Oriental Journal of Computer Science & Technology Vol. **3**(1), 109-113 (2010)

Effectiveness of software testing techniques on a measurement scale

SHEIKH UMAR FAROOQ and S.M.K. QUADRI

Department of Computer Sciences, University of Kashmir, Srinagar (India).

(Received: April 12, 2010; Accepted: June 04, 2010)

ABSTRACT

Testing remains the truly effective means to assure the quality of a software system of non-trivial complexity. An important aspect of test planning is measuring test effectiveness. To make testing more successful we need to choose effective testing techniques. To compare testing techniques we need to place software testing techniques on a measurement scale which can define the relative merits of the existing testing techniques, but due to differences in software's and its allied parameters this task seems to be complex, if not impossible

Keywords: Software Testing, Testing Techniques, Measurement Scales, Effectiveness.

INTRODUCTION

Software Testing is the process of executing a program with the intent of finding errors[1,2]. Testing is focused only on finding faults.[3] advocates that the primary goal of testing should be to measure the dependability of tested software. Effective software testing will contribute to the delivery of reliable and quality oriented product, more satisfied user, lower maintenance and more correct and reliable results. One surprising thing about programs is not that they often have bugs, but that they have so many bugs. As the number of bugs assumed in software is decreased, the effectiveness of testing will be greatly enhanced. Due to large number of testing limitations like Exhaustive (total) testing is impossible, compromise between thoroughness, time and budget, it is impossible to be sure that we have removed each and every bug in the program. An effective testing technique will maximize the number of bugs founds during the test effort. We always want to choose a testing technique that will accomplish this goal while consuming less resources and time. To choose an effective testing technique we need to place software testing techniques onto a measurement scale.

Testing effectiveness

Test effectiveness is a measure of bug finding ability of the testing technique. Testing effectiveness can be measured by dividing the number of faults found in a given test by total number of faults (including those found after test)[4].

Test Effectiveness = Errors reported by Testers / Total Errors reported

Where Total Errors = Tester reported + User reported Errors

For instance, a testing technique A finds 30 errors, testing technique B finds 40 errors and the total testing process finds 100 errors. Then as per the given definition for measurement of test effectiveness, testing technique A was 30% effective and testing technique B was 40% effective. However, if the system is delivered after the total of 100 errors were found in whole testing process, and 50 additional errors were uncovered during the initial operation of the software for some specified time then contribution of testing technique A in finding errors is 30 out of 150 errors (20%) while that of testing technique B is 40 out of 150(27% appox.).

Test effectiveness is only concerned with finding bugs and is not concerned with whether these bugs are ever fixed[5]. Test effectiveness only measures the percentage of bugs that the test effort found. Some bugs will be found by both the testers and the users. They are counted only once. The more effective the testing is in finding those defects, the fewer will escape into operation.

This approach of evaluating the impact of a particular testing technique can be adjusted in several ways. For example, failures can be assigned a severity level, and test effectiveness can be calculated by a level. In fact the number of errors found in test effort is not meaningful as a measure until we combine it with the severity of errors, type of errors found and so on. In this way, we can say testing technique A might be 50% effective at finding faults that cause critical failures, but 80% effective at finding faults that cause minor failures.

Effectiveness of testing is allied to several parameters

- Testing technique.
- Software type.
- Error detection effectiveness (detection of most errors).
- Error detection cost (#errors/effort)
- Error type (Class of errors found: Critical, Serious, Medium and Low)
- Tester's experience etc.

Measurement Scales

Questions that are frequently asked regarding software testing effectiveness are: Which techniques are more valuable? Which are easy to apply? Which find genuine faults? These questions can only be answered when we can place software testing techniques on a scale of measurement.

If something cannot be measured, it can not be managed or improved. The arrangement of acceptable changes defines the measurement level of a scale. Usually, the narrower the arrangement of acceptable changes, the smaller the number of scales, and the more informative the scale. Five measurement levels are generally used in Measurement Theory. These Scales are below explained from the least to the most informative one.

Nominal Scale

A scale is a nominal one if it divides the set of entities into categories, with no particular ordering among them. For instance, if the attribute of interest is gender, subject (population) under consideration can be classified as males and females. The specific values of the measures do not convey any particular information, other than the fact that they are different, so they can be used as labels for the different classes. In spite of being the least informative in the measurement scale hierarchy, Nominal Scale may provide important information. The key requirements for the categories in nominal scale are jointly exhaustive and mutually exhaustive[6].

Ordinal Scale

A scale is an ordinal one if it divides the set of entities into categories that are ordered according to some order. For instance, we may classify students according to their grades: grade A, grade B and grade C. The ordinal measurement is more informative than nominal Scale in measurement hierarchy. It not only groups subjects into categories, but also orders the categories according to some particular order. The arrangement of acceptable changes is the set of strictly monotonic functions i.e., those functions that is consistently increasing and never decreasing or consistently decreasing and never increasing. Two important properties of ordinal scale are

- Asymmetric (if A > B is true, then B > A is false), and
- Transitive (if A>B and B>C then A>C)[6].

Interval Scale

In an interval scale, the exact difference between two values is the basis for meaningful statements. For instance, weight of 50, 60 and 70 kilograms differ from each other by 10 kilograms. The difference (the interval) between 10 and 20, 40 and 50, and 70 and 80 is the same, but we cannot say that an 80 kilograms entity is ten times heavier than 8 kilogram entity. The arrangement of acceptable changes is positive linear (changes linearly). Interval scale works regardless of the origin of units we adopt and the unit used i.e.; we can always change the origin of the measure and the unit of measure. Arithmetic addition and subtraction are supported on interval scale data.

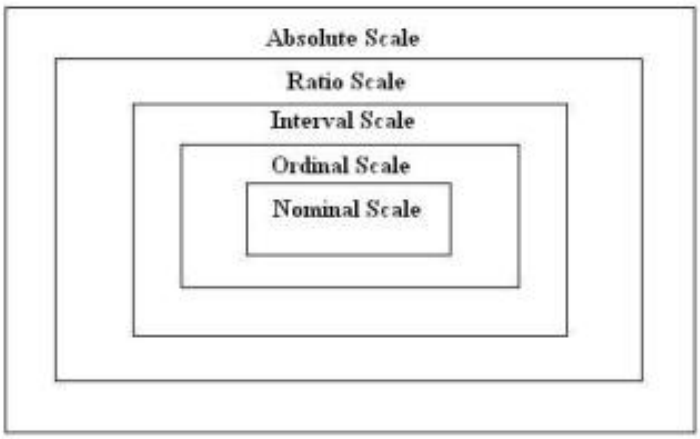
Ratio Scale

In a ratio scale, contrary to interval scale, there is an absolute and non arbitrary zero point[6]. A ratio scale allows the definition of meaningful statements of the kind *"twice as much"* on the single values of the measure. For instance, I am twice as tall as you. The arrangement of acceptable changes is proportional .We have to necessarily choose the reference origin of reference non arbitrarily which is fixed and its value is 0 (zero). We may change the unit in this scale too without changing the meaning of statements. For instance, when measuring the height of entities, we can use foots, inches, centimeters, etc. Apart from addition and subtraction, multiplication and division are also supported on ratio scale data.

Absolute Scale

Absolute scale is the most informative in the measurement scale hierarchy. In absolute scale, measurement is done by counting method. For instance, you have four testing books; I have only one .There is only one acceptable change, i.e., the identity change. The unit of measurement is fixed. This scale is seldom used in practice.

Measurement scales are hierarchal, and each level scale possesses all the properties of lower scales as shown below. Powerful analysis can be applied to data with more informative scale of measurement.



Measurement Scale Hierarchy

Testing Techniques on a measurement Scale

Now the question remains which technique among existing techniques is effective with respect to one another? To do so we need to place these testing techniques onto a scale of measurement aforementioned so that we can opt for those testing technique which are more effective to produce reliable and quality products. Although a large number of software testing techniques have been proposed, we are basically ignorant of their respective powers. There is still not adequate proof to indicate which of these techniques are effective. It is difficult to perform meaningful comparisons of the effectiveness of testing techniques. We need to know how much effort is involved in carrying out each testing technique, how many faults they are likely to catch, of what kind and so on. Indeed it can be said that software testing should do this if only to be categorized within software engineering properly. One way to solve this problem is to place software testing techniques on a measurement scale.

For instance, if we have to choose the more effective testing technique between A and B. Occasionally we can create an incomplete ordering of testing techniques, stating that A will achieve a high level of coverage than B [7]. It should be noted that, such incomplete ordering are unable to say anything how much more coverage is accomplished

by one testing technique over another, and very often testing techniques are placed at the same level because we are not able to distinguish between their relative coverage abilities. Sometimes we are even not able to place A and B in incomplete ordering that we cannot say anything about the relative coverage merits of the techniques. Even if one will argue that his work has revealed that a particular testing technique is more effective, that information cannot be appropriately established as result differs for various parameters (software type, complexity) for the same testing techniques. Many people like (Basili and Selby[8], Lauterbach and Randell[9], Howden [10], Girgis and Woodward[11] and Lesnaik and Betly[12]) have indeed worked a lot in this direction but their works unfortunately have a lot of contradiction in terms of their results and the types of parameters they have used. Even if there were no contradictions, still we can not place testing techniques on a measurement scale which can tell us about the relative effectiveness of testing techniques (at least Ordinal) because of the difference between parameters they have taken into consideration and the results also, do not reveal much information.

At times incomplete orderings can be useful, however, they only represent theoretical comparisons of the level of thoroughness with which a testing technique may test a program but they do not perfectly describe the ability of testing technique to find faults within a program and, therefore, are of little use to the software testing team. Sometimes even that incomplete ordering creates chaos because sometimes a stronger technique in that incomplete ordering will fail to locate a fault revealed by a much weaker technique.

For Instance: Consider the below example

```
if (condition1 && (condition2 || fun()))
        statement1;
        else
        statement2;
```

Code Coverage could consider that the above control structure is completely exercised without a call to fun. The test expression is true when *condition1* is true and *condition2* is true, and the test expression is false when *condition1* is false. In this case, the short-circuit operators preclude a call to fun. These type of errors can be noticed directly by a software developer or it can be found during a code reading (a much weaker testing technique), but the decision coverage report shows 100 percent coverage. If a manager sees 100 percent coverage, he or she may get a false sense of security, decide that testing is complete, and release the buggy code into production. Admittedly above quoted example is trivial one and these kinds of things increase with increasing size of program.

Most established measures look at the simplest building blocks of the program (lines of code) and the flow of control from one line to the next. Many bugs will not be detected even if there is complete line coverage, complete branch coverage, or even if there were complete path coverage. Using convenient measures and then ignoring everything else which is more subtle or difficult to measure can leave many bugs in the program undetected. [13] in his work shows the importance of using different testing techniques to dig out different kinds of errors. It is just not sufficient to rely on a single method for catching all errors or problems in a program.

There is no general scale of software testing techniques. Everybody classifies or places testing techniques in some order of effectiveness according to the empirical results they obtain on some software. Since we are not able to say anything about the relative merits of the available testing techniques (other than incomplete orderings); software testing techniques can only be placed on a nominal scale.

Conclusion and Future work

Presently we are ignorant about the relative ordering of software testing techniques and if we are to make software testing more effective then we need to place existing software testing techniques at least on an ordinal scale. To do so we need to carry out experimentation on large scale but that needs to in a way that can be compared so that we can put testing techniques on a scale that will have no contradictions. For that we also need to establish common and standard parameters so that there are little variations in experimentation goals.

REFERENCES

1. Myers, Glenford J., The art of software testing, Publication info: New York: Wiley, c1979. ISBN: 0471043281 Physical description: xi, 177 p.: ill. ; 24 cm.
2. Hetzel, William C. The Complete Guide to Software Testing, 2nd ed. Publication info: Wellesley, Mass.: QED Information Sciences, 1988. ISBN: 0894352423.Physical description: ix, 280 p.: ill; 24 cm.
3. J.B. Good Enough and S. L. Gerhart, "Toward a Theory of Test Data Selection," IEEE Transactions on Software Engineering, June 1975, pp. 156-173
4. Graham, Dorothy R.(1996b)."Measuring the effectiveness & efficiency of testing."In proceedings of software testing" 96 (Escape Champerret, Paris, France) (June)
5. Marnei L. Hutcheson, "Software Testing Fundamentals: Methods and Metrics" WILEY Pub.
6. Stephen H. Kan, "Metrics and Models In Software Quality Engineering", PEARSON
7. Simein C. Ntafos. A Comparision of some structural testing techniques. IEEE Transactions on software Engineering, 14(6):868-874, June 1998
8. Victor R. Basili and Richard W. Selby comparing the effectiveness of the software testing strategies,IEEE transactions on software engineering. SE-13(12);1278-1296, December 1987.
9. L. Lauterbach and W. Randell experimental evaluation of six test techniques. In proc. Compass 89, page 36-41, ACM Press, 1987.
10. William E. Howden. An evaluator of effectiveness of symbotic testing.Software-practice & experience, 8:381-397, 1982.
11. M.R. Girgis & M.R. Woodward. An experimental comparision of the error exposing ability of program testing criteria. In proc. Workshop on software testing. 64-73.IEEE, July 1986.
12. A.M Lesniak-Betley. Audit of statement analysis methodology. In proc. 3rd Annual International Phoenix conference on computers & communications, pages 174-180, march 1984.
13. Olsen, Neil(1993) "The software rush hour" IEEE software, 10(50 (September): 29-37.