Empirical Evaluation of Software Testing Techniques – Need, Issues and Mitigation

Sheikh Umar Farooq, SMK Quadri

Department of Computer Sciences, University of Kashmir
suf.cs@uok.edu.in, quadrsmk@kashmiruniversity.ac.in

Abstract- **Current scenario of software testing demands use of effective testing techniques which are practically possible. Though, at present, we have multitude of software testing techniques, which can reveal faults, but we do not have all the adequate practical information about them. Despite the number of studies which were conducted to evaluate these techniques, we are still without realistic and generalized results. This paper first explores the earlier studies on software testing techniques evaluation and identifies the problems associated with them. Based on the issues in these studies, we propose a set of guidelines which define a protocol to carry out such studies so that the issues identified are mitigated to a large extent. As a consequence, the results obtained are more realistic, generalized and comparable if these guidelines are followed while conducting the experiments.**

Keywords- **Comparison of Testing Techniques, Empirical Evaluation, Evaluation Guidelines, Experimentation, Software Testing Techniques.**

1. INTRODUCTION

Verification and validation activities are conducted to evaluate and enhance product quality throughout the entire cycle of software development. Verification and validation is a generic term which includes software testing. Software testing is an important phase in software development life cycle. Despite all the efforts people put in testing the software, effectiveness of testing remains lower than expectations. Testing is a widespread validation approach in the industry, but it is still largely adhoc, expensive, and unpredictably effective [1]. Unfortunately the lack of understanding of software testing usually leads to incomplete testing work. The choice of software testing technique in software testing influences quality of both process and the product. Taking into account current testing problems and failure consequences and cost associated with testing, using the most effective and efficient testing methods is most important need in testing. We have multitude of software testing techniques which makes testing technique selection a complex choice. When choosing a testing technique, practitioners want to know which one will detect the faults that matter most to them in the programs that they plan to test [2]. Which techniques should be chosen? Are there any particular benefits of using a specific technique? Which techniques are effective? Which are efficient? All testing techniques can reveal faults; but how effectively they do that and what kind of faults they find, how much resources they utilize, by what factor they increase the reliability, we do not have an exact answers to such questions. Although the utilization of these techniques is growing, we have very limited knowledge about their relative quantitative and qualitative statistics. Despite the large number of studies which attempt to study the testing techniques and its allied factors and conditions, we are not able to generalize the results as studies are not complete in all respects and vary significantly in terms of parameters they have taken into consideration. Additionally, many existing studies show contradictory results. There is no “silver bullet" testing approach and that no single technique alone is satisfactory has been pointed out by many leading researchers [3]. Therefore we should use a combination of several techniques to test software. We should select appropriate techniques which target different types of defects. However, many testing techniques belong to the same group and as such target same types of defects in the program. So we should use best candidate from each group to test a program but do we have knowledge of relative effectiveness and efficiency of techniques in a group? We guess, no!

It is obvious that testing technique selection and evaluation remains a key issue in software testing. So the need of the hour is to find the effective and efficient software testing techniques which can attain the goal of testing to maximum possible extent while consuming fewer resources. It would be appropriate if we first go for intra-family comparisons, then we can go for inter-family comparisons. As we know no single study can be complete and perfect in all respects, we cannot really expect one testing technique to supersede all other techniques. Regardless of our technique, it could be wise to try to understand what types of defects a particular technique can be expected to find and at what cost. We have to check whether testing technique effectiveness and efficiency depends on program to which it is applied, subject who applies it, the number of faults in the program or the type of faults in the program. However, it is not sufficient if testing techniques are only compared on fault detecting ability. They should also be evaluated to check which among them enhances reliability. To establish a useful theory for testing, we need to evaluate

existing and novel testing techniques not only for defect detection effectiveness and efficiency but also for their ability of enhancing software reliability. In this paper, we describe why there is a need to evaluate software testing techniques, identify problems with the existing studies and a framework (a set of guidelines) is proposed for carrying out such studies so that in future such studies show results which can be useful for testing professionals in particular and testing industry in general.

The paper is organized as: Section 2 gives us a brief description of software testing techniques. Section 3 explains why we should evaluate software testing techniques. Section 4 surveys the existing research on software testing techniques evaluation. Evaluation results of the surveyed studies are presented in subsection 4.1 and the problems and shortcomings with the existing studies are presented in subsection 4.2. Current status of testing technique selection is presented in Section 5. Section 6 proposes a framework (a set of guidelines) which should be taken into consideration while evaluating testing techniques. Section 7 presents the conclusions and future work.

2. SOFTWARE TESTING TECHNQIUES

We test software by selecting appropriate testing techniques and applying them. Testing techniques refer to different methods of testing particular features of a computer program, system or product. By a testing technique, we mean a method or approach that systematically describes how a set of test cases should be created (with what intention and goals) keeping into consideration possible rules for applying the test cases. Testing techniques aid in limiting the number of test cases that need to be created, since it will be targeting a specific type of input, path, fault, goal, measurement etc. [4]. Test techniques provide an understanding of the complexities imposed by most systems. Using testing techniques, we can reproduce tests, as it paves the path for creating a test ware. Some techniques are easy to apply while other techniques require a little experience and knowledge before they can be used. We have to understand when to use them and we should have a clear knowledge of which ones to use in any given situation. Before utilizing different testing techniques in proper manner for an appropriate purpose, we should have a profound knowledge of these testing techniques.

3. WHY EVALUATE SOFTWARE TESTING TECHNIQUES?

Software testing should be effective enough to prevent critical damages to the whole system for users, by taking into consideration potential failures of the program and its environments [5]. One way to avoid such failures is to go for exhaustive testing of the system, which tests the system with all possible combinations of inputs which includes both valid and invalid cases. However, excluding trivial cases, exhaustive testing is an impractical thing for most software systems. Besides, we are often have limited time and resources, which can limit our ability to effectively complete testing efforts. We do not want to go for exhaustive testing, rather we wants to select a testing technique in relation to the selected test strategy that will detect maximum possible critical faults and brings the product to an acceptable level while consuming less resources and time. Whether we opt for static or dynamic testing, there is a selection of testing methods to choose from. In each testing method there are so many testing techniques that can be used to test a system. Each testing technique meant for testing has its own dimensions i.e. for what purpose it is used, what aspect it will test, what will be its deliverables etc. Different approaches to software development require different testing methods and techniques [6]. This limits our ability to use a generic technique for testing a system. So at present we prefer to use variety of testing techniques to test a system as it will ensure that a variety of defects are found, resulting in more effective testing. But how long will we use numerous testing techniques to test software. Going this way means excessive use of resources (less efficiency), as using many testing techniques clearly implies more test cases, more time and more resources. So the need is to select appropriate testing techniques which can make testing process effective and efficient. However, for a given testing problem, there exist several techniques of the same kind which differ by the underlying mechanism. Among so many techniques, which are in a competition we would like to select a technique that will detect the maximum possible significant defects, while consuming less resources and time. Unfortunately, it is not known which testing technique to select as we do not have adequate information about relative effectiveness, efficiency and cost of testing techniques. [7] also states that we do not have all the information of interest about every testing technique. This kind of information can only be obtained by evaluating software testing techniques. It is also necessary to understand what types of defects a particular technique is expected to find and at what cost. We also need to analyze testing technique dependencies on program to which it is applied, subject who applies it, the number of faults in the program or the type of faults in the program. We should evaluate testing techniques to know about the relative merits and limitations of each testing technique, so that we are able to use it in appropriate scenario and for appropriate purpose. This information is useful before one has to implement a given testing technique; it is also useful (as a post mortem analysis) when one is finished with test-

ing as this post-implementation assessment and analysis is needed for subsequent improvement of the technique to increase its effectiveness [8].

4. EXISTING RESEARCH ON SOFTWARE TESTING TECHNIQUES EVALUATION

A lot of research has been carried out in the field evaluation of software testing techniques. By studying the major research results that have contributed to the growth of software testing techniques we can analyze the maturation of software testing techniques research. We can also assess the change of research paradigms over time by tracing the types of research questions and strategies used at various stages [9]. Three directions of research have been found related to evaluation of testing techniques:

1. Actual evaluations and comparisons of testing techniques based either on analytical or empirical methods.
2. Evaluation frameworks or methodologies for comparing and/or selecting testing techniques.
3. Surveys of empirical studies on testing techniques which have summarized available work and have highlighted future trends.

Many experiments and case studies have been conducted so far towards the goal of evaluation of testing techniques. Some significant and relevant controlled experiments are Hetzel [10], Myers [11], Basili and Selby [12], Weyuker [13], Beiman and Schultz [14], Frankl and Weiss [15], Hutchins et al. [16], Offutt and Lee [17], Kamsties and Lott [18], Wong and Mathur [19], Offutt et al. [20], Frankl et al. [21], Roper et al. [22], Wong et al. [23], Frankl and Iakounenko [24], Rothermal and Harrold [25], Vokolos and Frankl [26], Rothermal et al.[27], Elbaum et al. [28], Kim et al.[29], Bible et al.[30], Graves et al.[31], Juristo and Vegas [32], Juristo et al.[33] and many more. A comprehensive survey was conducted by [34] which surveys experiments comparing software testing techniques up to year 2004. Runeson et al. presents a survey of empirical studies on defect detection techniques which includes experiment as well as case studies [35]. They also define questions regarding defect detection techniques evaluation and interpret the findings in terms of practical use. Case studies which studied and evaluated different techniques include Concardi et al. [36], Aurum et al [37], Host et al. [38] and Berling and Thelin [39]. Juristo et al. in their paper "In search of what we experimentally know about unit testing" argue that aggregation of the results is not easy because separate experiments have different settings and subjects, tend to assess different variables, and are usually trying to optimize different phenomena [40]. They believe that aggregation results are not always as conclusive as we may desire. In effect, they aggregated results from unit-testing experiments with the aim of identifying information with some experimental basis that might help practitioners make decisions based on extensive search in IEEE Xplore and ACM digital libraries looking for publications of such experiments. Juristo et al. in "A look at 25 years of data" conclude that although results extracted so far from the experiments conducted are interesting, however, they seem to indicate that research in this area has focused on specific questions and hypotheses rather than on building a larger picture of available techniques and when to select them [41]. In case of frameworks and methodologies [42] describe a characterization scheme for experiments which is specific for software testing techniques. The schema is similar to [43] but adapted to deal with evaluating testing techniques. [44] and [45] defined the SIR (Software Artifact Infrastructure Repository) infrastructure to support controlled experiments with software testing techniques. The main contribution of their work is a set of benchmark programs that can be used to evaluate testing techniques. [46] describe a straightforward framework for the comparison of the efficiency, effectiveness and applicability of different testing techniques based on fault injection or seeding. [47] define a general methodological framework for evaluating software testing techniques, which focuses on the evaluation of effectiveness and efficiency, but the framework is very preliminary and needs significant improvement. We tried to include all significant experimental studies; however, we cannot guarantee that the list is complete. In addition to these, many studies are in progression, but their results are preliminary. These studies are also not included in the list.

Our focus in this paper is on empirical evaluation of testing techniques, so we will have a look at empirical studies conducted so far to evaluate testing techniques. During the past few decades, a large number of empirical evaluations of numerous testing techniques have been executed to compare various software testing techniques. The research on the comparison of testing technique traces back to as early as 37 years ago with Hetzel making a start in 1976 by conducting a controlled experiment in order to analyze three defect detection methods [10]. The empirical research on testing techniques is largely carried out through experiments comparing different dynamic testing techniques with each other or with different types of static testing techniques usually some reading technique. Most of the experimental studies are performed in a unit level testing context. A laboratory setting with student subjects is the most common design in the existing experiments. Industrial studies are very rare. The most commonly studied

factors in the experiments evaluating testing techniques are their effectiveness (i.e., number of detected defects) and efficiency (i.e., effort required to apply the technique) in programs. [34] identified two classes of evaluation studies on testing techniques; intra-family and inter-family. Based on this distinction, major intra family and inter family studies carried out till date to evaluate software testing techniques are listed in Table 1 and Table 2 respectively.

TABLE 1
INTRA-FAMILY COMPARISONS

Technique	Study	Year
Data-flow testing techniques	Weyuker Bieman and Schultz	1990 1992
Mutation testing techniques	Offut and Lee Wong and Mathur Offut et al.	1994 1995 1996
Regression testing techniques	Rothermel and Harrold Vokolos and Frankl Kim et al. Bible et al. Graves et al.	1998 1998 2000 2001 2001

TABLE 2
INTER-FAMILY COMPARISONS

Comparison Groups	Study	Year
Control-flow, data-flow and random techniques	Frankl and Weiss Hutchins et al. Frankl and Iakounenko	1993 1994 1998
Functional and structural techniques. Note: All studies also compare manual static testing technique. Code Reading with functional and structural techniques.	Hetzel Myers Basili and Selby Kamsties and Lott Roper et al. Juristo and Vegas Juristo et al.	1976 1978 1987 1995 1997 2003 2012
Mutation and data-flow techniques	Wong and Mathur Frankl et al.	1995 1997
Regression and improvement testing techniques	Wong et al. Rothermel et al. Elbaum et al. Kim et al. Graves et al.	1998 1999 2000 2000 2001

4.1 Evaluation results

Summarizing the results of the studies conducted to evaluate the software testing techniques. We observed that studies unfortunately have a lot of contradiction in terms of their results. The results also are very inconclusive and do not reveal much information. [41] also states that experimental results are conflicting, and the experiments lack a formal foundation and studies have a lot of difference between parameters they have taken into consideration. From the analysis of previous studies, we can conclude that the experimental studies on software testing techniques conducted so far does not provide a basis for making any strong conclusions regarding different software testing techniques. As a result we cannot generalize results of software testing techniques evaluation. Current studies point out that various other factors, in addition to the applied testing technique, have a strong effect on the results of defect finding effectiveness and efficiency. Even though the experiments are designed to study the effects of one or more selected testing techniques, the effects of all other factors cannot be excluded. The defect detection effectiveness and efficiency seems to depend on the person who test the software, the software being tested, and the actual defects that exist in the software. We can summarize the results of empirical studies on testing techniques as follows:

1. There is no clear, consistent evidence that one fault finding technique is stronger than others, rather the evidence to date suggests that each technique has its own merits and demerits.

2. While some studies conclude that technique A is ranked higher than technique B. Some studies conclude technique A and technique B find different kinds of defects, and are as such complementary.

3. The effectiveness of verification activities is low; only 25-50% of the defects are found using inspection, and 30-60% using testing.

4. Combining testing techniques uncovered more defects than did a single technique.

5. Combining individual testers seem to increase defect detection effectiveness more than combining test case design techniques. This fact was also reported by [22] [34].

6. Defect detection effectiveness highly depends on the individual differences between testers. The variation between individuals seems to be greater than the variation between techniques as the different testers seem to find clearly different defects despite using the same technique.

7. Defect detection effectiveness seems to be correlated with the amount of test cases.

8. The effectiveness of different techniques seems to depend on the type of software tested and the types of the actual defects in the software.

9. It seems that some types of faults are not well suited to some testing techniques.

10. There appears to be a relationship between the programs, or the type of faults entered in the programs, and technique effectiveness.

4.2 Problems with existing studies

We must first clearly recognize the issues in any field in order to make it more mature by resolving those issues. After so many years of empirical investigation, we are still without definite results. In a look at 25 years of data, the authors have reached the same conclusion after studying various experiments on software testing [41]. In addition they found that it is really difficult to compare different experiments; however, they do not present any solution to it. [48] discussed many issues facing empirical studies of testing techniques; criteria to quantify fault-detection ability of a technique is one such issue, while threats to validity arising out of the experimental setting (be it academic or industrial) is another. [49] and [50] has highlighted following issues with current studies:

1. Informality of the results analysis (many studies are based solely on qualitative graph analysis).

2. Limited usefulness of the response variables examined in practice, as is the case of the probability of detecting at least one fault.

3. Non-representativeness of the programs chosen, either because of size or the number of faults introduced.

4. Non-representativeness of the faults introduced in the programs.

5. We believe that there are many reasons for this inadequacy of knowledge and restricted results regarding the evaluation of software testing techniques. After analyzing the existing studies on software testing techniques evaluation, we conclude that most of the existing studies on evaluation of testing techniques have following problems:

4.2.1 Experimentation problems

1. Comparing testing techniques is to quantify fault detection effectiveness and efficiency. A comparison criterion for testing techniques is usually not well defined. [7] states that in the context of testing technique selection, the term best has different meanings depending on the person making comparisons.

2. Existing studies mostly differ in the number and type of parameters they have used in their study. A common standard is missing which makes it difficult to compare these studies.

3. Most of the studies do not take all the parameters necessary for comparison into consideration, as a result of that one technique do not supersede other techniques on all fronts; thereby creating ambiguity in test technique selection.

4. The inconclusive results also indicate the presence of factors that were not under experimental control. Factors seem to be at work that aren't measured or controlled but that nonetheless influence defect detection methods' performance [35]. It might be useful to investigate "softer" factors such as motivation and satisfaction and in particular to apply methods in practice, monitor them, and follow up. The compara-

tive study of the effectiveness of different techniques should be supplemented by studies of the fault types that each technique detects and not only the probability of detecting faults. That is, even if T1 and T2 are equally effective, this does not mean that they detect the same faults [50]. This would provide a better understanding of technique complementary, even when they are equally effective.

5. Fault seeding is used in most of the experiments. The advantage is that we can seed large numbers faults which results in very less random variation in fault ratios and more statistical power. However the disadvantage is that it often results in seeding unrealistic faults and we may only seed faults of a particular type. As a consequence, our results are mostly invalid or incomplete.

6. The experiments are often biased either towards academic or industrial system, as they are usually carried out with only academic or industrial settings into consideration. Most of the studies conducted so far are academic in nature. In all studies, subjects have not been chosen properly according to given scenario. Even though experiment conducted by [12] took into account several classes of professionals. Most experiments conducted the studies in student environments, which limit the transfer of experiment results to real world. Studies are less validated or hardly put to practice in testing industry. Studies in an academic setting are often a first step before studies are carried in industrial settings [41] [51]. So we should take both systems into consideration while carrying out such experiments. Therefore, we need to balance between academic and industry perspective.

7. Another major problem with testing technique evaluation is that experiments are mostly made on a small sample (code sample selection is in majority below 2K), and often with the demonstration that they either perform better than another specific technique. The main reason for this is the difficulty to get large amount of real data to perform research on. The number of faults on sample of this size may not be large enough to allow for quantitative, statistical analysis.

4.2.2 Knowledge problems

1. Existing studies do not tend to share the knowledge they acquire by using a testing technique with others [7]. The information related to these studies is not fully available which makes it difficult for researchers or industry professionals in drawing exact results from diverse studies. Also it becomes difficult to replicate the work already done. Everyone is going its own way, starting things from very beginning. It would have been good to validate and extend the earlier studies so that results can be generalized and implemented at industry level.

2. Usually the main focus is often to invent a special technique and compare it with one already known (often a similar technique). Little attention is given to evaluate already existing techniques which could have served professionals better.

3. The problem with testing techniques in industry is that their power is not known to all the testers, since often there is a belief of their efficiency and effectiveness, and it is seldom proven for larger complex system. The actual research setting of creating reasonable comparative models have not been totally explored. Also studies revealed that testers are often not trained in testing, but on system behavior [7]. Even if people are formally trained in test techniques, they easily fall back to approaching testing from a system usage viewpoint rather than applying a test technique, since requirements on testers are seldom assessed as long as they find some failures.

5. WHERE DO WE STAND AT THIS MOMENT?

The big achievement we had from conducting so many experimental experiments is that we do know with certainty that the usage of a testing technique is better than none, and that a combination of techniques is better than just one technique. We also know that the use of testing techniques supports systematic and meticulous work and that techniques are good for finding possible failures. No firm research conclusions exist about the relative merits of software testing techniques. The conclusions we drew may only apply to their specific experimental environment and are not general enough to be applied to other research environments, let alone to software testing industry [52]. Most of the research that has been performed is very academic and not terribly useful in the real testing world. At present we do not have adequate proof of any technique superseding other ones in terms of effectiveness or efficiency. Not only we have limited knowledge about numbers, we also lack the information regarding the types of faults.

How should one choose testing technique at present? Current studies suggest it is just not sufficient to rely on a single method for catching all defects in a program. Actually each technique is good for certain things, and not as good for other things. Each individual technique is aimed

at particular types of defect as well. For example, state transition testing is unlikely to find boundary defects. Some techniques are more applicable to certain situations and test levels; others are applicable to all test levels. Some testing techniques are never considered for use at all and others are used over again in different software projects without even examining, after use, whether or not they were really suited [53]. One conclusion that seems to have been reached is: There is no best technique. Each testing technique is good at finding some specific classes of defects, using just one technique will help ensure that many defects of those particular classes are found. Unfortunately, it may also help to ensure that many defects of other classes are missed! Using a variety of techniques will therefore help ensure that a variety of defects are found, resulting in more effective testing. However, it will also ensure the excessive use of resources which will in turn result in less efficiency. So it is argued that more experimental work is required to evaluate testing technique so that our testing will be both effective and efficient. We need to know how to evaluate testing methods, how much effective are testing techniques in terms of effort and defect finding capability as we always want to select a testing technique that will bring the product to an acceptable level. Recent surveys on comparisons of various software testing techniques also concludes that further empirical research in software testing is needed, and that much more replication has to be conducted before general results can be stated [34] [41]. But the experimentation should be carried out in such a way so that results can be realistic and with very less contradictions. Then only we can have firm knowledge about the effectiveness and efficiency of the testing technique in revealing faults, the classes of faults for which the technique is useful, and other allied aspects; then only can we apply the results in the real world.

6. PROPOSED GUIDELINES FOR SOFTWARE TESTING TECHNIQUES EVALUATION

The knowledge for selecting testing techniques should The knowledge for selecting testing techniques should come from studies that empirically justify the benefits and application conditions of the different techniques [50]. Basili et al. in their book “Empirical Software Engineering Issues - critical assessment and future directions” which contains a collection of excellent papers focused on empirical software engineering explains past successes and failures, assess the current state of the practice and research, identify challenges and issues, discuss promising opportunities and define future directions and roadmap for research, practice, education and training [54]. Briand also highlights many issues most commonly encountered while performing empirical studies of software testing techniques [55]. One issue that is raised by all studies is that we need to carry out experimentation on a large scale using a common benchmark/framework. A common standard is also required to standardize the evaluation process of such experiments. Empirical studies on large scale artifacts, within real world contexts, and replicated by several professional testers to attain generally valid results would be, of course, prohibitively expensive. A possible way out to overcome such difficult challenges could be that of combining the efforts of several research groups, currently conducting separate experimentations, and join their forces to carry out a widely replicated experiment, i.e., factorize a large experiment in pieces among several laboratories [57]. Bertolino’s idea is similar to that of launching an “Open Experiment" initiative, similarly to how some Open Source projects have been successfully conducted. However, not all open source projects are necessarily successful, and experimentation, to be credible, needs very careful planning and control.

Empirical software engineering research needs research guidelines to improve the research and reporting processes. There exists a real need in industry to have guidelines on which testing techniques to use for different testing objectives, and how usable these techniques are [47]. Apart for guidelines or a common standard required for reporting experiments like one proposed by [56], we need guidelines for carrying for out experiments to mitigate the issues reported so far in ESE. Here we present a framework (a set of guidelines) how experiments/studies for evaluating the testing techniques should be carried out so that definite, practical and comparable results about relative merits of testing techniques can be achieved. The proposed guidelines are general since no assumptions about the testing technique, subjects and a program have been made.

1. Studies should be carried on a set of common systems or at least similar systems. It will make comparisons of techniques much easier. [58] If is states that comparison of a testing technique is only possible to measure if you can compare two techniques for the same set (i.e. software) under same set of conditions.

2. Studies should be carried out on a large sample and on real data (most preferably on industrial data). The studies should be carried in industrial settings to extend the generalizations derived from them. The number of detected faults should be sufficient enough for quantitative, statistical analysis. Carrying out experiments on such data will draw results that will be near to perfection if not perfect. We need to build some standardized and better laboratory packages which should represent actual software engi-

neering practices. Carry out experiments on such packages will help in deriving realistic results. A better idea will be that a software engineering groups like ESERNET, SERG, FRAUNHOFER CENTER FOR EXPERIMENTAL SOFTWARE ENGINEERING etc helps in creating some standard experimental repositories preferably in open source fashion.

3. Carrying out experimentation on actual faults is more realistic from an external validity standpoint. However, extracting proper results from such results often is time consuming as we are unaware about the number of actual faults present in product. In this method detailed fault information can be expensive to collect. On the other hand fault seeding allows us to seed as many faults as necessary in order to work with a sample that is large enough to be amenable to statistical analysis. However, seeding does not always seed realistic faults. However, if we still want to go for fault seeding, we need an unbiased, systematic, and inexpensive way to do so. We need to investigate and define procedures to seed faults for the purpose of experimentally assessing test techniques. So it is advisable to use both methods on different systems, this way we can achieve more concrete and applicable results.

4. We can perform software testing experiments either by using human subjects or by using simulation. Using first one allows us to access other factors like cost effectiveness and human applicability, Test suites are perhaps more realistic, as derived by human subjects performing the actual test tasks; while second one allows us to test software rigorously (100% coverage) as we can generate large number of test cases, accounting for random variation but this technique can suffer from biasing problem if not implemented properly. It is preferable to use human subjects as mostly techniques are applied using humans subjects or at least require some human knowledge. Simulation however should not be discarded. If we will use both techniques the role and domain of each should be well defined.

5. Criteria for comparing testing techniques should take into consideration many parameters. We cannot expect experiments to be perfect with respect to all factors which need to be taken into consideration while evaluating software testing techniques. But we should strive towards carrying out experiments which take into consideration maximum factors while evaluating testing technqiues. Taking into account diverse parameters yield more appropriate results and makes testing techniques selection more valid and unambiguous. One observation that immediately jumps out is that researchers and developers must publish more information on the types, not just the numbers, of faults that the techniques can remove [41]. Some of factors necessary for comparison are

   a. Number of faults
   b. Fault rate
   c. Fault type
   d. Size (test case generated)
   e. Coverage
   f. Time (Usually it is execution time)
   g. Software/program type.
   h. Experience of subjects
   i. Reliability improvement

6. Experimental details should be shared. Besides results, experiments should lead to an experimentation package that would allow other researchers to easily replicate experiments. Ideally, it should contain all the necessary material to perform the experiment and should be publicly available in open source fashion. This would allow the research community to converge much faster towards credible results.

7. We should balance all dimensions of validity to achieve trustworthy empirical studies (i.e. the balance between internal (researcher point of view) and external (Practitioner point of view) validity). Studies in academia are often strong in terms of internal validity (i.e. our capability to draw proper conclusions from the data) and weak as far as external validity is concerned (i.e. it is hard to know the extent to which you can generalize your results to industrial contexts). Field studies have exactly the opposite strengths and weaknesses. Both academic and field studies are necessary. Field studies are more suited to assess the difficulties to apply techniques in practice and to confirm the results obtained on real sets of faults.

8. The information available about the techniques is normally distributed across different sources of information (books, articles and even people) [53]. We should work towards building a sharable centralized depository on testing techniques. In addition, as [41] states that researchers and practitioners could help the field progress by encouraging a commitment to try promising research in more representative industrial conditions and to make that accessible industry and software engineering groups should try to develop a centralized and accessible repository which will contain programs/software's and all the related information which are realistic and truly represent industry.

7. CONCLUSION AND FUTURE WORK

Despite the general feeling that everything is changing fast, techniques do not usually change overnight. One sure thing that we came to know is that we have to do testing anyhow. With so many testing techniques and the very inadequate quantitative and qualitative knowledge about them, we strongly believe that there is a need to further evaluate software testing techniques. Presently, we are unaware about the absolute relative ordering of software testing techniques and if we are to make software testing more effective by selecting effective testing techniques then we need to place existing software testing techniques at least on an ordinal scale. Present situation calls for replication and further work on evaluation of software testing techniques so as to acquire the basic knowledge about the relative effectiveness and efficiency of software testing techniques for both fault finding and reliability criterion. To do so we need to carry out experimentation on large scale but that needs to be done in a way that can be compared and should have no contradictions. For that we also need to establish common and standard parameters so that there are little variations in the experimentation goals. We also need to find out dimensions on the basis of which we can agree that one testing method is more effective than another testing method.

REFERENCES

[1] A. Bertolino, "Software testing research: Achievements, challenges, dreams", *In Future of Software Engineering, 2007*. FOSE'07, pp. 85-103. IEEE, 2007.

[2] J. Goodenough and S. Gerhart, "Towards a theory of test data selection", *ACM SIGPLAN Notices*, 10, no.6, pp.493-510, 1975.

[3] H. Chu, "An evaluation scheme of software testing techniques", *Technical Report Series*, NC University, 1997.

[4] S Eldh, "On Test Design". PhD thesis, *Malardalen University Press*, October 2011. URL http://www.mrtc.mdh.se/index.php?choice=publications&id=2635.

[5] T. Kurokawa and M. Shinagawa, "Technical trends and challenges of software testing", http://www.nistep.go.jp/achiev/ftx/eng/stfc/stt029e/qr29pdf/STTqr2902.pdf. Last accessed on Jan 2012.

[6] A. Tawileh, S. McINTOSH, B. Work and W. Ivins, "The dynamics of software testing", *In Proceedings of the 25th System Dynamics Conference,* July, 2007.

[7] S. Vegas, "What information is relevant when selecting testing techniques", *In Proceedings of the 13th International Conference on Software Engineering and Knowledge Engineering*, pp. 45-52, 2001.

[8] A. Farooq and R. Dumke, "Evaluation approaches in software testing", *Univ.-Bibliothek, Hochschulschr.-und Tauschstelle*, 2008.

[9] L. Luo, "Software testing techniques", *Institute for software research international Carnegie mellon university Pittsburgh*, PA, 15232:1-19, 2001.

[10] W. Hetzel, "An experimental analysis of program verification methods", *ACM*, 1976.

[11] G. Myers, "A controlled experiment in program testing and code walkthroughs/inspections", *Communications of the ACM*, 21, no.9, pp.760-768, 1978.

[12] V. R. Basili and R. Selby, "Comparing the effectiveness of software testing strategies", S*oftware Engineering, IEEE Transactions on*, no.12, pp.1278-1296, 1987.

[13] E. Weyuker, "The cost of data flow testing: An empirical study", *Software Engineering, IEEE Transactions on*, 16, no.2, pp.121-128, 1990.

[14] J. Bieman and J. Schultz, "An empirical evaluation (and specification) of the all-du-paths testing criterion", *Software Engineering Journal,* 7, no.1, pp.43-51, 1992.

[15] P. Frankl and S. Weiss, "An experimental comparison of the effectiveness of branch testing and data flow testing", *Software Engineering, IEEE Transactions on*, 19, no.8, pp.774-787, 1993.

[16] M. Hutchins, H. Foster, T. Goradia, and T. Ostrand, "Experiments of the effectiveness of data flow and Control flow-based test adequacy criteria", *In Proceedings of the 16th international conference on Software engineering*, pp.191-200. IEEE Computer Society Press, 1994.

[17] A. Offutt and S. Lee, "An empirical evaluation of weak mutation", *Software Engineering, IEEE Transactions on*, 20, no.5, pp.337-344, 1994.

[18] E. Kamsties and C. Lott, "An empirical evaluation of three defect-detection techniques", *Software Engineering ESEC*'95, pp.362-383, 1995.

[19] W. Wong and A. Mathur, "Fault detection effectiveness of mutation and data flow testing", *Software Quality Journal*, 4, no.1, pp. 69-83, 1995.

[20] A. Offutt, A. Lee, G. Rothermel, R. Untch, and C. Zapf, "An experimental determination of sufficient mutant operators", *ACM Transactions on Software Engineering and Methodology (TOSEM)*, 5, no.2, pp.99-118, 1996.

[21] P. Frankl, D. Hamlet, B. Littlewood, and L. Strigini, "Choosing a testing method to deliver reliability", *In Proceedings of the 19th international conference on Software engineering*, pp. 68-78. ACM, 1997.

[22] M. Roper, M. Wood, and J. Miller, "An empirical evaluation of defect detection techniques", *Information and Software Technology*, 39, no.11, pp.763-775, 1997.

[23] W. Wong, J. Horgan, S. London, and H. Agrawal, "A study of effective regression testing in practice", In PROCEEDINGS *The Eighth International Symposium on Software Reliability Engineering*, pp.264-274. IEEE, 1997.

[24] P. Frankl and O. Iakounenko, "Further empirical studies of test effectiveness", *In ACM SIGSOFT Software Engineering Notes*, volume 23, pp.153-162. ACM, 1998.

[25] G. Rothermel and M. Harrold, "Empirical studies of a safe regression test selection technique", *Software Engineering, IEEE Transactions on*, 24, no.6, pp.401-419, 1998.

[26] F. Vokolos and P. Frankl, "Empirical evaluation of the textual differencing regression testing technique", *In Software Maintenance, 1998. Proceedings. International Conference on*, pp. 44-53. IEEE, 1998.

[27] G. Rothermel, R. Untch, C. Chu, and M. Harrold, "Test case prioritization: An empirical study", *In Software Maintenance, (ICSM'99) Proceedings. IEEE International Conference on*, pages 179-188. IEEE, 1999.

[28] S. Elbaum, A. Malishevsky and G. Rothermel, "Prioritizing test cases for regression testing", *ACM SIGSOFT Software Engineering* Notes, 25, no.5, pp.2000.

[29] J. Kim, A. Porter and G. Rothermel, "An empirical study of regression test application frequency", *In Software Engineering, 2000. Proceedings of the 2000 International Conference on*, pp.126-135. IEEE, 2000.

[30] J. Bible, G. Rothermel, and D. Rosenblum, "A comparative study of coarse-and fine-grained safe regression test-selection techniques", *ACM Transactions on Software Engineering and Methodology (TOSEM),* 10, no.2, pp.149-183, 2001.

[31] T. Graves, M. Harrold, J. Kim, A. Porter and G. Rothermel, "An empirical study of regression test selection techniques", *ACM Transactions on Software Engineering and Methodology (TOSEM)*, 10, no.2, pp.184-208, 2001.

[32] N. Juristo, and S. Vegas, "Functional testing, structural testing and code reading: what fault type do they each detect?" *Empirical Methods and Studies in Software Engineering*, pp. 208-232, 2003.

[33] N. Juristo, et al. "Comparing the Effectiveness of Equivalence Partitioning, Branch Testing and Code Reading by Stepwise Abstraction Applied by Subjects", *Software Testing, Verification and Validation (ICST), 2012 IEEE Fifth International Conference on*. IEEE, 2012.

[34] N. Juristo, A. Moreno, and S. Vegas, "Reviewing 25 years of testing technique experiments", *Empirical Software Engineering*, 9, no.1, pp.7-44, 2004.

[35] P. Runeson, C. Andersson, T. Thelin, A. Andrews, and T. Berling, "What do we know about defect detection methods?" [Software testing]. *Software, IEEE*, 23, no.3, pp.82-90, 2006.

[36] R. Conradi, A.S. Marjara, and B. Skåtevik, "An Empirical Study of Inspection and Testing Data at Ericsson, Norway," *Proc. 24th NASA Software Eng. Workshop, NASA,* 1999; http://sel.gsfc.nasa.gov/website/sew/1999/topics/marjara_SEW99paper.pdf.

[37] A. Aurum, H. Petersson and C. Wohlin, "State-of-the-art: *software inspections after 25 years", Software Testing, Verification and Reliability,* 12, no.3, pp.133-154, 2002.

[38] M. Host, C. Wohlin and T. Thelin, "Experimental context classification: incentives and experience of subjects", *In Software Engineering, 2005. ICSE 2005. Proceedings. 27th International Conference on*, pp. 470-478. IEEE, 2005.

[39] T. Berling and T. Thelin, "An Industrial Case Study of the Verification and Validation Activities," *Proc. 9th Int'l Software Metrics Symp., IEEE CS Press*, pp. 226–238, 2003.

[40] N. Juristo, A. Moreno, S. Vegas & M. Solari, "In search of what we experimentally know about unit testing", *Software, IEEE*, 23, no.6, pp.72-80, 2006.

[41] A. Moreno, F. Shull, N. Juristo and S. Vegas, "A look at 25 years of data", *IEEE Software,* 26, no.1, pp.15-17, 2009.

[42] S. Vegas and V. Basili, "A characterisation schema for software testing techniques", Empirical Software Engineering, 10, no.4, pp.437-466, 2005.

[43] V. R. Basili, R. Jr, Selby and D. Hutchens, "Experimentation in software engineering", *Technical report*, *DTIC Document*, 1985.

[44] H. Do, S. Elbaum and G. Rothermel, "Infrastructure support for controlled experimentation with software testing and regression testing techniques", *In Empirical Software Engineering, 2004. ISESE'04. Proceedings. 2004 International Symposium on*, pp. 60-70. IEEE, 2004.

[45] H. Do, S. Elbaum and G. Rothermel, "Supporting controlled experimentation with testing techniques: An infrastructure and its potential impact", *Empirical Software Engineering*, 10, no.4, pp.405-435, 2005.

[46] S. Eldh, H. Hansson, S. Punnekkat, A. Pettersson and D. Sundmark, "A framework for comparing efficiency, effectiveness and applicability of software testing techniques", *In Testing: Academic and Industrial Conference-Practice And Research Techniques, 2006. TAIC PART 2006. Proceedings*, pp. 159-170. IEEE, 2006.

[47] T. Vos, B. Martin, I. Panach, A. Baars, C. Ayala and X. Franch, "Evaluating software testing techniques and tools", *Proceedings of JISBD 2011*, ISBN: 978-84-9749-486-1, pp. 531-536, 2011.

[48] L. Briand and Y. Labiche, "Empirical studies of software testing techniques: Challenges, practical strategies, and future research", *ACM SIGSOFT Software Engineering Notes*, 29, no.5, pp. 1-3, 2004.

[49] N. Juristo, A. Moreno and S. Vegas, "A survey on testing technique empirical studies: How limited is our knowledge", *In Empirical Software Engineering, 2002. Proceedings. 2002 International Symposium,* pp. 161-172. IEEE, 2002.

[50] N. Juristo, A. Moreno and S. Vegas, "Limitations of empirical testing technique knowledge", *SERIES ON SOFTWARE ENGINEERING AND KNOWLEDGE ENGINEERING*, 12, pp.1-38, 2003.

[51] N. Juristo and A. Moreno, "Basics of software engineering experimentation". *Springer, 2001.*

[52] X. Yang, "Towards a self-evolving software defect detection process", *PhD thesis, University of Saskatchewan*, 2007.

[53] S.Vegas, "Identifying the relevant information for software testing technique selection", *In Empirical Software Engineering, 2004. ISESE'04. Proceedings. 2004 International Symposium on*, pages 39-48. IEEE, 2004.

[54] V. R. Basili, "The role of controlled experiments in software engineering research", *In Empirical Software Engineering Issues. Critical Assessment and Future Directions (pp. 33-37). Springer* Berlin Heidelberg, 2007.

[55] L. C. Briand, "A critical analysis of empirical research in software testing", *In Empirical Software Engineering and Measurement, ESEM 2007. First International Symposium on (pp. 1-8)*. IEEE, September, 2007.

[56] A. Jedlitschka, M. Ciolkowski & D. Pfahl, "Reporting experiments in software engineering", I*n Guide to advanced empirical software engineering (pp. 201-228). Springer London*, 2008.

[57] A. Bertolino, "The (im) maturity level of software testing", *ACM SIGSOFT Software Engineering Notes*, 29, no.5, pp.1-4, 2004.

[58] E. Weyuker, "Can we measure software testing effectiveness?", *In Software Metrics Symposium, 1993. Proceedings., First International*, pp. 100-107. IEEE, 1993.

ABOUT THE AUTHORS

Sheikh Umar Farooq is an Assistant Professor in Department of Computer Sciences at University of Kashmir, India. He received his Ph. D. in Computer Sciences from University of Kashmir. His research interests include empirical software engineering and software testing techniques' evaluation. He is a member of various software engineering societies like SIGSE, IACIST and IAENG.

SMK Quadri is a Professor in Department of Computer Sciences at University of Kashmir, India. He did his M. Tech. in Computer Application from Indian School of Mines (ISM), Dhanbad and Ph. D. in Computer Sciences from University of Kashmir. His research interests include software reliability, software testing and disk file systems. He is a member of Computer Society of India.