1.00	0.89	1.75
Part Ib (within 90 min)				**Part IIb (within 90 min)**			
$\mathbb{P}_{11}$ (in E-group1)	1.00	1.00	1.13	$\mathbb{P}_{17}$ (in E-group3)	0.97	0.93	0.92
$\mathbb{P}_{12}$ (in E-group1)	1.00	0.93	2.00	$\mathbb{P}_{18}$ (in E-group3)	1.00	0.90	1.22
$\mathbb{P}_{13}$ (in E-group1)	0.95	0.93	0.99	$\mathbb{P}_{19}$ (in E-group3)	1.00	0.87	1.14
Mean	**0.98**	**0.96**	**1.72**	**Mean**	**0.98**	**0.88**	**1.37**

Consider the soundness (R_{g}) and completeness (R_{n}) measures in Table 7: Between Parts I (without on-demand selection) and II (with on-demand selection), the mean values (0.98) for R_{g} were identical; but there was a small percentage decrease of 8.3% ($\frac{0.96-0.88}{0.96} \times 100\%$) for R_{n}. For T_{g}, the percentage improvement was 20.3% ($\frac{1.72-1.37}{1.72} \times 100\%$) from Parts I to II, which was a large change.

Recall that soundness (R_{g}) is more important than completeness (R_{n}) in MT. In addition, although there was a small percentage decrease in R_{n} when using the on-demand selection, this decrease was more than offset by the large percentage improvement in T_{g} (to some extent, T_{g} can also be considered a soundness measure because it indicates how quickly genuine MRs are identified). In summary, with respect to RQ2, Table 7 confirms the merit of the on-demand selection feature for improving MR identification.

8. Discussion, limitations, and future work

8.1. Discussion

Based on our previous research experience on ad hoc MR identification, two problems associated with this approach are observed. Such problems, however, do not occur in METRIC. Firstly, the ad hoc MR identification often goes through several peaks and troughs. The testers spend time discovering new knowledge about the input and output aspects of the assigned specifications, but once understood, MRs related to the specification are quickly identified (corresponding to a peak). The MR identification then gradually slows down, eventually to the point where no further MR can be identified based on this new knowledge (corresponding to a trough). When another new piece of knowledge is later discovered, the MR identification again speeds up and reaches another peak. As time passes, and the testers gradually run out of new ideas, the time required to discover new knowledge becomes increasingly longer, until a stage is reached when no further new knowledge can be discovered. Thus, if the testers are asked to identify a fairly large number of MRs, say 100, using the ad hoc approach, then they may not be able to finish the task.

On the other hand, since selection of candidate pairs for evaluation is done automatically by the tool, this problem of decreasing MR identification will not occur when using MR-GEN. With MR-GEN, the users need only focus on one candidate pair at a time, to determine whether or not it is usable. In practice, the number of usable candidate pairs is very often much larger than the required number of MRs to be identified. Thus, the "unprocessed" usable candidate pairs will not be exhausted before the required number of MRs have been identified. Furthermore, in general,

usable candidate pairs are uniformly distributed over the candidate pairs selected by MR-GEN, if the on-demand selection feature is disabled. Therefore, the performance of the users in MR identification over time should not decrease due to the gradual use-up of the "unprocessed" usable candidate pairs. This point is confirmed by the observation that the values of T_g are smaller in Part Ib than in Part Ia in Table 6.

Secondly, the ad hoc approach does not have an *explicit* distinction between the two steps: (a) selecting, from a large pool of test cases, those which are potentially useful for MR identification; and (b) determining the corresponding MRs (if any) from the selected test cases. As such, in the ad hoc approach, many testers are unlikely to identify MRs involving differences in *multiple* variables. This is because such an attempt requires the testers to determine what simultaneous changes of multiple variables would give rise to MRs, which is a fairly complex cognitive task and further complicated by handling steps (a) and (b) above at the same time. On the other hand, the tendency of not identifying MRs involving differences in multiple variables does not exist in METRIC. In the methodology supported by MR-GEN, after the users have specified the numbers of differentiating categories and differentiating choices (and other parameters), the tool will *automatically* select different types of candidate pairs for consideration during MR identification, even including those involving *multiple* differentiating categories and choices (if the users so specify). Thereafter, the relevant candidate pairs will be displayed, one at a time, for user evaluation. This arrangement makes an explicit distinction between steps (a) and (b) above, resulting in their separate processing. This advantage, together with the visual side-by-side presentation of candidate pairs and highlighting of differentiating choices with different colors, ease the task of identifying MRs from candidate pairs, even if these candidate pairs involve several differentiating categories and choices.

To apply METRIC, users may start by identifying categories and choices, and then construct complete test frames. In the experiment, however, the participants were provided with complete test frames at the start of MR identification. This was done because category-choice framework techniques are now very well developed, with a lot of work already done for identifying proper and appropriate categories and choices (Chen et al., 2012; Hierons et al., 2003; Poon et al., 2011; Singh et al., 1997), and for constructing complete test frames (Chen et al., 2003a; Grochtmann and Grimm, 1993; Hierons et al., 2003; Kruse and Luniak, 2010; Singh et al., 1997). Furthermore, several tools have been developed to support the construction of complete test frames (Cain et al., 2004; Chen et al., 2003a; Grochtmann and Grimm, 1993; Lehmann and Wegener, 2000). Therefore, it should not be a difficult task to obtain categories, choices, and complete test frames from specifications. In other words, complete test frames are easily available for the application of METRIC.

As with most other methodologies, it is unrealistic to expect that METRIC could be effectively used to identify MRs for all application domains, including those such as "$\sin x = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \ldots$". Nevertheless, METRIC does have very broad applicability, partly because it is based on the category-choice framework, which has been confirmed by many studies to be applicable to a wide range of application domains, including non-scientific ones. Examples include airline catering systems, integrated ship management systems, adaptive cruise control systems for vehicles, rail ticketing systems, automatic mail sorting systems, online telephone inquiry systems, and inventory management systems (Chen et al., 2003a, 2012; Conrad and Krupp, 2006; Grochtmann and Grimm, 1993; Hierons et al., 2003; Singh et al., 1997). Based on previous work on MR, and

feedback obtained from software researchers and practitioners, there appears to be a general perception that, although it may not be difficult to identify MRs for scientific applications (such as the implemented SINE function in Example 1), it may be more challenging to do so for non-scientific applications. The work on METRIC provides a solution to this problem.

8.2. Study limitations

There are three limitations to the present studies. Firstly, the studies only involved 19 participants. Although it would be desirable to have more participants involved, it is not easy to find a large group of software testers (especially those with MT knowledge and experience), who are willing to participate. Secondly, only two commercial specifications were used in the studies. As is known, obtaining real-life specifications from industry is difficult because most companies are hesitant to release them for external use. Nonetheless, the studies do provide a convincing demonstration of the viability and effectiveness of METRIC. Thirdly, it would be better to compare METRIC with other similar methodologies, however, as pointed out in Section 1, we are not aware of any other systematic and generally applicable methodologies for identifying MRs directly from specifications. Thus, such a comparison is not applicable. This issue, in fact, clearly demonstrates the novelty and contribution of METRIC.

8.3. Future work

As the first study, the current version of METRIC focuses on generating those MRs which are associated with exactly two distinct complete test frames. Obviously, generating MRs involving more than two complete test frames is also possible with METRIC, by extending the current methodology to make use of more complete test frames for MR identification. This extension will be explored further in our future work. In addition, our current experimental study has mainly targeted at the effectiveness of MR identification with the use of METRIC. We plan to conduct further experiments (possibly involving mutation analysis) to investigate the fault-detection capability of the MRs generated by METRIC.

9. Related work

Kaiser and her colleagues (Mishra and Kaiser, 2012; Mishra et al., 2013) conducted two studies involving teaching students about MT at Columbia University. Their objective was to determine how well the students identified MRs in an ad hoc manner. Both studies found that the students occasionally failed to identify genuine MRs, and some of their identified MRs were non-genuine. Mayer and Guderlei (2006) conducted an empirical study involving Java applications, with a view to classifying MRs into several different types using mutation analysis. However, neither this study (Mayer and Guderlei, 2006), nor the studies by Kaiser and her colleagues (Mishra and Kaiser, 2012; Mishra et al., 2013) provide a systematic technique for MR identification.

Although systematic generation of MRs is essential for the automation of MT, to date, very limited work has been done in this area, with the only published investigations by Kanewala and Bieman (2013) and Liu et al. (2012). The technique developed by Kanewala and Bieman (2013) uses a machine-learning approach to detect MRs from a system's control flow graph, but machine learning requires the use of a training set for creating the predictive model. In other words, Kanewala and Bieman's technique requires that some MRs have already been identified as a training set, through which new MRs can then be generated. However, obtaining a good

and comprehensive training set in machine learning may not be easy (Cortes et al., 1995). The technique from Liu et al. (2012) attempts to construct new MRs from existing ones through composition and, hence, can only generate certain specified types of MRs. In summary, both of these techniques (Kanewala and Bieman, 2013; Liu et al., 2012) require the presence of some existing MRs in order to identify new ones. METRIC, however, differs from them because it does not have this requirement of existing MRs. Furthermore, METRIC is more generic and generally applicable than the other two techniques.

10. Summary and conclusion

The problem of systematic identification for metamorphic relations (MRs) directly from specifications is generally considered to be a challenging, but important, prerequisite for effective metamorphic testing (MT). To date, all published work on MR identification requires the pre-existence of some MRs for the generation of new ones, a situation which is far from desired. To address this problem, our paper has proposed the METRIC identification methodology, which helps users identify MRs from specifications in a systematic manner.

METRIC is built upon the category-choice framework. Through the systematic identification of usable candidate pairs, MRs can be easily and intuitively generated, even for those "complex" MRs involving multiple differentiating categories and choices. To further improve the effectiveness of METRIC, and ease the identification task, an associated generator tool MR-GEN has been developed. The tool includes several useful features such as: (a) specification of the type of candidate pairs for user evaluation; (b) specification of critical categories; (c) visual side-by-side presentation of candidate pairs and highlighting of differentiating choices to facilitate user evaluation; and (d) on-demand selection of subsequent candidate pairs.

To determine the viability and effectiveness of METRIC (supported by MR-GEN), an experiment using two commercial specifications was conducted, involving two groups of participants (one group experienced in MT, and the other not). The main purpose of the experiment was to answer two research questions: (i) Can METRIC and MR-GEN be practically and feasibly used for MR identification? (ii) Does use of MR-GEN's on-demand selection feature result in a significant improvement in MR identification? With respect to both research questions, the experimental results were largely positive and encouraging, even for those users without substantial MT knowledge. Thus, METRIC represents a significant contribution, and will, without doubt, improve the applicability, effectiveness, and automation of MT.

Acknowledgment

We would like to express our gratitude to Prof. Huimin Lin of the Institute of Software at the Chinese Academy of Sciences for providing the laboratory facilities for the experiment. We gratefully acknowledge the contribution of all the participants from Beihang University and the University of Science and Technology Beijing who were involved in the experiment. Furthermore, we are indebted to Dr. Dave Towey of the University of Nottingham Ningbo China for his invaluable comments and discussions.

References

Cain, A., Chen, T. Y., Grant, D., Poon, P.-L., Tang, S.-F., Tse, T. H., 2004. An automatic test data generation system based on the integrated classification-tree methodology. Software Engineering Research and Applications, 225–238.

Chan, W. K., Chen, T. Y., Lu, H., Tse, T. H., Yau, S. S., 2006. Integration testing of context-sensitive middleware-based applications: a metamorphic approach. International Journal of Software Engineering and Knowledge Engineering 16 (5), 677–703.

Chan, W. K., Cheung, S. C., Leung, K. R., 2007. A metamorphic testing approach for online testing of service-oriented software applications. International Journal of Web Services Research 4 (2), 60–80.

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

Chen, T. Y., Ho, J. W. K., Liu, H., Xie, X., 2009. An innovative approach for testing bioinformatics programs using metamorphic testing. BMC bioinformatics 10 (1), 24.

Chen, T. Y., Poon, P.-L., Tang, S.-F., Tse, T. H., 2012. Dessert: a divide-and-conquer methodology for identifying categories, choices, and choice relations for test case generation. IEEE Transactions on Software Engineering 38 (4), 794–809.

Chen, T. Y., Poon, P.-L., Tse, T. H., 2003a. A choice relation framework for supporting category-partition test case generation. IEEE Transactions on Software Engineering 29 (7), 577–593.

Chen, T. Y., Tse, T. H., Zhou, Z. Q., 2002. Semi-proving: an integrated method based on global symbolic evaluation and metamorphic testing. In: Proceedings of the ACM SIGSOFT International Symposium of Software Testing and Analysis (ISSTA 2002). IEEE Computer Society Press, Rome, Italy, pp. 191–195.

Chen, T. Y., Tse, T. H., Zhou, Z. Q., 2003b. Fault-based testing without the need of oracles. Information and Software Technology 45 (1), 1–9.

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

Conrad, M., Krupp, A., 2006. An extension of the classification-tree method for embedded systems for the description of events. Electronic Notes in Theoretical Computer Science 164 (4), 3–11.

Cortes, C., Jackel, L. D., Chiang, W.-P., 1995. Limits on learning machine accuracy imposed by data quality. In: Proceedings of the 1st International Conference on Knowledge Discovery and Data Mining (KDD 1995). Vol. 95. AAAI Press, Montreal, Quebec, Canada, pp. 57–62.

Gotlieb, A., Botella, B., 2003. Automated metamorphic testing. In: Proceedings of the 27th Annual International Computer Software and Applications Conference (COMPSAC 2003). IEEE Computer Society Press, Dallas, TX, USA, pp. 34–40.

Groce, A., Kulesza, T., Zhang, C., Shamasunder, S., Burnett, M., Wong, W.-K., Stumpf, S., Das, S., Shinsel, A., Bice, F., McIntosh, K., 2014. You are the only possible oracle: effective test selection for end users of interactive machine learning systems. IEEE Transactions on Software Engineering 40 (3), 307–323.

Grochtmann, M., Grimm, K., 1993. Classification trees for partition testing. Software Testing, Verification and Reliability 3 (2), 63–82.

Hamlet, R., 2002. Encyclopedia of Software Engineering, 2nd Edition. Wiley Online Library, Ch. Random testing, pp. 970–978.

Harman, M., McMinn, P., Shahbaz, M., Yoo, S., 2013. A comprehensive survey of trends in oracles for software testing. Tech. Rep. CS-13-01, Department of Computer Science, University of Sheffield.

Hierons, R. M., 2012. Oracles for distributed testing. IEEE Transactions on Software Engineering 38 (3), 629–641.

Hierons, R. M., Harman, M., Singh, H., 2003. Automatically generating information from a z specification to support the classification tree method. In: Proceedings of the 3rd International Conference on Formal Specification and Development in Z and B (ZB 2003). Springer, Turku, Finland, pp. 388–407.

Jones, J. A., Harrold, M. J., 2005. Empirical evaluation of the tarantula automatic fault-localization technique. In: Proceedings of the 20th IEEE/ACM International Conference on Automated Software Engineering (ASE '05). ACM Press, pp. 273–282.

Kanewala, U., Bieman, J. M., 2013. 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 (ISSRE 2013). IEEE Computer Society Press, Pasadena, CA, USA, pp. 1–10.

Kruse, P. M., Luniak, M., 2010. Automated test case generation using classification trees. Software Quality Professional Magazine 13 (1), 4–12.

Lehmann, E., Wegener, J., 2000. Test case design by means of the CTE XL. In: Proceedings of the 8th European International Conference on Software Testing, Analysis and Review (EuroSTAR 2000). Kopenhagen, Denmark.

Liu, H., Kuo, F.-C., Towey, D., Chen, T. Y., 2014a. How effectively does metamorphic testing alleviate the oracle problem? IEEE Transactions on Software Engineering 40 (1), 4–22.

Liu, H., Liu, X., Chen, T. Y., 2012. A new method for constructing metamorphic relations. In: Proceedings of the 12th International Conference on Quality Software (QSIC 2012). IEEE Computer Society Press, Xi'an, Shaanxi, China, pp. 59–68.

Liu, H., Yusuf, I. I., Schmidt, H. W., Chen, T. Y., 2014b. Metamorphic fault tolerance: An automated and systematic methodology for fault tolerance in the absence of test oracle. In: Companion Proceedings of the 36th International Conference on Software Engineering (ICSE Companion 2014). ACM Press, Hyderabad, India, pp. 420–423.

Lyu, M. R., Horgan, J., London, S., 1994. A coverage analysis tool for the effectiveness of software testing. IEEE Transactions on Reliability 43 (4), 527–535.

Mayer, J., Guderlei, R., 2006. An empirical study on the selection of good metamorphic relations. In: Proceedings of the 30th Annual International Computer Software and Applications Conference (COMPSAC 2006). Vol. 1. IEEE Computer Society Press, Chicago, IL, USA, pp. 475–484.

Mishra, K. S., Kaiser, G., 2012. Effectiveness of teaching metamorphic testing. Tech. Rep. CUCS-020-12, Department of Computer Science, Columbia University.

Mishra, K. S., Kaiser, G. E., Sheth, S. K., 2013. Effectiveness of teaching metamorphic testing, part ii. Tech. Rep. CUCS-022-13, Department of Computer Science, Columbia University.

Morell, L. J., 1990. A theory of fault-based testing. IEEE Transactions on Software Engineering 16 (8), 844–857.

Murphy, C., Raunak, M., King, A., Chen, S., Imbriano, C., Kaiser, G., Lee, I., Sokolsky, O., Clarke, L., Osterweil, L., 2011. On effective testing of health care simulation software. In: Proceedings of the 3rd Workshop on Software Engineering in Health Care (SEHC 2011). ACM Press, pp. 40–47.

Myers, G. J., Sandler, C., 2004. The art of software testing. John Wiley & Sons.

Nie, C., Leung, H., 2011. A survey of combinatorial testing. ACM Computing Surveys 43 (2), 11:1 – 11:29.

Poon, P.-L., Kuo, F.-C., Liu, H., Chen, T. Y., 2014. How can non-technical end users effectively test their spreadsheets? Information Technology and People 27 (4).

Poon, P.-L., Tse, T. H., Tang, S.-F., Kuo, F.-C., 2011. Contributions of tester experience and a checklist guideline to the identification of categories and choices for software testing. Software Quality Journal 19 (1), 141–163.

Segura, S., Hierons, R. M., Benavides, D., Ruiz-Cortés, A., 2011. Automated metamorphic testing on the analyses of feature models. Information and Software Technology 53 (3), 245–258.

Singh, H., Conrad, M., Sadeghipour, S., 1997. Test case design based on z and the classification-tree method. In: Proceedings of the 1st IEEE International Conference on Formal Engineering Methods (ICFEM 1997). IEEE Computer Society Press, Hiroshima, Japan, pp. 81–90.

Tao, Q., Wu, W., Zhao, C., Shen, W., 2010. An automatic testing approach for compiler based on metamorphic testing technique. In: Proceedings of the 17th Asia Pacific Software Engineering Conference (APSEC 2010). IEEE Computer Society Press, Sydney, Australia, pp. 270–279.

White, L. J., Cohen, E. I., 1980. A domain strategy for computer program testing. IEEE Transactions on Software Engineering 6 (3), 247–257.

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

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

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