20

An evaluation scheme of software testing techniques

H. D. Chu
Centre for Software Reliability, Department of Computing Science
University of Newcastle upon Tyne, NE3 2AP, U.K.
Tel:+44–191–222 8972 Fax:+44–191–222 8887
Email:huey–der.chu@newcastle.ac.uk

Abstract

In addressing the two major software testing issues, that is when to stop testing and how good is the technique after testing, this paper presents a scheme by a data flow diagram (DFD) for evaluating software testing techniques based on the works of classification. Following this diagram step by step, all the activities involved and the relative techniques were described. A strategy proposal for software testing in the development of applications is advocated later.

Keywords

Software testing, deterministic testing, statistical testing, test data adequacy

1 INTRODUCTION

The history of software testing is as long as the history of software development itself. It is an integral part of the software life–cycle and must be structured according to the type of product, environment and language used. In the absence of feasible and cost–effective theoretical methods for verifying the correctness of software designs and implementations, software testing plays a vital role in validating both. The goal of software testing is (Myers, 1978; Bertolino, 1991): Firstly, to reveal that hidden number of defects which are created during the specification, design and coding stages of development, secondly, to provide confidence that failures do not occur and thirdly, to reduce the cost of software failure over the life of a product.

In practice, the software development methodologies typically employ a combination of several software testing techniques. There is no "silver bullet" testing approach and no single technique alone is satisfactory. The need to combine testing techniques is further visible when we consider the primary characteristics of each approach and find that each testing strategy addresses only a narrow set of concerns.

From this viewpoint, a framework is presented in this paper to the classification of software testing techniques, to the evaluation of software testing techniques and to the proposal of testing strategy. An evaluation scheme of software testing techniques is presented in section 2. The proposal of software testing strategy will be discussed in section 3. Concluding remarks are made in Section 4.

D. Gritzalis (ed.), *Reliability, Quality and Safety of Software-Intensive Systems*

2 AN EVALUATION SCHEME OF SOFTWARE TESTING TECHNIQUES

The purpose of this evaluation scheme is to allow us to identify the strengths and weakness of current software testing techniques. This will provide the information for selecting the testing strategy in the development of applications. In addressing the two major testing issues, that is when to stop testing and how good is the technique (or the software) after testing, a Data Flow Diagram (DFD) depicting the evaluation scheme is shown in Figure 1; the circles in the diagram correspond to the tasks that will be identified in the following sub–sections.

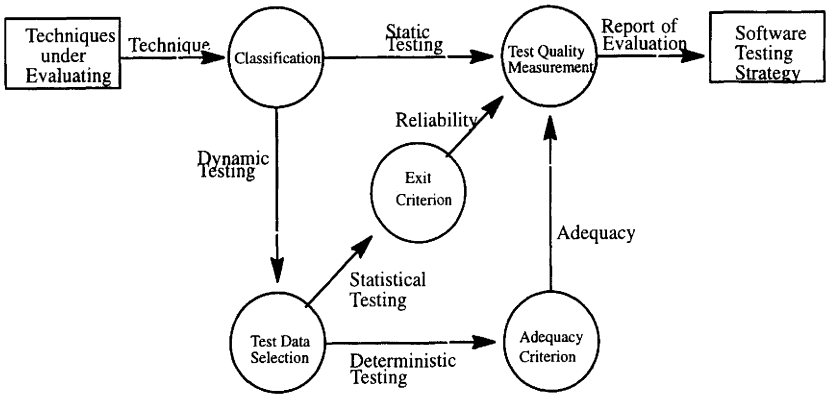

Figure 1 An evaluation scheme of software testing techniques

The classification of software testing techniques

The software testing techniques can be classified according to the following viewpoints:

- Does the technique require us to execute the software? If so, the technique is dynamic testing; if not, the technique is static testing.
- Does the technique require examining the source code in dynamic testing? If so, the technique is white–box testing; if not, the technique is black–box testing.
- Does the technique require examining the syntax of the source in static testing? If so, the technique is syntactic testing; if not, the technique is semantic testing.
- How does the technique select the test data? Test data is selected depending on whether the technique refers to the function or the structure of the software, leading respectively to functional testing and structural testing, where as test data is selected according to the way in which software is operated with respect to random testing.
- What type of test data does the technique generate? In deterministic testing, test data are predetermined by a selective choice according to the adopted criteria. In random testing, test data are generated according to a defined probability distributed on the input domain.

The classification of software testing techniques is shown in Figure 2.

The evaluation of software testing techniques

With reference to this classification, the work on the evaluation of software testing techniques can be done in correspondence with the two major testing issues as shown:

- When should testing stop? The exit criterion can be based on a reliability measure when the test data have been selected by random testing, whereas a test data adequacy criterion for determining whether or not a test set is sufficient for deterministic testing.

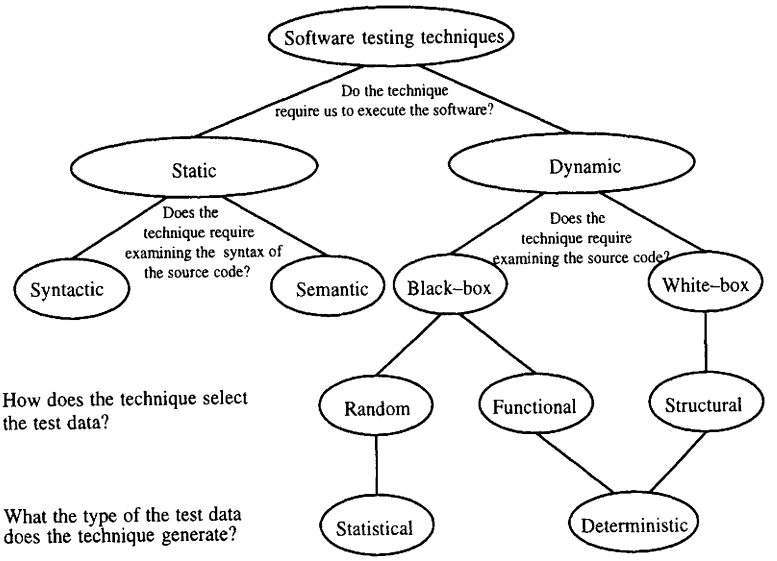

Figure 2 The classification of software testing techniques

- How good is the technique after testing? The definition of software reliability measure with failure rate can be applicable to test software with discrete or continuous test data (DeMillo, McCracken, Martin & Passafiume, 1987). Test data adequacy criteria are measures of the quality of testing. From this viewpoint, the classification of test adequate criteria can be divided into fault–based testing and error–based testing (Zhu, Hall & May, 1994). However, as this is the most important aspect of test quality, there are many experimental works to measure it by metrics, mutation analysis and the expected number of failures detected *et al.*

3 SOFTWARE TESTING STRATEGY

Some researchers have suggested that static testing techniques should completely replace dynamic testing techniques in the verification and validation process and that dynamic testing is unnecessary (Sommerville, 1996). However, static testing can only check the correspondence between a program and its specification but it cannot demonstrate that the software is operationally useful. Therefore, although static testing techniques are becoming more widely used, dynamic testing is necessary for reliability assessment, performance analysis, user interface validation and to check that the software requirements are what the user really wants. The dynamic testing strategy advocated here combines deterministic and random testing. The way to mix the two testing techniques is deduced from their complementary features, that is, to use the deterministic testing techniques first for removing the more easily discovered faults and to use the random testing techniques later for assessing the reliability of the resulting software. The strategy proposal for software testing is shown in Figure 3.

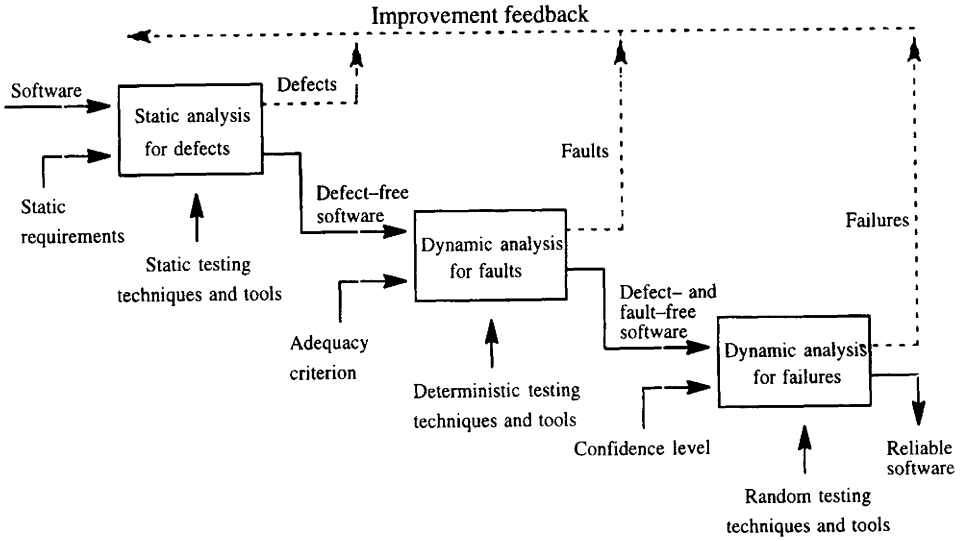

Figure 3 The strategy proposal for software testing

4 CONCLUSION

Software testing is characterized by the existence of many methods, techniques and tools, that must fit the test situation, including technical properties, goals and restrictions. There is no single ideal software testing techniques for assessing software quality. Therefore, we must ensure that the testing strategy was chosen by the combination of testing techniques at the right time on the right work products. From this viewpoint, a scheme for evaluating software testing techniques is presented to the classification and evaluation of software testing techniques. A strategy proposal for software testing also is advocated in this paper. We expect that the proposal will provide a guide-line to testers in the development of applications.

5 REFERENCES

Bertolino, A. (1991). An Overview of Automated Software Testing. *Journal Systems Software*, **15**, 133 – 138.

DeMillo, R. A., McCracken, W. M., Martin, R. J. and Passafiume, J. F. (1987) *Software Testing and Evaluation*, The Benjamin/Cummings Publishing Company, Inc., Workingham.

Myers, G.J. (1978). *The Art of Software Testing*. John Wiley & Sons, New York.

Sommerville, I. (1996). *Software Engineering* (Fifth ed.). Addison–Wesley Pub., Wokingham.

Zhu, H., Hall, P. & May, J. (1994). *Software Test Coverage and Adequacy* (Technical Report 94/15). The Open University.

6 BIOGRAPHY

H.D. Chu is currently a PhD student in computing science at the University of Newcastle upon Tyne funded by the National Science Council in Taiwan. He is also a lecturer in the Information Management Department at the National Defense Management College in Taiwan. His research interests include methodical and statistical techniques for automating testing.