A Roadmap for Effective Software Testing
Sheikh Umar Farooq1 and S. M. K. Quadri2
1,2P. G. Department of Computer Sciences, University of Kashmir
1sheikh.umar.farooq@gmail.com and 2quadrismk@hotmail.com

ABSTRACT
Software testing is the dominant approach in industry to assure software quality. Although crucial to software quality, software testing is mostly less formal and rigorous than it should, and a primary reason for that is because it is not performed systematically. At present more defects are probable due to increased complexity of software while on the other hand organizations cannot afford to let slip many bugs in the field, so software testing should be a balanced exercise in resource and quality management. Serious efforts are required to make software testing process efficient and effective. Although, different organizations have their own software testing guidelines but in general, there are some basic guidelines which must be followed by them to perform testing more effectively and efficiently. The primary focus of this paper is to recommend guidelines to maximize the success probability of software testing process.

KEYWORDS

1. INTRODUCTION
Software testing is the process of executing a program with the intent of finding errors [1, 2]. Software testing is also used to test the software for other software quality factors like reliability, usability, integrity, security, capability, efficiency, portability, maintainability, compatibility etc. For many years, the major software development companies have only focused on just the development of software rather than caring enough for its ‘verification’ and ‘validation’. But with time and with the losses that companies have experienced because of faulty software, software testing has appeared as an extremely important part of the software development cycle. Successful testing is a critical concern for most of the software majors. While some leave it to their in-house human resources to take care of, some pay extra to hire trained staff, specifically skilled to take care of core software testing manually. It is necessary to understand importance of successful software testing. Software testing does not necessarily deal only with the inefficiencies of the software application or bugs within it. It rather safeguards the software from being labeled as defective – which may create a negative ripple on the market regarding the product as well as the parent company itself. When a company develops a product, the acceptance of the product among its target audience is a vital factor regulating its critical and commercial success. Software testing should be performed efficiently and effectively, within the budgetary and scheduling limits. Due to large number of testing limitations like Exhaustive (total) testing is impossible, compromise between thoroughness, time and budget, it is impossible to be sure that we have removed each and every bug in the program [3]. Following established guidelines can make testing easier and more effective, and can also ensure that testing goals are achieved to its maximum.

2. GUIDELINES FOR EFFECTIVE TESTING
Testing guidelines are suggestions for the testing team, so that they can reveal defects in the system and will make testing process more successful.

2.1 START EARLY AND BE PREPARED!
Software testing has to be planned much before the actual testing begins, possibly right from the inception of the software itself. If you want to find errors, start as early as possible. Fixing errors at early phases cost less as compared to later phases. Reviewing helps to highlight defects and if conducted early in the process will save money by reducing the cost of expensive repairs or corrections. Defects found in the requirements phase are on average 100 times cheaper to correct than the same defects found in production [4]. Figure 1 depicts the increase in cost of fixing bugs detected/fixed in later phases.

![Figure 1. Cost of fixing bugs in different phases.](image-url)
documents (e.g. requirements) also help to prevent defects appearing in the code [4]. Testing should be introduced from the outset, this way you have an objective look at the project feasibility and its risks prior to committing valuable effort. Testing should start early in the development process, you cannot afford to wait for code to start testing your application. Make sure you start reviewing your documents right from the outset; as well reviewed documents tend to result in 65% less defects being carried forward into later phases of the development cycle. This can amount to huge saving of time and money. Preparing your test execution is paramount to getting your project completed in the shortest possible timescale. Valuable time can be saved during the test execution phase by specifying your test scripts and creating a test scenario before the application under test is delivered.

2.2 DEFINE TEST PLAN
Building your test plan lets you see the entire project written out, and figure out what actions you’ll need to take to accomplish your business objectives. Test plan should efficiently meet the needs of an organization and clients as well. Test Plan usually describes test scope, test objectives, test strategy, test environment, deliverables of the test, risks and mitigation, schedule, levels of testing to be applied, methods, techniques and tools to be used. The testing is conducted in view of a specific purpose (test objective) which should be stated in measurable terms, for example test effectiveness, coverage criteria. Although the prime objective of testing is to find errors, a good testing strategy also assesses other quality characteristics such as portability, maintainability and usability.

2.3 SELECT A TESTING METHOD WHICH FITS YOUR NEEDS.
There is no such thing as a ‘universal test approach’. Testing processes/methods that work for one company or project may be totally inappropriate for another. Testing is done differently in different contexts [5]. Testing should be appropriate and different for different points of time. For example, a safety-critical software is tested differently from an e-commerce site. Even a system developed using the waterfall approach is tested significantly differently than those systems developed using agile development approach. Even the objectives of testing differ at different point in software development cycle. For example, the objective of unit and integration testing is to ensure that code implemented the design properly. In system testing the objective is to ensure the system does what customer wants it to do [6]. Selection of testing methods should be done taking into account different aspects/factors (both internal and external) which include but are not limited to: the system being tested, for what purpose, regulatory standards, user requirements level and type of risk involved and resources available for testing, time allotted for testing.

2.4 PRIORITIZE

In a normal application or system, there are far too many functionalities and to many possibilities to test every single one of them. Testing everything (all combinations of inputs and preconditions) is not feasible except for trivial cases. It is impossible to test a program sufficiently to guarantee the absence of all errors [1]. It is important to test those aspects of your application where the risk of damage or frequency of use is expected to be the highest. Instead of exhaustive testing, we use risks and priorities to focus testing efforts more on suspected modules as compared to less suspected and infrequently encountered modules. Remember: No Risk, No Test.

2.5 DESIGN EFFECTIVE TEST CASES
Complete and precise requirements are crucial for effective testing. If you do not understand the user requirements and architecture of the product you are testing, then you will not be able to design test cases which will reveal more errors in short amount of time. An effective test case has high probability of finding more undiscovered errors in less time. In addition to that effective test cases have good test coverage [7].

2.6 TESTING MUST BE DONE BY DIFFERENT PERSONS AT DIFFERENT LEVELS
Different purposes are addressed at the different levels of testing. Factors which decide who will perform testing include the size and context of the system, the risks, the development methodology used, the skill and experience of the developers. Testing of individual program components is usually the responsibility of the component developer (except sometimes for critical systems); Tests at this level are derived from the developer’s experience. Testing at system/sub-system level should be performed by the independent persons/team. Tests at this level are based on a system specification [8]. Development staff shall be available to assist testers. Acceptance Testing is usually performed by end user or customer. Release Testing is performed by Quality Manager. Figure 2 shows persons involved at different levels of software testing.

3. TEST EVALUATION
We should have some criterion to decide whether a test is successful or not. If limited test cases are executed, the test oracle (human or mechanical agent which decides whether
program behaved correctly on a given test [9]) can be tester himself/herself who inspects and decides the conditions that makes test run successful. When test cases are quite high in number, automated oracles must be implemented to determine the success or failure of tests without manual intervention. One good criterion for test case evaluation is test effectiveness (number of errors it uncovers in given amount of time). In addition to that we can take other factors into consideration also like type of faults etc.

4. MEASURE
We should gather data to monitor efficiency and effectiveness of the test process and product under test. We should utilize metrics which provide the information we need to know about test process, i.e. Number of test cases designed, number of test cases executed, number of test cases passed, number of test cases failed, defect density, number of defects found in the first month of production, etc and for product under test, i.e. Number of failures experienced in a time interval, time interval between failures, mean time between failure (MTBF). Metrics provide information that will help to focus and evaluate product and process improvements.

4.1 KNOW WHEN TO STOP TESTING.
Software testing is an ongoing process, which is potentially endless but has to be stopped somewhere. Under normal circumstances, we need to set exit criteria to decide in advance when testing should stop. Realistically, testing is a trade-off between budget, time and quality [10]. The effort spent on testing should be correlated with the consequences of possible program errors [11]. The possible factors for stopping testing are:
1. The risk in the software is under acceptable limit.
2. Coverage of code/functionality/requirements reaches a specified point.

4.2 PRESERVE
The strength of software testing partially lies in the structured preservation of testware. Documenting both the successes and failures helps in easing the process of testing. What was tested, and how it was tested, are needed as part of an ongoing testing process. We should agree on the common established documentation methods to avoid the chaos and to make documentation more useful in error prevention in future. Test cases should be stored in such a way that they can be reused in later projects- or even during the maintenance phase. Preserving test ware can save 30% to 50% in overall test effort.

5. CONCLUSION
Software testing has become indispensable to every software development project. It should be kept in mind that proper implementation of the testing process is also a big a challenge. Every organization is unique and has its own needs. It is difficult to suggest “one size fit all” solution. But following the guidelines given in this paper increases the probability of success of the testing process. These guidelines might prove to be quite effective taking into consideration business architecture, available human resources, nature of the project, level of quality needed and budget constraints.

REFERENCES