

Software Quality and Testing

Hassan Pournaghshband
Asaleh Sharifi

School of Computing and Software Engineering
Southern Polytechnic State University

Email: hpournag@spsu.edu

Shahriar Movafaghi

Information Techonology Department
Southern New Hampshire University

Email: shahriar@movafaghi.com

Abstract

Software testing is a major part of the software development process that must be carefully carried out in order to produce reliable and dependable software products. An effective software testing technique can play a significant role for engineering high quality software products. It is the purpose of this paper to address major issues of software testing practices, and to propose possible solutions to some of the very common potential problems regarding software quality. More specifically, we examine some of the key problems of software testing, characteristics of people responsible for the job, how quality testing can be done by these people, and finally we show how software quality can be improved through effective testing.

1. Introduction

In as much, the need for high quality software and software-based services is more critical than ever before. Business competitiveness and bottom line profitability is now dependent on the quality of the software. Many well

known and not so well known events have been attributed to software bugs or software related problems. These events have cost cooperate competitiveness and millions in lost earnings. Many recent major computer systems failures caused by software bugs are given in [1]. As an example, “in July 2004 newspapers reported that a new government welfare management system in Canada costing several hundred million dollars was unable to handle a simple benefits rate increase after being put into live operation. Reportedly the original contract was never tested for its ability to handle a rate increase” [1]. While software problems take the blame, it is in fact the human equation that is ultimately responsible. Incorrectly established requirements, inadequate development methodologies, processes, and procedures to evaluate the quality of a software product are only some of the true drivers of the problems attributed to software bugs that have been discussed in literature. Quality is a subjective term and the fact that quality is subjective has led to a myriad of perspectives on quality as well as numerous definitions. The fact is, software quality must be defined and measured for improvement before high quality can be achieved. For

our purposes here, quality as defined in [2] is “the degree to which a software product possesses a specified set of attributes necessary to fulfill a stated purpose.” To engineer high quality software products we must among other quality artifacts, have a software quality program. Such a program must establish requirements for quality. It must also establish, implement, and enforce methodologies, processes and procedures to develop, operate and maintain the software [2]. Effective software testing has always been crucial to producing reliable and dependable software products. It is the purpose of this paper to examine some of the common software testing problems and address issues regarding an effective software testing technique. These common problems along with suggestions on how to deal with them are discussed in Section 2. Conclusions and recommendations for further research are given in Section 3.

2. Software testing: issues and concerns

While software testing is a major part of the software development process, it is the least understood part. Documented procedures for testing software must be in place before testing begins. The lack of software testing procedures or using procedures that are not clearly and completely defined often results in time delays and cost overruns. Testing is an important part of the software process and a task that must be carefully carried out throughout the software development life cycle. For example, requirements must be checked during requirements analysis phase, specifications must be checked during specification phase, and during implementation not only each

module must be tested separately, the product as a whole should be checked. To better discuss software testing issues and learn how to deal with them specially regarding software quality, we have come up with the following major questions that we will focus on them in this paper.

1. What are the potential key problems of software testing?
2. What are the attributes of software quality requiring quality testing?
3. Who are the key people involved and responsible for quality testing?
4. How can quality software testing be done by these people to overcome the key problems?
5. How can software quality be improved through good testing?

Below, in the following five subsections, we will elaborate on these issues and give a detailed discussion for each of the above questions.

2.1 What are the potential key problems of software testing?

Several key problems of current software testing practices are given in [3.] Below we discuss some of the very common of these problems and show how software quality can be improved through elimination or alleviation of each of these problems. While all of these problems are considered “key” problems in a sense that they affect the quality of the product, some of them can have more sever consequences than others. Therefore, we discuss these problems in order of their importance (i.e., their severity regarding their negative effects on software quality.)

Poor Planning – A careful test plan can help the testers to perform a quality job. On the other hand, poor planning can negatively affect the process which could lead to production of low software products. Poor planning is one of the major problems of software testing practices. Software can be tested at various phases of software development and with various degrees of strictness. Like any development activity, software testing consumes effort, and effort costs money. A poor testing plan not only can distress the quality of the software product, it also can increase the development cost. Software developers should carefully plan for the testing process and specify what percent of the project effort should be devoted for testing related activities including verification and validation activities. This, with no doubt, would increase the effectiveness of the testing outcome and thus, would lead to an increase in quality as well as a decrease in cost of the end result. To have a good test plan we have to carefully select strategies for testing. Test strategies are the general rules, principles and rules to support the goal of effective and efficient testing. To have a good test planning process, according to [4], we have to establish a clear criterion for each phase, establish common commitment and expectations, complete test planning processes at a reasonable time, promote reasonable creativity and flexibility, generate appropriate documentation, and provide the opportunities for catching errors.

Poor coordination - Poor coordination between software testers and software developers results in lower quality software product. Testers and developers should have good coordination in debugging activity, meaning that they

work side by side in detecting errors and bugs and fixing them appropriately. When testers find a bug they should communicate with coders and let them know and, once coders fix the bug and send it back to testers there should be a mechanism to make sure that nothing gets lost or forgotten during these communications. We also have to make sure that the link between testers and coders is clear and direct [3, 4, 5.]

Lack of user involvement - No matter how well we think we know about application domain and how knowledgeable we are, we always need the user involvement because we always should consider users' feedback to improve our software quality (i.e. user friendliness of our software) and make sure that we have met users' requirements [3.] Experience has shown that user involvement in the whole period of software development will result in a better quality software product. This is because it decreases the likelihood of errors resulting from misunderstanding the application domain.

Shortcuts in testing - Shortcuts in testing may result in unrecognized bugs and errors, which is going to cost us a great amount of time and money later. So if we look at it in a long term and see the consequences that making shortcuts in testing is going to have for us, we would rather to perform a test completely without any shortcuts. Indeed, shortcuts will only save us time and money at that moment but not in the long term.

Poor Testability - Testability could be defined as testing complexity of the software. When software has high testability, testing and finding bugs will

be easier and as the result there will be less undetected bugs. On the other hand, for software products with low testability finding bugs will be harder because more tests will be required to find a bug [3, 5.] In the case of hard to test and verify code, the code must be rewritten in a more testable way and this itself can become very time consuming.

2.2 What are the attributes of software quality requiring quality testing?

Among many attributes of software quality the five most significant ones that software engineers usually try to have a good balance of them are usability, efficiency, reliability, maintainability, and reusability. Although quality testing can contribute to all these attributes, but it directly plays a vital role regarding reliability of the software. The higher the testing quality, the more reliable (i.e., fewer failures) the software. Therefore, for those applications that reliability is a must, testing must be done thoroughly and with extreme care to avoid the software failure in the first place and to ensure an efficient and effective recovery in case of failure.

2.3 Who are the key people involved and responsible for quality testing?

Among many factors for successful software testing practices choosing right people for the job plays a significant role. It is very important to assign only those people who have required skills for performing a quality job. More specifically, the software testing team

members should possess not only testing skills, but they should also have application domain skills, and technical skills [4.] Assigning new hires or failed programmers, for whatever reason, can easily lead to the job failure and/or ultimately production of low quality software. The attitude of the testing team members, though not as vital as their skills, is also crucial for a successful job. In addition, the significance of the management role in the process should not be overlooked. The management continuous involvement and support is essential all the way through the testing and all phases of software development. This of course might not easily happen unless the management is properly educated in this regard.

2.4 How can quality software testing be done to overcome the key problems?

To produce high quality software we must, among other quality artifacts, have a software quality program. Such a program must establish requirements for quality. There are two main concerns that have to be taken into consideration when developing software. One involves with the client's needs (i.e., we have to make sure the final product satisfies their needs) and the other one is to produce an error-free product. Ideally, software development processes should be so advanced that no errors will enter a software system during development. Current practices can only help to reduce the number of errors, and not prevent all errors. However, if the best practices were available, it would be risky to assume no errors enter a system. The use of error analysis allows for early error detection and correction. When an error made early in the software development

life cycle goes undetected, problems and cost can accrue rapidly. An incorrectly stated requirement may lead to incorrect assumptions in the design, which in turn causes subsequent errors in the code [6.] It is interesting to know that average cost per defect, if found in production, is very high (it is well over \$100K), while it is extremely low (less than \$100), if detected during requirements analysis phase. Of course, for many cases, not all defects can be detected as early as in the requirements phase, but every effort should be made to not to delay them until the production. In fact, for most cases, the objective should be to detect defects as early as possible. Among a variety of issues that can be considered regarding quality testing, for almost all cases, we have to deal with the following three issues:

1. We should distinguish between different types of defects,
2. We should come up with good strategies for efficient detection of each type of defects, and
3. We should write test cases with “quality” in mind.

We first need to distinguish between different types of defects. It is easier to carry out the testing procedure if we have prior knowledge about potential defects that can occur. To come up with effective strategies for efficient detection and correction of each type of defect, we need to identify parameters for such strategies. In addition to the above issues, as discussed earlier, a good plan can play a vital role as far as quality is concerned. A careful test planning can help testers to appropriately perform their job and achieve their goals. It involves many steps such as establishing test objectives, designing test cases,

writing test cases, executing test cases, and evaluating the results [7.]

2.5 How can software quality be improved through good testing?

To discuss how software quality can be improved in an organization, regarding good testing, we need to learn how the organization views the quality and how they prefer to implement it. Some of the good candidate areas for improvements are organizational commitment, participative testing, focus of testing, better planning, quest for continuous improvement, and design of testability [2.] First and foremost, it is very crucial for the organization to commit itself or quality may be compromised and not improved. An active participation and an emphasis on testing along with an effective planning can also well contribute to software quality improvements. Continuous software quality improvement, like everything else, requires continuous feedback and quest.

3. Conclusions

In this paper, we have provided an overview of software testing, and have discussed what makes a software testing technique an effective one for producing quality software products. We have focused on five major questions regarding quality testing, and have given suggestions for overcoming those problems. A good testing strategy must possess a series of properties. We have discussed some of the major properties and their role in production of software quality. Presently, we are investigating

the issues regarding the list of properties necessary for such strategies to find a more complete list of those properties.

References:

[1] Lexico Publishing Group. LLC.
<http://dictionary.reference.com/>

[2] Ron Kenett. "Software Process Quality: Management and Control". Marcel Dekker, Inc. 1999.

[3] Nasib S. Gill. "Factors Affecting Effective Software Quality Management Revisited". ACM SIGSOFT Software Engineering Notes, March 2005 Volume 30 Number 2.

[4] Rex Black. "Critical Testing Processes: Plan, Prepare, Perform, Perfect". 1st Edition, Addison-Wesley, 2004.

[5] Vaughn T. Rokosz, IBM. "Long-Term Testing in a Short-Term World". IEEE SOFTWARE, Published by the IEEE Computer Society, 2003, 0740-7459.

[6] Glenford J. Myers, "The Art of Software Testing", John Wiley & Sons, 1979.

[7] Shari L. Pfleeger, "Software Engineering – Theory and Practice", Prentice-Hall, 2005.