A UNIQUE TESTING NAMED METAMORPHIC TESTING FOR SOFTWARE QUALITY, VERIFICATION, VALIDATION AND QUALITY ASSESSMENT: AN APPROACH TO SEARCH ENGINES

K.B.V.K.NAGASREE,
Assistant Professor, Department of Information Technology, Mahaveer Institute ofScience and Technology, Hyderabad, Telangana, India,
nagasree_k@yahoo.com

V.NARESH KUMAR REDDY
Assistant Professor, Department of Information Technology, Mahaveer Institute ofScience and Technology, Hyderabad, Telangana, India,
vnkreddy12@gmail.com

ABSTRACT: *A Unique Testing named Metamorphic testing is a technique used to identify the functional conformity of software in the deficit of an ideal oracle. This paper enhances metamorphic testing into a user friendly access to software verification, validation, and quality assessment, and conducts dominant pragmatic studies with major web search engines: eg: Google,. This search engine is very crucial for testing and assess using traditional ways by which it leads to the lack of an objective and generally recognized oracle. The results are useful for both search engine developers and users, and authenticates that our approach can productively amend the oracle problem and challenges surrounding a lack of specifications when verifying, validating, and evaluating substantial and complicated software systems.*

Index Terms—Software quality, verification, validation, quality assessment, oracle problem, lack of system specification, metamorphic testing, user-oriented testing, search engine

INTRODUCTION

THE goal of software engineering practices is to develop high quality software. It is therefore crucial to develop evaluation methods for various types of software qualities [1]. Testing is a widely used approach for evaluating soft- ware qualities and helping developers to find and remove software faults. The majority of software testing techniques assume the availability of an *oracle*, a mechanism against which testers can verify the correctness of the outcomes of test case executions [2]. In some situations, however, an oracle is not available or is available but is too expensive to be used—a situation known as the *oracle problem*, a fundamental challenge for software testing.

A *metamorphic testing* (MT) method has been developed to alleviate the oracle problem [3], [4], [5]. Unlike conventional testing methods, MT does not focus on the verification of each individual output, but instead checks the relationships among the inputs and outputs of *multiple* executions of the program under test. Such relationships are known as *metamorphic relations* (MRs), and are necessary properties of the intended program's functionality: If an MR violation is detected, then a fault is said to be revealed. MT has been used to check the functional correctness of var- ious applications [6], [7], [8], [9], [10], [11], [12], [13], [14] and has also been applied to program proving and debugging [15], [16]. Its effectiveness has also been carefully studied [17], [18].

The present research extends metamorphic testing into a quantifiable approach for software quality assessment, which includes, but is not limited to, the verification and validation of software correctness. We applied our approach to alleviate the oracle problem for the testing and quality assessment of (web) search engines. Search engines are software systems designed to search for information on the World Wide web, and are the main interface through which people discover information on the Internet; web searching is one of the most popular functionalities of the Internet, second only to email [19]. As more and more services and data are being made available on the Internet, search engines are becoming increasingly important. In today's highly competitive search market, it is imperative

that search engines provide the desired result according to the queries entered. It is, however, extremely difficult to assess some key qualities of these search engines. T he empirical study results. Section 6 further discusses sev- eral important software engineering issues and puts this work in context by examining some fundamental software quality models used in quality control. Section 7 summarizes the paper and provides future research directions.

1 PRELIMINARIES

In this section, the background information and basic concepts of our approach are introduced and explained. A summary of the contributions of this research is also presented.

1.1 Search Engine Software Characteristics and User Validation Difficulties

Software testing can be performed for different purposes, with verification and validation being the core tasks. Fig. 1 sketches typical verification and validation activities in conventional software development projects [21]. The white arrows represent verification activities, which check the consistency between two descriptions (that is, the consistency of an implementation with its specification). The shaded arrows (*a*, *b*, *c*, *d*, *e*, and *f*) represent validation activities, where the software systems, designs, and specifications are tested or reviewed by actual users to assess the degree to which they meet the users' real needs. Pezzè and Young [21] further pointed out that validation activities refer primarily to the overall system specification and the final code.

Software validation for search engines is much harder than the conventional validation activities depicted in Fig. 1. This is because the vast majority of search engine users do not have access to the intermediate products. For search engines, therefore, software validation is limited to activity *a* only, where the users can only use an online search service with- out knowledge about its design. For users, search engines are representative of a large body of software products that come without specifications (or with incomplete specifications) [22]. The lack of knowledge about the software design and specifications details, coupled with the oracle problem discussed in Section 1, makes user validation extremely difficult, if not impossible: It is hard for the users to decide if, and to what degree, the search engines are appropriate for their specific information needs.

Search engine developers usually do not publicize detailed specifications of their systems for two main reasons: First, the design and algorithms are commercial secrets and, therefore, should not be released to the general public; and second, because of the complexity of the software systems. Owing to the unprecedented scale of the web, the unique characteristics of the web documents and web users, as well as the realtime requirements, the design of the software systems (and subsystems and modules) of search engines entails many practical considerations and tradeoffs, such as between quality and speed; quality and quantity; response time and storage; and precision and flexibility. The developer has to consider these things when designing software systems and modules, and often has to choose an algorithm/design from multiple candidates, each of which has its advantages and limitations. The developer's choice might be validated at the individual module's level, but when the modules are integrated to form systems, the interplay of the different factors involved in the considerations and tradeoffs can become so complex that it can be impossible for the developer to clearly explain to the end users. In fact, through our interactions with colleagues in industry, we found that it can often be hard even for the developers themselves to explain or to verify certain search engine behavior.

Without access to a system specification, the online help pages (referred to as *online specifications* hereafter) are the only type of specification available to the users. However,

these are usually very brief, and only explain how to use the basic search operators, without actually revealing any technical design detail of the software systems. They are in no way equivalent to the "system specification" shown in Fig. 1, defined as "an adequate guide to building a product that will fulfill its goals" [21]. For instance, Bing online specification states "We design algorithms to provide the most relevant and useful results"— how the relevance or usefulness is determined is not specified. This lack of technical detail makes it difficult for users to validate the search engine for their specific needs as different users can have very different criteria of relevance and usefulness of results.

1.2 Our Approach: User-Oriented Testing

In this paper, we present a testing approach that alleviates the difficulties in search engine verification, validation, and quality assessment. Our approach is based on the following observation: A search is always performed in the context of a particular scenario, and involves certain specific functions, which are only a very small set of all functions offered by the search engine. Therefore, *the user does not need to understand the system in its entirety in order to validate the search engine*; instead, he/she only needs a testing technique that tells him/her whether or not the few functions directly involved in the search can deliver what he/she wants. When the test fails, it can either indicate a fault in the implemented software system or a deficiency in the algorithm(s) chosen by the search engine developer for validation purposes, the user does not need to distinguish between these two cases.

Based on the above observation, we propose a user oriented testing approach using the concept of MT. Our approach demonstrates the feasibility of MT being a unified framework for software verification, validation, and quality assessment. It is user-oriented because we propose to make use of MRs defined from the users' perspective: The users can define MRs as necessary properties of what they would expect a "good" search engine to have, to meet their specific needs, irrespective of how the search engine was designed. In other words, the MRs can be designed based on the users' expectations to reflect what they really care about, not based on the algorithms/designs chosen by the developer (which are unknown to the users anyway)—note that this view is different from that of conventional MT, where MRs are identified based on the target algorithm to be implemented [3], [4], [5]. From this perspective, our approach performs software validation.

In addition to validation, our approach can also be used to do verification—because the MRs can be designed not only based on the user's expectation, but also based on the online specification shown to the user (the search engine's online help pages). Testing the search engine against such an MR is therefore a check of consistency between the online specification and the implementation and, hence, is verification.

Our approach can also be used for software quality assessment beyond verification and validation. This is because the software qualities that we consider include, but are not limited to, functional correctness.

Our user-oriented testing approach, however, is different from previous work on end user software engineering [23], [24], where the user, tester, and developer are the same person (known as the "end-user programmer"), such as a secretary or a scientist writing some programs to support their own work. In this situation, the user knows the details of the algorithms or formulas to be implemented and, hence, can identify MRs based on these algorithms or formulas to verify the implementation. The user also has access to the implemented source code. In the context of the present research, however, the user has no knowledge about the designs/algorithms/formulas of the software at all, and the software is treated as a sheer

blackbox.

1.3 Metamorphic Testing (MT)

MT [4], [5], [18], [25] alleviates the oracle problem by testing against MRs, which are necessary properties of the software under test (SUT). MRs differ from other types of correctness properties in that an MR is a relationship among *multiple* executions of the target program. As a result, even if no oracle is available to verify each individual output, we can still check the multiple outputs of the SUT against the given MR. If the MR is violated for certain test cases, a failure is revealed. Consider, for instance, a program *pðG; a; bÞ* purportedly calculating the length of the shortest path between nodes *a* and *b* in an undirected graph *G*. When *G* is nontrivial, the program is difficult to test because no oracle can be practically applied. Nevertheless, we can perform MT. Let *ðG1; a1; b1Þ* and *ðG2; a2; b2Þ* be two inputs, where *G2* is a permutation of *G1* (that is, *G2* and *G1* are isomorphic), and (*a2*, *b2*) in *G2* correspond to (*a1*, *b1*) in *G1*. Then an MR can be identified as follows: *pðG1; a1; b1 Þ¼ pðG2; a2; b2Þ*. A metamorphic test using this MR will run *p* twice, namely, a *source execution* on the *source test case* *ðG1; a1; b1Þ* and a *follow-up execution* on the *follow-up test case* *ðG2; a2; b2Þ*. Many other MRs can also be identified, such as *pðG; a; bÞ¼ pðG; b; aÞ*, and so on [26].

MT was originally proposed as a *verification* technique, where an MR is a necessary property of the algorithm to be implemented. Therefore, a violation of the MR reveals a fault in the implementation. When studying MT for machine learning software, Xie et al. [8] coincidentally discovered that an MR can be a necessary property of the target software rather than the selected algorithm in this case, even if the selected algorithm has been correctly implemented, the MR can still be violated if the algorithm has some deficiency and cannot meet the user's expectation for the target software. In this situation, MT is used for *validation*. Nevertheless, the study conducted by Xieetal. was small scale, and at the algorithm selection level (checking whether the adopted algorithm was appropriate or not). In our research, we used MT for software validation at the top level (the system/service level) and conducted very large scale empirical studies. Further, we extended MT into a *quality assessment* method.

Using MT to test search engines was first proposed by Zhou et al. [27], [28], where some logical consistency relationships among multiple responses of the search engines were employed. For instance, such a relationship, an MR, is that the number of web pages satisfying *c1* should be less than or equal to the number of web pages satisfying *c1 or c2*, where *c1* and *c2* are two search criteria. Zhou et al. con- ducted pilot studies to test the functional correctness of key- word-based search of Google, Yahoo! and Live Search [27], [28]. Imielinski and Signorini [29] essentially also used the logical consistency relationships to test "how semantic" a search engine is. They employed a (metamorphic) relation that a "truly semantic search engine" should return the same results for two semantically equivalent queries. For instance, the queries "biography of George Bush" and "find me bio of George Bush" should return the same results when submitted to a semantic search engine. It is to be noted that Zhou et al. [27], [28] only considered keyword- based search with a focus on the functional correctness, whereas Imielinski and Signorini [29] only considered semantic search with a focus on how the search engines could imitate the behavior of a human operator.

More recently, MT techniques have also been implicitly used to detect search engine censorship [30], [31], [32]. In these studies, the tester first identified some query terms politically sensitive to the Chinese government. These query terms were then sent to the search engine under test multi- ple times using different languages, such as English, complex Chinese, and simplified Chinese characters. The search results for the

different languages were then compared. The US based search engine www.bing.com was found to return very different results when the queries were in simplified Chinese characters. For instance, it was reported that "If you search a term on Bing that is politically sensitive in China, in English the results are legitimate. Conduct the search in complex Chinese characters (the kind used in Tai- wan and Hong Kong) and on the whole you still get authentic results. But conduct the search with the simplified characters used in mainland China, then you get sanitized pro-Communist results. . . . this is true wherever in the world the search is conducted—including in my office in New York" [30]. Discussion of the validity of these studies is beyond the scope of this paper, but it is worth noting that Microsoft has acknowledged the presence of "error" and "bug" in Bing that caused some of these problems [30], [31], [32], [33]. The above testing method is essentially MT where the source test case is a query that contains a politically sensitive word or phrase in English (such as "Dalai Lama"), and the follow-up test case is another query having the same meaning but typed in Chinese characters. The search engine's responses for the source and follow-up test cases are then compared against predefined logical consistency relationships.

2 A DESCRIPTION OF THE IDENTIFIED MRS

To apply MT to the automatic quality assessment of search engines, without the need for an oracle or human assessor, two groups of MRs were used: The "No Missing web Page" group assesses the search engines' capability in retrieving appropriate web pages to meet the users' needs; and the "Consistent Ranking" group assesses the ranking quality of the search results. This section provides a brief description of these MRs. Their validity and the experimental design will be discussed in Section 4.

2.1 Metamorphic Relation: MPSite

MPSite belongs to the "No Missing web Page" group of MRs, which assess the search engine's web page retrieval capability. MPSite is focused on the search engine's reliability when retrieving web pages that contain an exact word or phrase. It therefore assesses the keyword-based search feature. MPSite is described as follows:

Let A be a *source query* for which the search engine returns a non-empty list of results (called the *source response*), namely, $\eth p1; p2; \ldots; pn \text{Þ}$, where $0 < n$ and p_i is a web page from domain d_i, $1 \leq i \leq n$. To enhance accuracy and validity of our approach, in MPSite we only consider situations where $0 < n \leq 20$ so that we can avoid the inaccuracy associated with large result sets (such as a large list being truncated by the search engine to improve response time).

For the source response $\eth p1; p2; \ldots; pn \text{Þ}$, n *follow- up queries* are constructed as follows: The ith follow-up query B_i $(1 \leq i \leq n)$ is constructed in such a way that B_i is identical to A except that B_i includes an additional criterion which requires that all results be retrieved from domain d_i. Let FR_i (a *follow-up response*) be the list of web pages returned by the search engine for query B_i. The metamorphic relation MPSite requires that p_i 2 FR_i (note that there is no requirement on the ranking of p_i in FR_i). For example, let us test Google by issuing the following source query:

"side effect of antibiotics in babies"

where the quotation marks are part of the query. Google returned a total of seven web pages.

This webpage is from the uk domain. The metamorphic relation MPSite enables the construction of the following fol- low-up query: ["side effect of antibiotics in babies" site:uk],[2] where "site:" is a Google search operator that specifies domains(see Fig.2(lower)).Obviously, the previously returned top result meets this search criterion, is indexed in Google database, and therefore should still be returned by Google for this follow-up query. In this

example, Google returned a total of seven web pages for the source query. Therefore, seven follow-up queries are constructed by referring to MPSite.[3]

Using MPSite, even if the assessor is unable to verify or evaluate each individual response, he/she can still

3 MR VALIDITY AND EXPERIMENT DESIGN

This section discusses the validity of the MRs identified in Section 3. For each MR, we also identify some basic usage patterns and evaluation metrics, and explain the design of the experiments, which are summarized in Table 1 and will be explained in the following.

3.1 MPSite

We next discuss the validity of MPSite, and then design the experiments.

earch (using double quotation marks) to avoid ambiguity; and (iii) avoid large result lists—in our experiments, the source response never contains more than 20 results. In situations where the initial attempt returns more than 20 results, additional words are generated and appended into the double quotation marks of the query until the number of returned web pages is less than or equal to 20. For example, consider a query term "tempted," for which Google returned 57,000,000 results. This result count is too large and, therefore, an additional word *peaceably* was selected to change the query term from "tempted" into "tempted peaceably." As shown in Fig. 3a, Google returned "About eight results" for "tempted peaceably." Since 8 < 20, the query term "tempted peaceably" serves as a valid source query, and the initially attempted query term (for which the result count was 57,000,000) is dis- carded. In situations where the search returned zero results, the query was discarded, and a new query generated. Thus, we avoid the inaccuracy and other problems associated with large result sets.

1) In this research, all queries were constructed using correct syntax as per the online specifications of the search engines under test (for instance, see Fig. 2).

2) Because search engines may truncate queries which are too long, or contain too many words, in all our experiments, we ensured that the query length was well within limits.

3) Because search engines filter results, which may cause inaccuracies and inconsistency, we disabled all filters in our experiments. Furthermore, all cookies, history, cache, active logins, and so on, were also cleared, to avoid personalized results. Advertisements were not counted either. Search engines some- times display a message such as: "In order to show you the most relevant results, we have omitted some entries very similar to the xxx already displayed. If you like, you can repeat the search with the omitted results included." Whenever this happened during the experiments, the option "repeat the search with the omitted results included" was always selected, which effectively disabled a filter.

Fig. 3 shows an example of Google search failure detected by our automated testing tool using MPSite. The source query is ["tempted peaceably"] (Fig. 3a). According to Google online specification (Fig. 2), quota- tion marks are used to search for an exact phrase. This "is helpful when searching for song lyrics or a line from a book." Google returned eight results, the top one from books.google.com.au (see Fig. 3a). Using MPSite, our test- ing tool immediately constructed a follow-up query ["tempted peaceably" site:au] and sent it to Google again. According to MPSite, the web page (books.google. com.au/...) must still be retrieved. However, this time Google reported "No results found" (see Fig. 3b. Note that the message "About 44,000 results" is not relevant as it is for a suggested search without quotes.) The source and follow-up queries were issued consecutively within a very short period of time from the same PC (this practice applied to all experiments reported in this paper).

4 RESULTS OF EXPERIMENTS

This section will present the empirical study results for each MR identified in Section 3. Both statistical and practical significance of the results will be analyzed.

4.1 Experiments with MPSite

We first identify the independent and dependent variables of the experiments, and then perform visual and statistical analyses of the results.

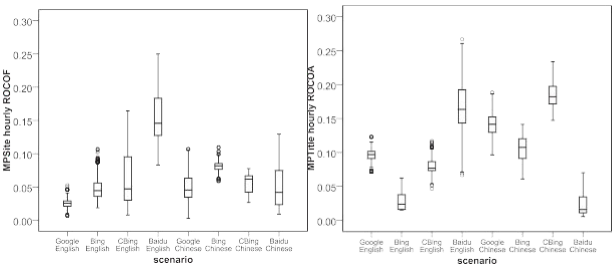
5.1.1 Independent and Dependent Variables

The independent variables of the experiments with MPSite include (1) usage pattern ("English queries" or "Chinese queries") and (2) search engine ("Google," "Bing," "Chinese Bing," or "Baidu"). Therefore, there are a total of 2 × 4 ¼ 8 combinations (scenarios) of independent variable values. The dependent variables are the ROCOF scores collected hourly for each of the eight scenarios throughout 379 hours of experiments. All eight scenarios were tested in the same hours (that is, at the same time).

For Google, about 1,000 pairs of source and follow-up responses were checked per hour for each language, and hence a total of approximately 379,000 different pairs were checked across the 379 hours for each Google usage pattern. For the other six scenarios (that is, Bing English, Bing Chi- nese, Chinese Bing English, Chinese Bing Chinese, Baidu English, and Baidu Chinese), in each scenario about 3,000 pairs of source and follow-up responses were checked per hour and, hence, approximately 3;000 × 379 ¼ 1;137;000 different pairs were checked across the 379 hours. (Google's test amount was lower because, during the experimental period, Google processed fewer queries per hour than did the other engines.) As a result, the empirical study with MPSite checked a total of approximately 379;000 × 2 þ 1;137;000 × 6 ¼ 7;580;000 pairs of source and follow-up responses.

5.1.2 Comparison of MPSite ROCOFScores

Fig. 9a shows the box plots of the distributions of the MPSite hourly ROCOF scores (the *y*-axis) for the eight scenarios (the *x*-axis), where "CBing" is short for Chinese Bing. A box plot shows the median, interquartile range, outliers and extreme values of the dataset. The top and bottom bars represent the maximum and minimum values, respectively (excluding outliers). The top and bot- tom of the box denote the third quartile (25 percent of data are greater than this value) and the first quartile (75 percent of data are greater than this value), respectively. The horizontal line inside the box represents the



(a) MPSite hourly ROCOF.

(b) MPTitle hourly ROCOA.

Fig. 9. Distributions of MPSite and MPTitle results.

Median. Circles and asterisks represent outliers. Using the software package SPSS, outliers are identified as cases that are more than 1:5 × *d* below the lower or above the upper hinge of the box, where *d* is the box length.

As explained in Section 3.1, MPSite is focused on the search engines' reliability for retrieving web pages containing an exact word or phrase. Lower ROCOF values indicate higher reliability of the search service. A visual analysis of Fig. 9a shows that all hourly ROCOF scores of all scenarios are above 0. This means that none of the search engines was perfect: Each search engine under each usage pattern produced failures in each and every hour. An interesting observation is that different search engines had very different performance. Relatively speaking, the most reliable service was Google's English search, whereas the least reliable service was Baidu's English search, whose hourly ROCOF reached as high

as around 0.25 in the worst case. Baidu's Chinese search, however, strongly outperformed its English search, whereas Google's and Bing's English search outperformed their corresponding Chinese search. These findings seem to agree with the fact that Baidu is a China-based search engine (and hence better designed for, or more trained in, Chinese queries than English queries), whereas Google and Bing are US-based engines with more English queries.

A one-way ANOVA (analysis of variance) confirmed that there is a statistically significant[9] and practically significant difference among the means of the eight scenarios ($p < 0{:}0005$, $h^2 ¼ 0{:}658$). *Partialeta squared* (h^2) is a measure of the *effect size* in ANOVA [39]. There are several effect size measures/estimates in ANOVA, such as omega squared, epsilon squared, and eta squared. But they tend to differ only slightly, especially with large samples as in the present study [40]. In one-way ANOVA, the values of partial eta squared and eta squared are the same, for which small, medium and large effects are generally considered to be 0.01, 0.06, and 0.14, respectively [39].

We further performed *post hoc multiple comparisons* to compare the means. Because equal population variances are not assumed, we used the Games-Howell procedure for the multiple comparisons as this procedure generally offers the best performance in this situation [41]. Games-Howell is also accurate when sample sizes are unequal [41]

Statistical analysis results are consistent with t analysis: A one-way ANOVA shows that there is a statistics been found that, in most situations, the query language had a statistically significant impact on the search engines' performance and that the performance of different search engines for the same language was also different with a statistical significance.

In the multiple comparisons, we also measured the practical significance (effect size) using Cohen's *d* [39]: *d* is the absolute value of the difference of the two means divided by the square root of the mean of the two variances. While h^2 estimates the effect size in one-way ANOVA, Cohen's *d* estimates the effect size when comparing two means. Cohen suggested the following benchmarks to interpret *d*: medium effect size ($d ¼ 0{:}5$) is "an effect of a size likely to be appar- ent to the naked eye of a careful observer," small effect size ($d ¼ 0{:}2$) is noticeably "smaller yet *not trivial*," and large effect size ($d ¼ 0{:}8$) is "the same distance above medium as small is below it" [42]. Therefore, we considered a *d* value of 0.20 or above to be significant (nontrivial). In our statistical analyses, all statistically significant differences are also found to be practically significant, and most of the *d* values are large (above 0.8).

5.2 Experiments with MP Title

The experimental procedure of MPTitle is very similar to that of MPSite, and the two sets of experiments were also conducted during the same period of time, except that MP Title experiments had 380 (not 379) hours of observations. Similar to MPSite, about 1,000 pairs of source and fol- low-up responses were checked per hour for Google English and Google Chinese, and about 3,000 pairs were checked per hour for the other six scenarios.

Fig. 9b shows the box plots of the distributions of hourly ROCOA scores for all eight scenarios. A visual analysis shows that none of the search engines was perfect as their hourly ROCOA scores are all above 0: They all produced failures or anomalies in each and every hour. The query language has a strong impact on the performance: Google English and Bing English continued to outperform Google Chinese and Bing Chinese, respectively. CBing English also outperformed CBing Chinese, indicating that CBing did not favor Chinese search even though it was designed for Chinese users. On the other hand, Baidu Chinese continued to of returning similar results for similar queries.

5.3.1 Independent and Dependent Variables

The independent variables include (1) usage

pattern, which is "people," "companies," or "drugs," and (2) search engine, which is "Google," "Bing," "Chinese Bing," or "Baidu." Therefore, there are 3 × 4 ¼ 12 scenarios. The dependent variables are the average Jaccard coefficients calculated hourly for each of the 12 scenarios. More details of the experiments are given in Table 2. Because of resource limitations, there are some differences in the number of hours and number of test case pairs between different search engines, as shown in Table 2. Nevertheless, for the same search engine, the three sets of experiments for the three usage patterns were conducted at exactly the same hours.

5.3.2 Comparison of MPReverse JDJaccardCoefficients

Fig. 10 shows the box plots of the distributions of the MPReverseJD hourly mean Jaccard coefficients for all 12 scenarios. A higher coefficient implies better stability in web page retrieval. A visual analysis of Fig. 10 shows that all hourly scores of all search engines are below 1. This means that none of the search engines was perfect in any hour. Among all four engines, Google was the most stable whereas Baidu was the least stable. The usage pat- tern (word category) appears to have an impact on the search engines' performance.

A one-way ANOVA confirmed that there is a statistically and practically significant difference among the means of the 12 scenarios ($p < 0{:}0005$, $h^2 \text{¼} 0{:}759$). Post hoc multiple comparisons and effect size analysis show that the usage pattern (that is, the type of the query words) had a statistically and practically significant impact on the web page retrieval stability of all search engines. Furthermore, there are statistically and practically significant differences among the search engines' performance under every usage pattern, where Google and Baidu were found to be the most and least stable search engines, respectively.

A further question is: What is the difference among the three types of words that makes the stability of their search results significantly different? Every test case used for MPReverseJD consists of names. For a search engine to process such a query, *named entity recognition* techniques are probably used [43]. It was reported that the levels of difficulty in recognizing these three types of named enti- ties (people, companies, and drugs) are different [44]. The frequencies of occurrence of these three types of names in user queries are very different, too [43], which might have caused the search engines' machine learning software to be trained differently. Furthermore, company names nor- mally have good commercial and advertising values, and drug names also have such values to a certain degree. Per- sonal names are more neutral. The above factors, when combined, may have influenced the quality of search for different types of names.

5 DISCUSSIONS

In this section, we discuss several important software engineering issues, namely, MR identification, test case selection, causes of failures, software quality models, and implications of the empirical study results.

5.1 Identification of Metamorphic Relations

The identification of MRs requires knowledge of the prob- lem domain, and is therefore a manual process. In this study, we manually identified five MRs by referring to the online specifications of the search engines. These five MRs were selected because they address two types of crucial qualities (page retrieval and ranking) that are difficult to assess using conventional methods owing to the oracle problem. It is however not the intention of this study to identify an exhaustive list of MRs; instead, we studied the effectiveness of using MT as a software quality assessment method with the four search engines and five MRs. The study shows that our method is effective. By referring to the online specifications, it would not be difficult to iden- tify additional MRs to cover other features and search operators which are increasingly being

supported by mod- ern search engines. As a general rule, we should consider two situations when attempting to identify a comprehensive list of MRs.[10] In the first one, the follow-up input is only related to the source input without referring to the source output. For instance, MPReverseJD and SwapJD are both this kind of MR, where the follow-up query is constructed by reordering the source query's terms. In the other situation, the follow- up input is related to both the source input and the source output. Examples of this type include MPSite, MPTitle, and Top1Absent, where the follow-up query is constructed by not only referring to the source query's terms but also refer- ring to the domains or titles of the search results included in the source response. The above observation may provide hints for the development of methods and tools to assist software testers with the identification of useful MRs.

5.2 Selection of Test Cases

The effectiveness of MT not only depends on the MR but also on the source test cases. Obviously, if a source test case leads to an incorrect output (though unrevealed because of the oracle problem), it would be likelier to trigger a violation of the MR.

MRs for software quality assessment, we restricted our- selves to use randomly sampled words as source test cases in our experiments. This practice is also in line with the reliability estimation principle that requires unbiased samples for valid statistics [21].

A question then arises: To what degree do the test results based on random words reflect the search engines' actual performance for human-generated query terms? To answer this question, we need to consider both verification and validation. For software verification purposes, random testing is quite cost-effective for fault detection when compared with other techniques [46], [47]. It is simple in concept and easy to implement, and the random test cases can often reveal unexpected failures. The large number of failures and anomalies detected in the present research provide further evidence of the fault-detection capability of random testing.

For the purpose of validation, random sampling of test cases following a usage-profile-based probability distribution is often preferred [47]. There are, however, practical reasons why we did not adopt this approach in the present study. First of all, the usage profiles (such as the user logs) of the search engines under test were not available to us. Second, even if we were able to collect such a usage profile, given that we test four different search engines together, it would be unfair to test one search engine using another search engine's usage profile. Needless to mention, for the same search engine, its users also have different usage pro- files. Third, user interests change quickly over time; hence yesterday's popular query terms may no longer be popular today. Our random sampling strategy, which does not refer to the system logs, is a simple solution to the above problems and provides a common ground for fair comparison of different systems.

A further question is: Are the randomly generated query terms really useful/meaningful from the users' perspective? Or are they just meaningless random strings? To answer this question, we should first note that different users have different information needs. Even a random string that is useless/meaningless for the majority of users can be very useful for a searcher looking for a particular product model number or zip code [28]. Indeed, it is hard to say which queries are not important. It is therefore reasonable for the users to expect that the search engine works for all queries regardless of whether they are "popular" or not.

Furthermore, although this research adopted a random sampling strategy for test case generation, the query terms thus generated are by no means meaningless random strings. For MPSite and MPTitle, a source query is a

phrase enclosed by double quotation marks—such a phrase is meaningful because, first, it consists of valid English or Chinese words and, second, the design of the experiments guarantees that the source response always contains one or more results, which means that the phrase is part of some web documents and is hence meaningful. According to Google specification (Fig. 2), users are likely to issue this type of query when searching for song lyrics or a line from a book/article. The follow-up query of MPTitle further includes the page title generated by the search engine, such as [Indianapolis Correspondence Google News] as shown in Section 3.2.2, which is meaningful and mimics human-generated input.

For MPReverseJD, three types of names (that is, named entities) were used as test cases, which are meaningful, too. According to an analysis conducted by Guo et al. [48], about 71 percent of real-world queries contained named entities. People search and company search are common types of searches; drug search is a specific type of search that may be issued by patients and medical personnel.

For SwapJD, the following three word lists were used to generate test cases: *Swhere*, *Swhen*, and *Swhat*. These three lists were selected to ensure that (1) any pairwise combination is meaningful and falls within the common types of searches that users perform, and (2) swapping the word order in the query will not change the meaning of the query. The generation of these three sets of words involved manual inspection of randomly sampled words to ensure the satisfaction of the above two requirements.

Finally, for Top1Absent, each source query contains one word randomly sampled from an English dictionary, excluding common words such as "of." Every English word is meaningful; so is the query.

It should be noted, however, that reliability estimation (as part of validation) is always usage-profile dependent. Different user groups may have different usage profiles as they use the search engine in different ways. Scientific researchers, for instance, may frequently look for research papers by issuing queries containing scientific terminologies; whereas prospective students may frequently search for course information on the .edu domain. Developers, on the other hand, may have a usage profile based on the system's user logs. As a result, different stakeholders may get different reliability estimates for the same search engine. As an independent tester, we used a random sampling strategy in the present research. For verification purposes, our strategy has been very effective in failure detection. For validation purposes, our strategy has also been very effective in revealing the search engines' behavioral differences under different operational profiles. In Section 6.3, we will further show that our approach complements the developer's user log based approaches to software quality assurance.

They were able to repeat and confirm some of our reported software failures.

Colleagues at Google indicated that the reported cases could have been caused by software related factors and that further investigation was under way.

In summary, our approach can be used for verification, validation, and quality assessment of the software systems of the search engines. Without more details of the design and algorithms used (which are commercial secrets), it is hard for a user or an independent tester to tell whether a detected failure is caused by a fault in the implemented code or a flaw in the design or algorithm. But, from the users' perspective, it does not matter, because users are only concerned about the quality of the final operational software, regardless of whether the fault is in the code or in the design. To this end, our approach enables the users to perform validation and quality assessment of the software. Developers, on the other hand, know the details of the algorithms adopted, and therefore can identify MRs for these

algorithms, verify the implementation using these MRs, and find the root cause of detected failures.

6 SUMMARY AND FUTURE WORK

Metamorphic testing was initially proposed as a verification technique, where metamorphic relations were identified by referring to the target algorithm to be implemented [3], [5]. In this paper, we have demonstrated the feasibility of MT being a unified framework for software verification, validation, and quality assessment. We conducted a study on search engines, where we identified MRs from the users' perspective without referring to the target algorithms or system specifications. More generally, this approach allows users to recognize whether or not a system is appropriate for their specific needs in the absence of complete software documentation, which is often the case with web services, poorly evolved software, and open source software.

We have applied our approach to assess several key soft- ware qualities of search engines under different operational profiles in the absence of an objective and generally recognized oracle. All ANOVA analyses returned statistically significant results with large effect size (h^2) values. Most multiple-comparison results also had a statistical and practical significance (with large Cohen's *d* values in most situations), indicating that our approach is effective. We have also discussed the investigated software qualities in the framework of the software quality model standard ISO/IEC 25010.

The empirical results demonstrate that our approach is useful for both developers and users. First, our approach can effectively detect various kinds of failures. Second, we found that the operational profiles have a significant impact on the quality of search. For a given search engine, its quality of search can be significantly different for different query languages, different types of query words, and different domains being searched. This finding provides a hint for the developers to identify the strength and weakness of their systems, and is also useful for the users to choose a suitable search engine or to better construct their queries. The ability to automatically detect failures and anomalies using MRs can also provide hints for the construction of run-time self-correction mechanisms, which will be a future research topic.

Similar to the use of a set of mutants to assess the effectiveness of a test suite in the context of mutation analysis, a set of properly selected MRs can potentially be used to assess certain software quality characteristics. Further study on this topic will be a future research area.

ACKNOWLEDGMENTS

The authors are grateful to Microsoft Research for supporting a pilot study of this work under an academic research collaboration scheme. They wish to thank Baidu Inc. Google Inc. for discussions on their work. They would also like to thank Dave Towey, Antony Tang, and Lei Wang for their comments. This research was supported in part by a linkage grant of the Australian Research Council (Project ID: LP100200208).

REFERENCES

[1] *C. Ghezzi, M. Jazayeri, and D. Mandrioli, Fundamentals of Software Engineering, 2nd ed. Englewood Cliffs, NJ, USA: Prentice-Hall, 2002.*

[2] *R. M. Hierons, "Oracles for distributed testing," IEEE Trans. Softw. Eng., vol. 38, no. 3, pp. 629–641, May/Jun. 2012.*

[3] *T. Y. Chen, S. C. Cheung, and S. M. Yiu, "Metamorphic testing: A new approach for generating next test cases," Dept. Comput. Sci., Hong Kong Univ. Sci. Technol., Hong Kong, Tech. Rep. HKUST- CS98-01, 1998.*

[4] *T. Y. Chen, T. H. Tse, and Z. Q. Zhou, "Semi-proving: An inte- grated method based on global symbolic evaluation and meta- morphic testing," in Proc. ACM SIGSOFT Int. Symp. Softw. Testing Anal., 2002, pp. 191–195.*

[5] *T. Y. Chen, T. H. Tse, and Z. Q. Zhou, "Fault-based testing with-out the need of oracles," Inf. Softw. Technol., vol. 45, no. 1, pp. 1–9, 2003.*

[6] *T. Y. Chen, J. W. K. Ho, H. Liu, and X. Xie, "An innovative approach for testing bioinformatics programs using metamorphic testing," BMC Bioinformat., vol. 10,*

no. 24, p. 24, 2009.

[7] C. Murphy, K. Shen, and G. E. Kaiser, "Automatic system testing of programs without test oracles," in Proc. 18th ACM SIGSOFT Int. Symp. Softw. Testing Anal., 2009, pp. 189–200.

[8] X. Xie, J. W. K. Ho, C. Murphy, G. Kaiser, B. Xu, and T. Y. Chen, "Testing and validating machine learning classifiers by metamor- phic testing," J. Syst. Softw., vol. 84, pp. 544–558, 2011.

[9] J. Zhang, J. Chen, D. Hao, Y. Xiong, B. Xie, L. Zhang, and H. Mei, "Search-based inference of polynomial metamorphic relations," in Proc. 29th IEEE/ACM Int. Conf. Autom. Softw. Eng., 2014, pp. 701–712.

[10] V. Le, M. Afshari, and Z. Su, "Compiler validation via equivalence modulo inputs," in Proc. 35th ACM SIGPLAN Conf. Program. Lang. Des. Implementation, 2014, pp. 216–226.

[11] J. Regehr. (2014, Jun. 20). Finding compiler bugs by removing dead code [Online]. Available: http://blog.regehr.org/archives/ 1161

[12] A. Núñez and R. M. Hierons, "A methodology for validating cloud models using metamorphic testing," Ann. Telecommun., vol. 70, no. 3-4, pp. 127–135, 2015.

[13] S. Segura, A. Durán, A. B. Sánchez, D. L. Berre, E. Lonca, and A. Ruiz-Cortés, "Automated metamorphic testing of variability anal- ysis tools," Softw. Testing, Verification Rel., vol. 25, pp. 138–163,

[14] D. Lo, J. Li, L. Wong, and S.-C. Khoo, "Mining iterative generators and representative rules for software specification discovery," IEEE Trans. Knowl. Data Eng., vol. 23, no. 2, pp. 282–296, Feb. 2011.

[15] T. Y. Chen, F.-C. Kuo, and Z. Q. Zhou, "An effective testing method for end-user programmers," in Proc. Workshop End- User Softw. Eng., 2005, pp. 21–25.

[16] A. J. Ko, R. Abraham, L. Beckwith, A. Blackwell, M. Burnett, M. Erwig, C. Scaffidi, J. Lawrance, H. Lieberman, B. Myers, M. B. Rosson, G. Rothermel, M. Shaw, and S. Wiedenbeck, "The state of the art in end-user software engineering," ACM Comput. Surveys, vol. 43, no. 3, pp. 21:1–21:44, 2011.

[17] T. Y. Chen, F.-C. Kuo, D. Towey, and Z. Q. Zhou, "Metamorphic testing: Applications and integration with other methods," in Proc. 12th Int. Conf. Quality Softw., 2012, pp. 285–288.

[18] T. Y. Chen, D. H. Huang, T. H. Tse, and Z. Q. Zhou, "Case studies on the selection of useful relations in metamorphic testing," in Proc. 4th Ibero-Am. Symp. Softw. Eng. Knowl. Eng., 2004, pp. 569– 583.

[19] Z. Q. Zhou, T. H. Tse, F.-C. Kuo, and T. Y. Chen, "Automated functional testing of web search engines in the absence of an oracle," Dept. Comput. Sci., The Univ of Hong Kong, Hong Kong, Tech. Rep. TR-2007-06, 2007.

[20] Z. Q. Zhou, S. Zhang, M. Hagenbuchner, T. H. Tse, F.-C. Kuo, andT. Y. Chen, "Automated functional testing of online search serv- ices," Softw. Testing, Verification Rel., vol. 22, no. 4, pp. 221– 243, 2012.

[21] T. Imielinski and A. Signorini, "If you ask nicely, I will answer: Semantic search and today's search engines," in Proc. IEEE Int. Conf. Semantic Comput., 2009, pp. 184–191.

[22] N. Kristof, "Boycott Microsoft Bing," The New York Times, Nov. 20, 2009.

[23] S. Denyer, "Microsoft denies censoring results in the 'uncensored' version of Chinese-language Bing," The Washington Post, Feb. 12, 2014.

[24] P. Carsten, "Microsoft denies global censorship of China-related searches," Reuters, Feb. 12, 2014.

[25] A. Sohn. (2009, Nov. 20). Committed to comprehensive results [Online]. Available: http://bit.ly/6CD49e

[26] S. Morasca, D. Taibi, and D. Tosi, " T-DOC: A tool for the automatic generation of testing documentation for OSS products," in Open Source Software: New Horizons (series. I○FIP Advances in Information and Communication Technology), P. Agerfalk, C. Boldyreff, J. M. González-Barahona, G. R. Madey, and J. Noll, Eds. Berlin, Germany: Springer-Verlag, 2010, vol. 319, pp. 200–213.

[27] Quora. Will .es domainhurtmysearchresults [Online]. Available: http://www.quora.com/Will-es-domain-hurt-my-search-results, 2012.

[28] Y. Freund, R. Iyer, R. E. Schapire, and Y. Singer, "An efficient boosting algorithm for combining preferences," J. Mach. Learn. Res., vol. 4, pp. 933–969,2003.