Software Testing and Quality Assurance

Dennis DeVolder
Department of Computer Science
Western Illinois University
Macomb, Illinois 61455
d-devolder@wiu.edu

Shahin Ghazanshahi
Department of Electrical Engineering
California State University
Fullerton, CA 92834
sghazanshahi@fullerton.edu

Jeff Zadeh
Department of Mathematics and Computer Science
Virginia State University
Petersburg, VA 23806
jzadeh@att.net

ABSTRACT

Software testing is the execution of software with actual test data which produce expected results. There are two basic concerns in software testing: what test cases should be used, and how many test cases are necessary? Software quality assurance is the set of activities that ensure software processes and products conform to requirements and standards. In this paper, we will discuss software testing, software quality assurance and related issues. This is very important for all software development, especially for medical related software.

1. INTRODUCTION

Testing is the process of finding differences between the expected behavior, specified by the requirements, and the observed behavior of the system. Testing is often performed by developers who were not involved with the construction of the system. A fault (defect or bug) is the mechanical or algorithmic cause of an error. The goal of testing is to maximize the number of discovered faults, which then allows developers to correct them and increase the reliability of the system.

The importance of medical software must not be underestimated. The success and failure of a medical practice hinges on these medical software systems. These software systems are mission critical and require critical real time systems development. Medical software systems also are required independent verification in order to increase the objectivity and the productivity. The medical software engineers develop and maintain these critical software applications by applying techniques and knowledge from areas of computer science, computer engineering, biomedical engineering and project management.

The complexity of current software applications can be difficult to comprehend for anyone without experience in modern-day software development. Client-server and distributed applications, data communications, and enormous relational databases have all contributed to the exponential growth in software complexity. If there are changes in dependencies among parts of the project, they are likely to interact, cause problems, and may result in errors. Time pressures and scheduling of software projects contribute to these problems. Management must understand the resulting risks, and software tester must adopt a plan for continuous extensive testing to keep the inevitable faults from running out of control.

2. SOFTWARE TESTING

Software testing is a process which measures the quality of developed computer software. Usually, quality is constrained to such topics as correctness, completeness, security, reliability, efficiency, portability, maintainability, compatibility, and usability. Testing can never completely establish the correctness of the computer software. In case of a failure, the software does not do what the user expects.

A common practice of software testing is to provide a software product to an independent group of testers after finishing the product. As important as this method is, it is inadequate if it is the sole method of testing. The better practice is to start regular, incremental software testing at the beginning of the project and continue the process until the project finishes. It is commonly believed that the earlier a defect in a project is found, the less expensive it is to fix.

There are two basic concerns in software testing: what test cases to use and how many test cases are necessary? Test case selection can be viewed as an attempt to space the test cases throughout the input space. Of course, some areas in the domain may be inherently error-prone and may need extra attention.

A program can be seen as a mapping from a domain space to an answer space. Given an input, which is a point in the domain space, the program produces an output, which is a point in the range. Correctness in software is defined as the program mapping being the same as the specification mapping. A test case should always include the expected output.

A test criterion is a rule about how to select a test and when to stop it. One basic issue is how to compare the effectiveness of different test coverage criteria. The standard approach is to use the "subsumes" relationship. To say that one criterion subsumes another means that one criterion is entirely contained within another. For example, if one test criterion required every statement to be executed and another criterion required every statement to be executed plus some additional tests, then the second criterion subsumes the first criterion. Although subsumes is a characteristic that is used for mapping test criterion, it does not measure the relative effectiveness of two criteria. In addition, selecting the minimal set of test cases to satisfy a criterion is not as effective as choosing good test cases.

3. TEST CASES

A test case is a set of inputs and expected results that exercises a program component with the purpose of exposing failures and detecting as many faults as possible. Test case selection can be based on the specifications (functional), the structure of code, the flow of data, or random selection of test cases.

Functional Testing: In functional testing, the specification of the software is used to identify sub - domains that should be tested. One of the first steps is to generate a test case for every distinct type of expected output. Distinctive error messages should be generated and all special cases should have a test case. In addition, any tricky situations, common mistakes and misconceptions should be tested. The result should be a set of test cases that will test the program when it is implemented. The generation of test cases may also help to develop or clarify some of the expected behavior of the project.

In 1979, Myers posed the following functional testing problem: Develop a good set of test cases for a program that accepts three numbers a, b, and c, interprets those numbers as the lengths of the sides of a triangle, and outputs the type of the triangle. He reported that most software developers will not respond with a good test set. Today, developers have similar experiences in developing software products.

Functional testing can be focused on every-statement, every branch, every path, or sub-domain. In "every statement" testing, every statement of the source code should be executed by some test case. Another test criterion is "every branch" which executes both sides of every decision by a test case. In the "every path" testing criterion, a path is a unique sequence of program nodes that are executed by a test case. Since many programs with loops have an infinite number of paths, every path testing is not reasonable.

Data Flow Testing: This is testing based on the flow of data through a program. It is clear that data flows from where it is defined to where it is used. Data definition occurs where a particular value is assigned to a variable. There are two different data definition uses, namely "definition-computation use" and "definition-predicate use." In a definition-computation use, the variable appears on the right-hand side of an assignment statement. In a definition-predicate use, the variable is used in the condition of a decision statement. A definition-predicate use is assigned to both branches of the decision statement.

A definition-free path is a path from a definition of a variable to a use of that variable which does not include another definition of the variable. One of the data flow testing criteria is called definition-computation use, which requires a definition-free path from every definition to a computation use. Another data flow testing is called definition-predicate use, which requires a definition-free path from every definition to a predicate use. One of the most rigorous criteria is all definition use path which requires all definition-free paths from every definition to every possible use.

Random Testing: Random testing is accomplished by randomly selecting the test cases. This approach has the advantage of being faster than other testing methods. Often the tests are selected randomly from the operational profile. By drawing the test cases from the operational profile, the tester will have greater confidence that the behavior of the program during testing is more predictive. If random testing is done in this manner, then the behavior of the software in testing should be the same as its behavior in the operational environment.

4. SOFTWARE QUALITY ASSURANCES

Software Quality Assurance (SQA) is the planned and systematic set of activities that ensures that software conforms to requirements, standards, and procedures. SQA includes the process of assuring that standards and procedures are established and followed throughout the software acquisition life cycle.

Software development is a process of managing and handling many risks. These risks can be both technical and programmatic; risks that the software will not perform as intended or will be too difficult to operate, modify, or maintain are technical risks, while risks that the project will overrun cost or schedule are programmatic risks. The goal of software assurance is to reduce many of these risks.

Establishing standards and procedures for software development is critical. These provide the framework from which the development and control processes are compared. Proper documentation of standards and procedures is necessary since the SQA activities of process monitoring and product evaluation rely upon unequivocal definitions to measure project compliance.

Software Quality Assurance Activities: Product evaluation and process monitoring are the SQA activities that assure the software development is described correctly, and that the project's procedures and standards are followed. It is recommended that the first products monitored by SQA should be those standards and procedures. SQA assures that clear standards exist and evaluates compliance of the software product to the established standards. Another SQA activity is process monitoring, which ensures that appropriate steps to carry out the process are being followed. SQA monitors processes by comparing all of the actual steps carried out with those in the documented procedures.

A fundamental SQA technique is the audit, which looks at a product in depth and evaluates it against established procedures and standards. Audits are used to review management and assurance processes to provide an indication of the quality and status of the software product. The purpose of an SQA audit is to assure that proper control procedures are followed and that the developer's status reports accurately reflect the status of the activity.

Some of the more important relationships of SQA to other management and assurance activities are Configuration Management (CM), Verification and Validation (VV), and Format Testing (FT). SQA assures that CM activities are performed in accordance with development plans, standards, and procedures. The CM activities monitored and audited by SQA include baseline control, configuration identification, configuration control, configuration status accounting, and configuration authentication.

SQA assures Verification and Validation (V&V) activities by monitoring technical reviews, inspections, and walkthroughs. SQA also ensures that all actions are assigned, documented, scheduled, and updated. Technical reviews should be conducted at the end of each phase of the life cycle to identify problems and determine whether the product meets all applicable requirements. In technical reviews, actual work done is compared with established standards to determine whether the product is ready to proceed with the next phase of development. An inspection or walkthrough is a detailed examination of a product on a step-by-step or line-by-line basis to find errors.

SQA assures that formal software testing, such as acceptance testing, is done in accordance with plans and procedures. It includes test plans, test specifications, test procedures, and test reports. By FT, SQA assures software is complete and ready for delivery.

5. CONCLUSIONS

Software development is a process of dealing with many risks. These risks can be both technical and programmatic. The goal of software assurance is to reduce many of these risks, especially for critical systems such as medical or biomedical areas. Although there are many tools and techniques available to use for software testing, the best testing requires a tester's creativity and experience. Testing is not only used to locate defects but also correct them. It is also used in the validation, verification process, and reliability measurement of the software. Testing should be an integral component of the software process and it must be carried out throughout the life cycle.

6. REFERENCES

[1] Beizer, Boris. Software Test Techniques. Second Edition, International Thomson Computer Press, 1990.

[2] Black, Rex. Managing the Testing Process. Second Edition, John Wiley and Sons, 2002.

[3] DeVolder, D. and Zadeh, J. Two-Discipline Development. The 10th World MultiConference on Systemics, Cybernetics and Informatics. July, 2006. Orlando, FL.

[4] Fewster, Mark, and Graham, Dorothy. Software Test Automation. Addison Wesley, 1999.

[5] Hetzel, W. The Complete Guide to Software Testing, QED Information Systems, Wellesley, MA, 1988.

[6] Kaner, Cem, Bach, James, and Pettichord, Bret: Lessons Learned in Software Testing. A Context-Driven Approach. John Wiley & Sons, 2001.

[7] Lieberma, H. and Fry, C. Will Software Ever Work, Communication of the ACM 44, March 2001, pp. 122-124.

[8] Mallory, R. Steven. Software Development and Quality Assurance for the Healthcare Manufacturing Industries. Interpharm/CRC, 2001.

[9] McGraw, Gary. Software Security: Building Security In. Addison-Wesley, Upper Saddle River, NJ, 2006.

[10] Myers, J. Glenford. The Art of Software Testing. John Wiley and Sons, 1979.

[11] Nguyen, Hung, Johnson, Robert, and Hackett, Michael. Testing Applications on the Web (2nd Edition): Test Planning for Mobile and Internet-Based Systems.

[12] Robert V. Binder: Testing Object-Oriented Systems: Objects, Patterns, and Tools. Addison-Wesley Professional, 1999.

[13] Sykes, D. and McGregor. Practical Guide to Testing Object Oriented Software, Addison Wesley, Reading, MA, 2000.

[14] Zadeh, J. and DeVolder, D. Software Development and Related Security Issues, IEEE – SoutheastConf, 07. March, 2007. Richmond, VA.