Volume 2, No. 6, June 2011
Journal of Global Research in Computer Science
ISSN-2229-371X
RESEARCH PAPER
Available Online at www.jgrcs.info

IMPROVING QUALITY USING TESTING STRATEGIES

Sahil Batra[*1], Dr. Rahul Rishi[2]

* [1]Department of Computer Science and Engineering
The Technological Institute of Textile and Science, Bhiwani-127021, Haryana – India
sahil.batra23@gmail.com
[2]Department of Computer Science and Engineering
The Technological Institute of Textile and Science, Bhiwani-127021, Haryana – India
rahulrishi@rediffmail.com

Abstract: Software testing is a technique aimed at evaluating an attribute or capability/usability of a program or product/system and determining that it meets its quality. Although crucial to software quality and widely deployed by programmer & testers, software testing still remains an art, due to limited understanding of the principles of software. Software testing is an important technique for assessing the quality of a software product. In this paper, various types of software testing technique and various attributes of software quality are explained. Identifying the types of testing that can be applied for checking a particular quality attribute is the aim of this thesis report. All types of testing can not be applied in all phases of software development life cycle. Which testing types are applicable in which phases of life cycle of software development is also summarized.

Keywords: SDLC, Testing, Quality, Efficiency, Software.

INTRODUCTION

Software testing is both a discipline and a process. It is a separate discipline from software development. Software development is the process of coding functionality to meet defined end-user needs. While Software testing tends to be considered a part of development, it is really its own discipline and should be tracked as its own project. Software testing, while working very closely with development, should be independent enough to be able to hold-up or slow product delivery if quality objectives are not met. The objective of software testing is to find problems and fix them to improve quality. Software testing typically represents 40% of a software development budget. There are four main objectives of testing:

Demonstration:

It show that the system can be used with acceptable risk, demonstrate functions under special conditions and show that products are ready for integration or use.

Detection:

It discovers defects, errors, and deficiencies. Determine system capabilities and limitations quality of components, work products, and the system.[1,4]

Prevention:

It provides information to prevent or reduce the number of errors clarify system specifications and performance. Identify ways to avoid risks and problems in the future.

Improving quality:

By doing effective testing, we can minimize errors and hence improve the quality of software.

NEED FOR TESTING

Well, while making food, it's ok to have something extra, people might understand and eat the things we made and may well appreciate our work. But this isn't the case with software project development.[2,8,6] If we fail to deliver a reliable, good and problem free software solution, we fail in our project and probably we may loose our client. So in order to make it sure, that we provide our client a proper software solution, we go for testing. We check out if there is any problem, any error in the system, which can make software unusable by the client. We make software testers test the system and help in finding out the bugs in the system to fix them on time.

DIFFERENCE BETWEEN TESTING AND DEBUGGING

The purpose of debugging is to locate and fix the offending code responsible for a symptom violating a known specification. Debugging typically happens during three activities in software development, and the level of granularity of the analysis required for locating the defect differs in these three. The first is during the coding process, when the programmer translates the design into an executable code. [8,11].During this process the errors made by the programmer in writing the code can lead to defects that need to be quickly detected and fixed before the code goes to the next stages of development. Most often, the developer also performs unit testing to expose any defects at the module or component level. The second place for debugging is during the later stages of testing, involving multiple components or a complete system, when unexpected behavior such as wrong return codes or abnormal program termination may be found. A certain amount of debugging of the test execution is necessary to conclude that the program under test is the cause of the unexpected behavior.

SOFTWARE QUALITY

Everyone is committed to quality; however, the following statement shows some of the confusing ideas shared by many individuals that inhibit achieving a quality commitment: Quality requires a commitment, particularly from top management. Close cooperation of management and staff is required in order to make it happen.

a. Many individuals believe that defect-free products and services are impossible, and accept certain levels of defects as normal and acceptable.
b. Quality is frequently associated with cost, meaning that high quality equals high cost. This is confusion between quality of design and quality of conformance.
c. Quality demands requirement specifications in enough detail that the software produced can be quantitatively considered alongside those requirements. Many organizations are not capable or willing to expend the effort to produce specifications at the level of detail required.
d. Technical personnel often believe that standards stifle their creativity, and thus do not abide by standards compliance. However, for quality to happen, well-defined standards and procedures must be followed.[4,8,7]

Quality cannot be achieved by assessing an already completed product. The aim therefore, is to prevent quality defects or deficiencies in the first place, and to make the products assessable by quality assurance measures. Some quality assurance measures include: structuring the development process with a software development standard and supporting the development process with methods, techniques, and tools.

The undetected bugs in the software that caused millions of losses to business have necessitated the growth of independent testing, which is performed by a company other than the developers of the system [8]. In addition to product assessments, process assessments are essential to a quality management program. Examples include documentation of coding standards, prescription and use of standards, methods, and tools, procedures for data backup, test methodology, change management, defect documentation, and reconciliation. Quality management decreases production costs because the sooner a defect is located and corrected, the less costly it will be in the long run [7]. With the advent of automated testing tools, although the initial investment can be substantial, the long-term result will be higher-quality products and reduced maintenance costs. The total cost of effective quality management is the sum of four component costs: prevention, inspection, internal failure, and external failure. Prevention costs consist of actions taken to prevent defects from occurring in the first place. Inspection costs consist of measuring, evaluating, and auditing products or services for conformance to standards and specifications [9]. Internal failure costs are those incurred in fixing defective products before they are delivered.

Quality Attributes

Quality can be measured using various quality attributes. Common ones are discussed here:

Understandability:

The purpose of the software product is clear. This goes further than just a statement of purpose all of the design and user documentation must be clearly written so that it is easily understandable. Obviously, the user context must be taken into account, e.g. if the software product is to be used by software engineers it is not required to be understandable to lay users.[9,10]

Completeness:

All parts of the software product are present and each of its parts are fully developed. For example, if the code calls a sub-routine from an external library, the software package must provide reference to that library and all required parameters must be passed. All required input data must be available.

Conciseness:

No excessive information is present. This is important where memory capacity is limited, and it is important to reduce lines of code to a minimum. It can be improved by replacing repeated functionality by one subroutine or function which achieves that functionality. This quality factor also applies to documentation.

Portability:

The software product can be easily operated or made to operate on multiple computer configurations. This can be between multiple hardware configurations (such as server hardware and individual computers), multiple operating systems [7, 8] (e.g. Microsoft Windows and Linux-based operating systems), or both.

Consistency:

The software contains uniform notation, symbology and terminology within itself.

Maintainability:

The product should facilitate updating to satisfy new requirements and software product that is maintainable is simple, well documented.

Testability:

The software product facilitates the establishment of acceptance criteria and supports evaluation of its performance. Such a characteristic must be built-in during the design phase if the product is to be easily testable, since a complex design leads to poor testability.

Usability:

The product is convenient and practicable to use. The component of the software which has most impact on this is the user interface (UI), which for best usability is usually graphical.[4,5]

Reliability:

The software can be expected to perform its intended functions satisfactorily over a period of time. Reliability also encompasses environmental considerations in that the product is required to perform correctly in whatever conditions it is operated in; this is sometimes termed robustness.

Structure:

The software possesses a definite pattern of organization in its constituent parts.

Efficiency:

The software product fulfils its purpose without wasting resources, e.g. memory or CPU cycles.[3]

Security:

The product is able to protect data against unauthorized access and to withstand malicious interference with its operations. Besides the presence of appropriate security mechanisms such as authentication, access control and encryption, security also implies reliability in the face of malicious, intelligent and adaptive attackers .

In order to measure quality, we need to analyse requirements to design test cases, then design the test cases, document them, implement them and execute these test cases. Then the results are analysed. Before all this, we need to plan for testing, including risk analysis and test management practices. An example is IBM RUP software tools used by testers to execute a software test plan . This all includes communication skill for the effective tester.

PROBLEM STATEMENT

The main purpose of software testing is to uncover errors which are not simply syntax errors in code but various other types of errors in all the documents produced during the software development e.g. software requirements document, design document, test plan etc.[8,6,5] Various types of software testing techniques have been developed till date, but which type of testing technique will be suitable and sufficient for checking a particular document in which phase of software development life cycle is not yet clear. So here the problem is to

a. Identify the testing techniques which can be applied at different levels and phases of software development life cycles.
b. Identify the testing techniques which can be applied to measure which software quality attribute.

Proposed Scheme

Application of Testing to Measurement of Quality Attributes

Different quality attributes need different types of testing to measure software quality. Various types of testing according to the quality feature it applies to in the table 1.[4,6,7] In given table we identified that for a particular software quality feature which type of software testing technique can be applied:

Table 1: Testing Strategies for different Quality aspects

Recovery	Recovery testing
Completeness	Boundary/Statement/Loop/Condition/ Path coverage testing
Efficiency	Performance testing
Understandability	Usability Testing
Structure	Structural testing
Maintainability	Regression testing

Quality Attribute	Types of Testing
Functionality	Functional testing
Security	Security testing
Complexity	Unit testing
Performance	Performance testing
Compatibility	Compatibility testing
Reliability	Stress testing, Robustness testing, load testing
Vulnerability	Penetration testing
Usability	Comparison testing
Consistency	Database testing, Table testing
Correctness	Database testing, Table testing
Portability	Portability Testing

Experimental Scenario:

As the figure given above we mention the different type of Quality aspects and for that which type of Software testing is needed. Here we are taking the Efficiency aspect for that we have to perform Performance testing. For the Performance testing we are performing a scenario with the help of a WAPT Software.

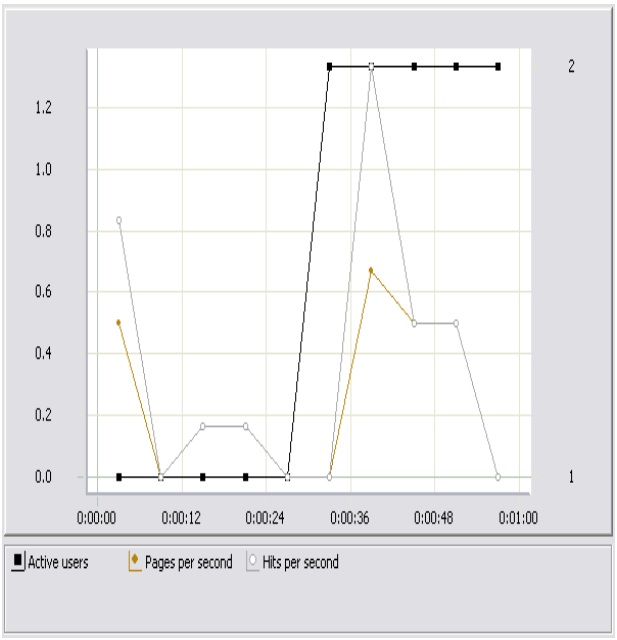

Figure 1 overall performance

Successful sessions per second

Profile	0:00:00-0:00:06	0:00:06-0:00:12	0:00:12-0:00:18	0:00:18-0:00:24	0:00:24-0:00:30	0:00:30-0:00:36	0:00:36-0:00:42	0:00:42-0:00:48	0:00:48-0:00:54	0:00:54-0:01:00	Total
Profile1	0	0	0	0	0	0	0	0	0	0	0
Total	0	0	0	0	0	0	0	0	0	0	0

Successful pages per second

Profile	0:00:00-0:00:06	0:00:06-0:00:12	0:00:12-0:00:18	0:00:18-0:00:24	0:00:24-0:00:30	0:00:30-0:00:36	0:00:36-0:00:42	0:00:42-0:00:48	0:00:48-0:00:54	0:00:54-0:01:00	Total
Profile1	0.50	0	0.17	0.17	0	0	0.67	0.50	0.50	0	0.25
Total	0.50	0	0.17	0.17	0	0	0.67	0.50	0.50	0	0.25

Successful hits per second

Profile	0:00:00-0:00:06	0:00:06-0:00:12	0:00:12-0:00:18	0:00:18-0:00:24	0:00:24-0:00:30	0:00:30-0:00:36	0:00:36-0:00:42	0:00:42-0:00:48	0:00:48-0:00:54	0:00:54-0:01:00	Total
Profile1	0.83	0	0.17	0.17	0	0	1.33	0.50	0.50	0	0.35
Total	0.83	0	0.17	0.17	0	0	1.33	0.50	0.50	0	0.35

CONCLUSION

Quality is the main focus of any software engineering project. Without measuring, we cannot be sure of the level of quality in software. So the methods of measuring the quality are software testing techniques. This thesis report relates various types of testing technique that we can apply in measuring various quality attributes. Also which testing are related to various phase of SDLC. General SDLC processes are applied to different type of projects under different conditions and requirements. There are various type of SDLC model. But in all these models, testing is applied after a particular stage and not in all the phases. In this thesis report, it is concluded that testing should be applied in all the phases of SDLC and not at a particular stage. Which type of testing technique can be applied to which type of SDLC phase is also summarized?

REFERENCES

[1] Jain Deepak, "Software Engineering Principle and Practices" First edition by Oxford University Press, ISBN-13: 978-0-19-569484-0, (2009).

[2] Bersoff, E.H. and A.M. Davis, "Impact of Lifecycle Modes on Software Configuration Management", ACM, pp104-108 (1991).

[3] Boris Beizer, "Software testing techniques", Second edition, (1990).

[4] Chilarege, Ram, "Software Testing Best Practises" Center for Software Engineering, IBM Research (1999).
[5] Joe W. Duran, Semeon, C. Ntafos, "An Evaluation of Random Testing ", IEEE Transactions on Software engineering,Vol.SE-10,No.4, pp438-443 (July 1984).
[6] Beizer, Boris, "Black-Box Testing Technique for Functional Testing of Software and System" New York Wiley, ISBN: 0471120944 Physical description: xxv, 194 p. ill.; 23cm (1995).
[7] Barber, Scott "Software Testing: An Introduction", PerfTestPlus (2006).
[8] Cem Karner, "Testing Computer Software", (1993).
[9] IEEE "Standard Glossary of Software Engineering Terminology" (IEEE Std 610.12-1990), IEEE Computer society, (dec.10, 1990).
[10] Ballista COTS "Software Robustness Testing Harness" (1999).
[11] Kropp, N P Koopman, P J Siewiorek D P "Automated Robustness Testing of the-Shelf Software Component" 28th Annual International Symposium on Fault-Tolerant Computing (1995).'

