Available Online at www.ijcsmc.com

International Journal of Computer Science and Mobile Computing

A Monthly Journal of Computer Science and Information Technology

ISSN 2320–088X

IJCSMC, Vol. 3, Issue. 6, June 2014, pg.34 – 37

RESEARCH ARTICLE

Software Engineering and Testing Methods

Miss. Harshali S. Sawale, Prof. Mr. R.R.Keole

M.E (CS-IT) SGBAU, INDIA
M.E (CS-IT) SGBAU, INDIA
Harshalisawale8@gmail.com; ranjitkeole@gmail.com

Abstract— The purpose of testing can be quality assurance, verification and validation, or reliability estimation. Testing can be used as a generic metric as well. Correctness testing and reliability testing are two major areas of testing. Software testing is a trade-off between budget, time and quality. Software testing, depending on the testing method employed, can be implemented at any time in the software development process. Traditionally most of the test effort occurs after the requirements have been defined and the coding process has been completed, but in the Agile approaches most of the test effort is on-going. As such, the methodology of the test is governed by the chosen software development methodology, Software testing is any activity aimed at evaluating an attribute or capability of a program or system and determining that it meets its required results.

Keywords— Testing, Engineering, Software, Attribute

I. INTRODUCTION

The scope of software testing often includes examination of code as well as execution of that code in various environments and conditions as well as examining the aspects of code: does it do what it is supposed to do and do what it needs to do. In the current culture of software development, a testing organization may be separate from the development team. Testing can never completely identify all the defects within software.[1] Instead, it furnishes a criticism or comparison that compares the state and behavior of the product against oracles—principles or mechanisms by which someone might recognize a problem. Testing can never completely identify all the defects within software. Instead, it furnishes a criticism or comparison that compares the state and behavior of the product against oracles—principles or mechanisms by which someone might recognize a problem. A primary purpose of testing is to detect software failures so that defects may be discovered and corrected. Testing cannot establish that a product functions properly under all conditions but can only establish that it does not function properly under specific conditions

II. Testing Methods

Static vs. dynamic testing

There are many approaches to software testing. Reviews, walkthroughs, or inspections are referred to as static testing, whereas actually executing programmed code with a given set of test cases is referred to as dynamic testing. Static testing is often implicit, as proofreading, plus when programming tools/text editors check source code structure or compilers (pre-compilers) check syntax and data flow as static program analysis. Dynamic testing takes place when the program itself is run. Dynamic testing may begin before the program is 100% complete in order to test particular sections of code and are applied to discrete functions or modules. Typical techniques for this are either using stubs/drivers or execution from a debugger environment. Static testing involves verification, whereas dynamic testing involves validation.

Together they help improve software quality. Among the techniques for static analysis, mutation testing can be used to ensure the test-cases will detect errors which are introduced by mutating the source code. Software testing methods are traditionally divided into white- and black-box testing. These two approaches are used to describe the point of view that a test engineer takes when designing test cases.

White-Box testing

White-box testing (also known as clear box testing, glass box testing, transparent box testing and structural testing) tests internal structures or workings of a program, as opposed to the functionality exposed to the end-user. In white-box testing an internal perspective of the system, as well as programming skills, are used to design test cases. The tester chooses inputs to exercise paths through the code and determine the appropriate outputs. This is analogous to testing nodes in a circuit, e.g. in-circuit testing (ICT).

While white-box testing can be applied at the unit, integration and system levels of the software testing process, it is usually done at the unit level. It can test paths within a unit, paths between units during integration, and between subsystems during a system–level test. Though this method of test design can uncover many errors or problems, it might not detect unimplemented parts of the specification or missing requirements.

Black box testing

The black-box approach is a testing method in which test data are derived from the specified functional requirements without regard to the final program structure. It is also termed data-driven, input/output driven or requirements-based testing. Because only the functionality of the software module is of concern, black-box testing also mainly refers to functional testing -- a testing method emphasized on executing the functions and examination of their input and output data. The tester treats the software under test as a black box -- only the inputs, outputs and specification are visible, and the functionality is determined by observing the outputs to corresponding inputs. In testing, various inputs are exercised and the outputs are compared against specification to validate the correctness. All test cases are derived from the specification. No implementation details of the code are considered.

Security testing

Software quality, reliability and security are tightly coupled. Flaws in software can be exploited by intruders to open security holes. With the development of the Internet, software security problems are becoming even more severe.

Many critical software applications and services have integrated security measures against malicious attacks. The purpose of security testing of these systems include identifying and removing software flaws that may potentially lead to security violations, and validating the effectiveness of security measures. Simulated security attacks can be performed to find vulnerabilities in testing. The degree of automation remains at the automated test script level. The problem is lessened in reliabilities.

Performance testing

Not all software systems have specifications on performance explicitly. But every system will have implicit performance requirements. The software should not take infinite time or infinite resource to execute. "Performance bugs" sometimes are used to refer to those design problems in software that cause the system performance to degrade. Performance has always been a great concern and a driving force of computer evolution. Performance evaluation of a software system usually includes: resource usage, throughput, stimulus-response time and queue lengths detailing the average or maximum number of tasks waiting to be serviced by selected resources. Typical resources that need to be considered include network bandwidth requirements, CPU cycles, disk space, disk access operations, and memory usage. The goal of performance testing can be performance bottleneck identification, performance comparison and evaluation, etc. The typical method of doing performance testing is using a benchmark -- a program, workload or trace designed to be representative of the typical system

Testing automation

Software testing can be very costly. Automation is a good way to cut down time and cost. Software testing tools and techniques usually suffer from a lack of generic applicability and scalability. The reason is straight-forward. In order to automate the process, we have to have some ways to generate oracles from the specification, and generate test cases to test the target software against the oracles to decide their correctness. Today we still don't have a full-scale system that has achieved this goal. In general, significant amount of human intervention is still need testing and performance testing.

In robustness testing, the simple specification and oracle: doesn't crash, doesn't hang suffices. Similar simple metrics can also be used in stress testing.

Available tools, techniques, and metrics

There are an abundance of software testing tools exist. The correctness testing tools are often specialized to certain systems and have limited ability and generality. Robustness and stress testing tools are more likely to be made generic. Mothora is an automated mutation testing tool-set developed at Purdue University. Using Mothora, the tester can create and execute test cases, measure test case adequacy, determine input-output correctness, locate and remove faults or bugs, and control and document the test. They are run-time checking and debugging aids. They can both check and protect against memory leaks and pointer problems. Ballista COTS Software Robustness Testing Harness. The Ballista testing harness is a full-scale automated robustness testing tool. The first version supports testing up to 233 POSIX function calls in UNIX operating systems. The second version also supports testing of user functions provided that the data types are recognized by the testing server. The Ballista testing harness gives quantitative measures of robustness comparisons across operating systems. The goal is to automatically test and harden Commercial Off-The-Shelf (COTS) software against robustness failures. Software testing is an art. Most of the testing methods and practices are not very different from 20 years ago. It is nowhere near maturity, although there are many tools and techniques available to use. Good testing also requires a tester's creativity, experience and intuition, together with proper techniques. Testing is more than just debugging. Testing is not only used to locate defects and correct them. It is also used in validation, verification process, and reliability measurement. Testing is expensive. Automation is a good way to cut down cost and time. Testing efficiency and effectiveness is the criteria for coverage-based testing techniques. Complete testing is infeasible. Complexity is the root of the problem. At some point, software testing has to be stopped and product has to be shipped. The stopping time can be decided by the trade-off of time and budget. Or if the reliability estimate of the software product meets requirement. Testing may not be the most effective method to improve software quality. Alternative methods, such as inspection, and clean-room engineering, may be even better.

For Verification & Validation (V&V)

Just as topic Verification and Validation indicated, another important purpose of testing is verification and validation (V&V). Testing can serve as metrics. It is heavily used as a tool in the V&V process. Testers can make claims based on interpretations of the testing results, which either the product works under certain situations, or it does not work. We can also compare the quality among different products under the same specification, based on results from the same test.

We cannot test quality directly, but we can test related factors to make quality visible. Quality has three sets of factors -- functionality, engineering, and adaptability. These three sets of factors can be thought of as dimensions in the software quality space. Each dimension may be broken down into its component factors and considerations at successively lower levels of detail. Table 1 illustrates some of the most frequently cited quality considerations.

Functionality (exterior quality)	Engineering (interior quality)	Adaptability (future quality)
Correctness	Efficiency	Flexibility
Reliability	Testability	Reusability
Usability	Documentation	Maintainability
Integrity	Structure	

Table 1. Typical Software Quality Factor

Conclusion

According to my aims and objectives of this paper I have done the detailed research and get the knowledge about it and the factors that need to be consider carefully. Software testing is an art. Most of the testing methods and practices are not very different from 20 years ago. It is nowhere near maturity, although there are many tools and techniques available to use. Good testing also requires a tester's creativity, experience and intuition, together with proper techniques. Testing is more than just debugging. Testing is not only used to locate defects and correct them. It is also used in validation, verification process, and reliability measurement. Testing is expensive. Automation is a good way to cut down cost and time. Testing efficiency and effectiveness is the criteria for coverage-based testing techniques. Complete testing is infeasible. Complexity is the root of the problem. At some point, software testing has to be stopped and product has to be shipped. The stopping time can be decided by the trade-off of time and budget. Or if the reliability estimate of the

software product meets requirement. Testing may not be the most effective method to improve software quality. Alternative methods, such as inspection, and clean-room engineering, may be even better.

ACKNOWLEDGMENT

I express my sincere thanks to **Prof P.L.Ramteke, HOD of CS-IT dept of HVPM, COET, Amravati** for allowing me to present this paper also I express my heartiest thanks to **Prof. R.R.keole Assist. Prof. of HVPM, COET, Amravati** for his valuable guidance while preparing this paper and guiding me time to time. Also all the friends and Staff who help me in preparing this paper.

REFERENCES

Ballista99] http://www.cs.cmu.edu/afs/cs/project/edrc-ballista/www/

Ballista COTS Software Robustness Testing Harness homepage.

[Basili85] Victor R. Basili, Richard W. Selby, Jr. "Comparing the Effectiveness of Software Testing Strategies",

Technical Report, Department of Computer Science, University of Maryland, College Park, 1985.

[Beizer90] Boris Beizer, Software Testing Techniques. Second edition. 1990

Kolawa, Adam; Huizinga, Dorota (2007)

