Software Testing Process: A Prescriptive Framework

R. A. Khan
Department of Information Technology
B. B A. University (A Central University)
Lucknow-UP, India
00-91-522-2964765
khanraees@yahoo.com

R. K. Choudhary
Asia Pacific Institute of Information Technology
SD India, Panipat- 132 103, India
00-91-180-6532555
Ramesh_chy@rediffmail.com

ABSTRACT
Early identification of defects and prevention of defects migration are key goals of the testing process. It is highly desirable to optimize the test activities for a fast time-to-market while delivering a product that meets quality expectation. Software engineering research has led to a raft of systematic testing activities applicable during the development life cycle. Software test process elaborates various testing activities and describes which activity is to be carried out when. Given the need and significance of phased approach of testing, this paper proposes a prescriptive framework elaborating testing activities to be carried out while integrating it within the development life cycle.

Categories and Subject Descriptors
D.2 [**Software Engineering**]: Testing and Debugging – *diagnostics, testing tools*

General Terms
Verification

Keywords
Testing, Test Process, Test Cases, Test Data, Test Execution, Test Design.

1. INTRODUCTION
Software testing is an important process during software development life cycle. An appropriate testing can increase software product's quality. In the early days of 50's, the testing was seen as a follow on activity which involved detection and correction of coding errors. It began to be decoupled from debugging in late 50's. With the advancement of development methodologies, the importance of testing increased through experience and economic motivates in 60's. More rigorous testing methods were introduced and more resources made available. In 70's, testing was seen more as a means of obtaining confidence that a program actually performs as it was intended. Quality was introduced in 80's as big issues to software development organizations. As a result, software industry started taking testing as a challenge to achieve quality product. Tools and techniques appear in 90's in order to address testing in a better way, and for making testing more effective[1]. Now a day, software testing has not only evolved for finding defects or bugs in the software, it has become a complete dedicated discipline of evaluating the quality of the software. Therefore, it has become imperative to test the software components and software as a whole for integrity, capability, reliability, efficiency, portability, maintainability, compatibility and usability of the software.

Traditionally testing has been seen as a single phase in the life cycle. But recently there have been some efforts made to define testing as a life-cycle activity. It is established that the testing process is a phased activity which encompasses different states with the set of activities defined in each phase[2]. Now the development communities have started to acknowledge the same. Capturing test activities and their interdependencies with the development life cycle results in a test development life cycle that helps to ensure that the resultant test environments meet the needs of a program's integration and verification by test activities[3]. Software engineering research has led to a raft of systematic testing activities applicable during several stages of software development. One of the most common testing processes proposed by Sommervilla in 2000 comprises of four states including design test cases, prepare test data, run program with test data and compare results to test cases. Several works are cited in literature on developing software test process. All of the contributions made moves around the six generic phases in test process including test planning, test design, test case preparation, test execution and test cycle closure. Unfortunately, all of the activities discussed in these processes are highly dependent on the development methods. This limits the effective implementation of the each of the activities with development process.

2. SIGNIFICANT NEED
No matter how well the software is developed, in a world of complex software, the only way to ensure that application software are secure and stable is to rely on software testing. There is a great demand to develop a scientific, structured approach to testing that will find and fix the highest number of defects in the shortest span of time at most reasonable cost. Researchers and practitioners are advocating minimizing test activities for a fast time to market while delivering a product that meets quality expectations[4]. Software test process elaborates various testing activities and describes which activity is to be carried out when. Every organization has to undertake testing of each software product. The way it is conducted differs from one organization to another. This refers to the life cycle of the testing process which consists of various phases through which a software product or component goes and describes the various activities pertaining to testing.

Since decades, researchers and industry professionals are advocating to carryout testing process from the initial phase of development life cycle. Software testing has been proven as a series of activities carried out methodologically to help certify the software product quality. It is also true that enabling communication and synchronization between the various groups that have input to the overall testing is mandatory and is possible only through testing process. Quality assurance checks need to be carried out to ensure the application is error and risk free. Even though testing differs between organizations, there must be a common approach to carryout the different activities within the testing to achieve the set objective, in the shortest time, optimal resources and lesser effort. There appears a need to have such a framework prescribing testing activities in each phase of testing process, which, on implementation can generate the results as per expectation. Given the need and significance of mapping a testing process with the development life cycle, a prescriptive framework for test process is proposed in the next section.

3. THE FRAMEWORK
The challenges of implementing testing activities within the development life cycle with unpredictable run-time environments make the already difficult problem even harder. In a typical commercial development organization, the cost of ensuring that a program will perform satisfactorily in terms of its functional and non-functional specifications within the expected environments via testing can easily range from 50% to 70% of total development cost[5]. Therefore, it becomes mandatory and even challenging to investigate what is involved in these activities that make it so expensive and important. It is

DOI: 10.1145/1988997.1989019 http://doi.acm.org/10.1145/1988997.1989019

highly desirable to lookup what needs to be done at each activity, how to optimize the time and finally elucidate the areas where it could be used to save time. If, during testing process, optimal activities are identified and implemented, significant time could be saved with the adoption of innovating testing techniques. In addition to this, an effective and prescriptive process of testing specifying very clear prioritized activities may be advantageous in different perspectives as follows:

- A good and efficient testing process may lead to a dependable tested software;
- Implementing a correct process, scheduling in a systematic way and applying them smartly may result a significant reduction in time and effort with increased quality;
- Development with the prescriptive test activities may lead to quality product with shortest span of time;
- Through efficient testing activities, the power of automation could be exploited;
- An appropriate and accurate test activity implemented during development may make the software more profitable;

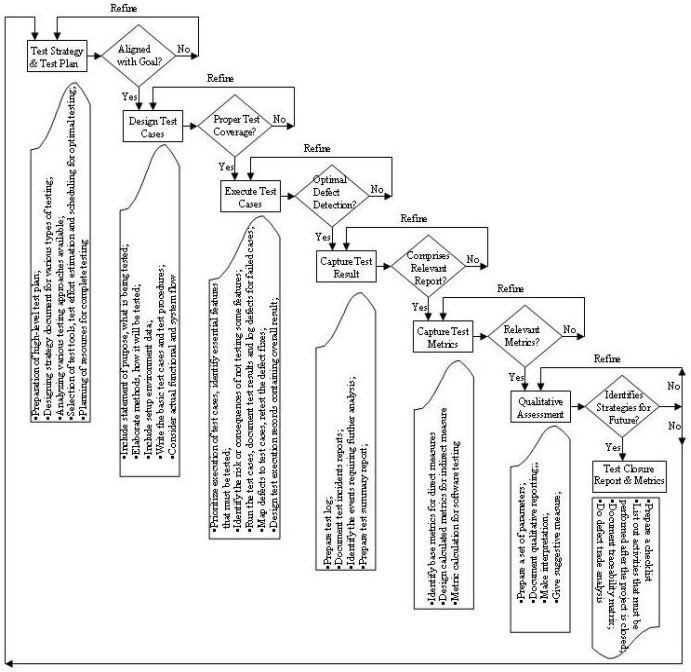

Figure 3.0(a): Software Testing Framework

The above advantages necessitate the need to migrate from an immature, adhoc way of working to have a full-fledged testing process. Effective test process requires a wide verity of skills and activities, including the identification, collection and documentation and supporting materials. Within this context, test activities should be prioritized with the ultimate objective of delivering maximum benefits to the end-users[6]. Therefore, there appears an urgent need of having a commonly accepted process of software testing that can be integrated with each phase of software development life cycle. A prescriptive framework is proposed in figure 3.0(a) in order to cater the need. The proposed process comprises of the seven phases including *Test Strategy and Test Plan*, *Design Test Cases*, *Execute Test Cases*, *Capture Test Results*, *Capture Test Metrics*, *Qualitative Assessment and Test Closure Report*. Each phase is briefly described in the following section.

Figure 3.1(b) shows test life cycle process stating as a separate process that has a very tight interconnection with development activities. If quality of the software being developed can not be compromised then the test life cycle process should be involved from the inception of the project and requirement specification should be analyzed thoroughly. A rigid deadline and schedule can contribute greatly towards the timely initiation and ending of both software test life cycle process and development life cycle.

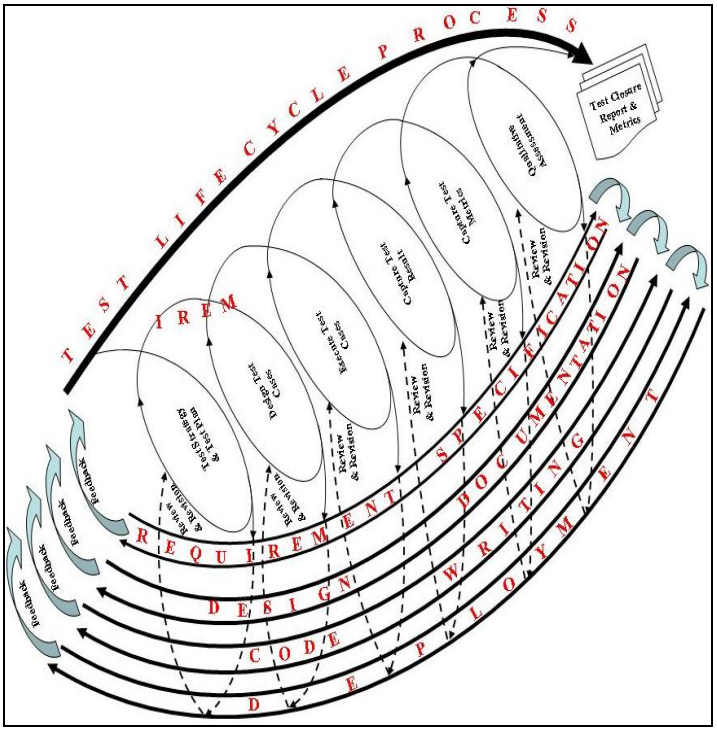

Figure 3.0(b): Test Life Cycle Process integration within the Development Phases

3.1 Test Strategy and Test Plan

Without a good plan no work is a success. The testing process of software should also require good plan. Test plan documents bring in a process-oriented approach. Test planning is an act of describing the scope, approach, resources and schedule with the objective of revealing as many errors in the software as possible. It is an ongoing process throughout the project life cycle, and is integrated with each development phase. An appropriate test plan enables the mapping of tests to the software requirements and defines the entry and exit criteria for each phase of testing. Proper planning would greatly reduce the risk of low quality software. During test planning , high level test plan is prepared which identifies the items to be tested, the features to be tested, the types of testing to be performed, the personnel responsible for testing, the resources and schedule required to complete testing, and risks associated with the plan. Test planning has become one of the most important phase of software test process. Without a detailed test plan, an end product can be bug ridden and in turn unsuccessful. Some of the suggestive checks are proposed in table 3.1. A prescriptive step in preparing test strategy and test plan is depicted in figure 3.1.

Table 3.1: Suggestive Checks for Test Plan

• Items to be tested are identified;
• Features to be tested are identified;
• Types of testing to be performed;
• Personnel responsible for testing are identified;
• Resources required for complete testing are identified;
• Schedule for test completion is decided;
• Risk associated with test plan is discussed;
• Testing tool selection criteria is discussed;
• Guideline for test effort estimation is presented;
• Mapping of testing to the software requirements are done;
• Entry and exit criteria for each phase of testing is defined;

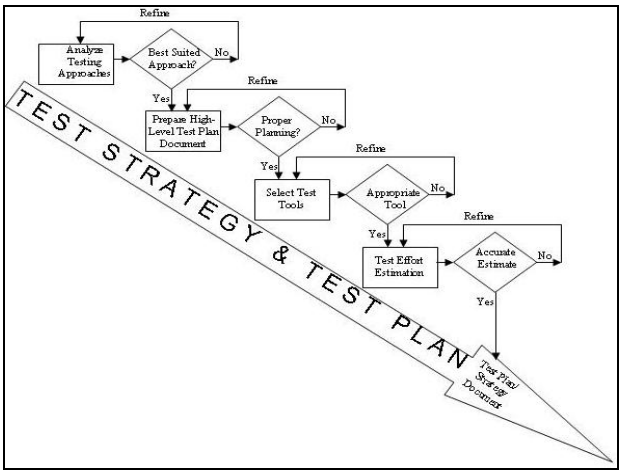

Figure3.1: Test Strategy & Test Plan

3.2 Design Test Cases

The basic objectives of writing test cases are to validate the testing coverage of the application. This phase involves creation, verification, and rework of test cases and test scripts. After designing test cases and scripts, test data is identified and reviewed. Test cases should be prepared based on the four scenarios including positive scenario, negative scenario, boundary conditions and real world scenario. Test cases are actually the valuable assets. The test cases should be developed in an economical ways and no unnecessary steps should be there. The development should keep on going and the cases developed are repeatable and reusable. The created test cases should be traceable to requirements. A check should be there whether the developed test cases are appropriate for test environment and testers. All of the created test should be self standing and should not depend on writer. It should also be self checking. Some of the suggestive checks are proposed in table 3.2. A prescriptive step in preparing test design is depicted in figure 3.2.

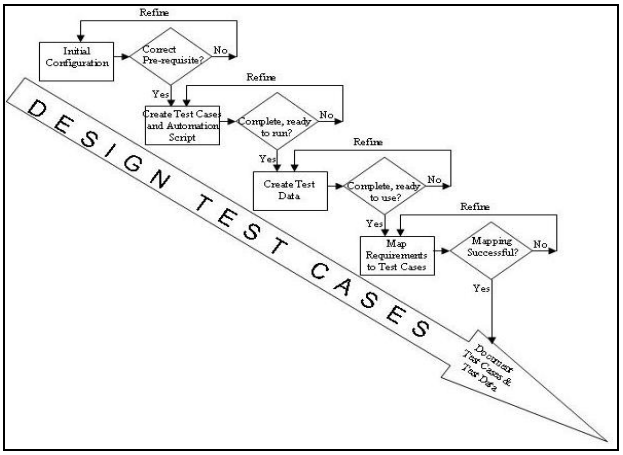

Figure 3.2: Design Test Cases

Table 3.2: Suggestive Checks for Test Case Design

- Setup complete, correct and coherent;
- Making cases too long is avoided;
- Pass or fail results are clearly mentioned;
- Whether tester or system does action is clearly specified;
- Test cases created are accurate;
- Test cases designed are economical;
- Developed test cases are repeatable;
- Test cases are traceable to a requirement;
- The designed test cases are independent of the writer;

3.3 Execute Test Cases

Test case execution is the process of executing test scripts in a logical sequence with specific input data. The results are monitored and output is recorded in test sheets. If after execution, the desired output is achieved, the test cases is said to be successfully executed. In this phase testing is carried out based on the test plans and test cases prepared. Out of a large cluster of test cases, it is needed to scientifically decide their priorities of execution based on some rational, non-arbitrary criteria. Prioritizations of the test cases are done with the objectives to reduce the overall number of test cases in the total testing feat. Test results are documented and log is prepared with defects for failed cases. Accordingly test plans and test cases are reviewed and updated if required. Identified defects are mapped with the test cases. Finally, the defect fixes are retested and defects are tracked to the closure. The test execution records map the test environment information to the test cases. The test execution records also contain the overall result including pass, fail, blocked associated with the execution of test cases. Some of the suggestive checks are proposed in table 3.3. A prescriptive step in executing test cases is depicted in figure 3.3.

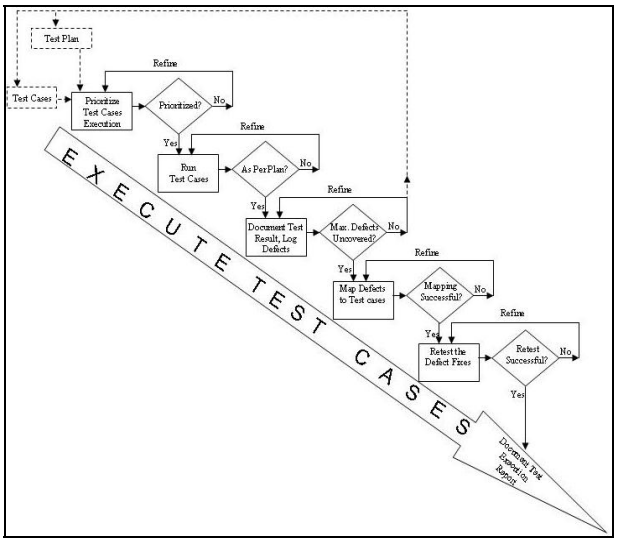

Figure 3.3: Test Cases Execution

Table 3.3: Suggestive Checks for Test Case Execution

- Test plan and test scripts are available;
- Test environment is ready;
- Test case execution is prioritized;
- Test data setup is done;
- Test results are documented;
- Log is prepared with defects for failure cases;
- Defects are mapped with test cases;
- Defect fixes are retested;
- Test execution records are prepared;
- Test execution records map the test environment;
- Test execution records contain overall result;

3.4 Capture Test Result

After test case execution, the identified bugs are fixed and retesting is done to declare the result as pass or fail. Test reports are documented properly. A test log is prepared to present chronological notes of the relevant details about the execution of the tests. After detailing the relevant information, any event that happens during the test that requires further analysis is documented. A test summary report is prepared at last which summarizes the results of the test activities associated with one or more test design specifications and provides evaluations based upon such results. Some of the suggestive checks are proposed in table 3.4. A prescriptive step in capturing test results is depicted in figure 3.4.

DOI: 10.1145/1988997.1989019 http://doi.acm.org/10.1145/1988997.1989019

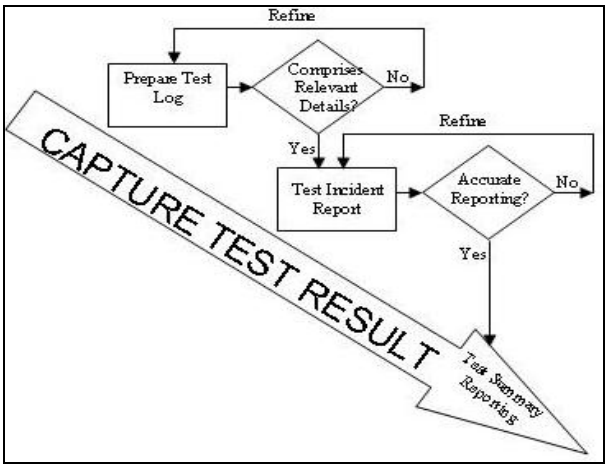

Figure 3.4: Capturing Test Results

Table 3.4: Suggestive Checks for Capturing Test Results

- A chronological notes of the relevant details about the execution of test is present;
- Any event that has happened during test requiring further analysis is reported;
- Results of test activities associated with one or more test design specification is summarized;

3.5 Capture Test Metrics

Test metrics assist in the improvement of the software development process by providing pragmatic, objective evidence of process change initiatives. Test metrics helps in taking decisions for next phase of activities. It assists in taking decisions on process or technology change. Software test metric is a useful for test managers, which aids in precise estimation of project effort, addresses the interest of metric group, software managers who are interested in estimating software test effort and improve both development and testing processes. Basic metrics are used to provide project status reports to the test lead and project managers. Many of the basic metrics are simple counts that most test analyst already tracks in one form or other. Calculated metric converts the basic metric data into more useful information. Some of the suggestive checks are proposed in table 3.5. A prescriptive step in capturing test metrics is depicted in figure 3.5.

Table 3.5: Suggestive Checks for Capturing Test Metrics

- Metrics for total number of test cases are identified;
- Metrics for total number of test cases executed are identified;
- Metrics for total number of test cases passed are identified;
- Metrics for total number of test cases failed are identified;
- Metrics for total number of test cases under investigations are identified;
- Metrics for total number of test cases blocked are identified;
- Metrics for total number of test cases re-executed are identified;
- Metrics for total execution are identified;
- Metrics for total passes are identified;
- Metrics for total failures are identified;
- Metrics for test case execution time are identified;

3.6 Qualitative Assessment

Only through efficient testing, quality and safety of an application can be guaranteed. Product quality measurement parameter is set forth in this phase in order to assess the quality of the product. A commonly accepted set of parameters is identified for which the assessment is to be made. On the basis of the quantitative results, a qualitative reporting of quality of the work product s made to the end user. Based on the qualitative assessment, interpretations are made and suggestive measures are proposed. Some of the suggestive checks are proposed in table 3.6. A prescriptive step in qualitative assessment is depicted in figure 3.6.

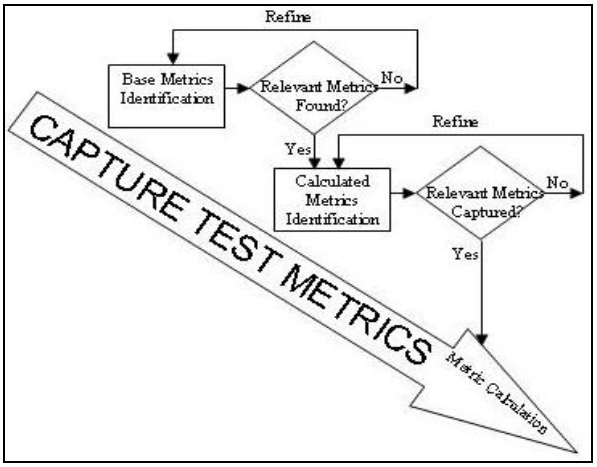

Figure 3.5: Capturing Test Metrics

Table 3.6: Suggestive Checks for Qualitative Assessment

- Commonly accepted set of parameters are identified;
- Quantitative measures are available;
- Qualitative reporting is made;
- Suggestions for improvement are proposed;

3.7 Test Closure Report

Once the test meets the exit criteria based on the time, test coverage, cost, software, critical business objectives, quality, the activities including capturing key outputs, lesson learned, results, logs, documents related to the project are archived and used as a reference for future project. Test closure document contains a checklist of all of the items that must be met in order to close a test project as well as a list of activities that must be performed after the project is closed.

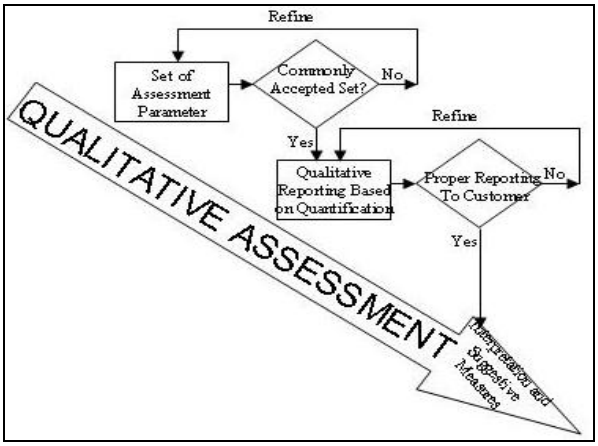

Figure 3.6: Qualitative Assessment

4. CONCLUSION

Phased testing process should be integrated within the development life cycle with the intent of finding errors at each phase well in advance in order to reduce developmental cost, delivery time and rework efforts. A prescriptive framework consisting of seven phase is proposed with the objective of identifying defects early and preventing defect migration. Suggestive checks for each phase of the testing process are presented to

DOI: 10.1145/1988997.1989019 http://doi.acm.org/10.1145/1988997.1989019

crosscheck their adherence while implementing testing. The proposed work may help testers to better understand and execute test in an efficient and effective manner.

5. REFERENCES

[1] A. Ireland, Software Engineering 4, The Software Testing Life-Cycle, School of Mathematical and Computer Sciences, Heriot-Watt University, Edinburgh.
Web Reference: www.macs.hw.ac.uk/~air/se4/pdflec/lec-1-life-cycle-a5.pdf

[2] D. Pemberton, and I. Sommerville, VOCAL: A Framework For Test Identification & Deployment, IEE Proc. Softw Eng. Vol. 144, No. 5-5, October-December 1997, pp. 249-260.

[3] J. K. Thompson, Recognizing the Test Development Life Cycle, ITEA Journal 2010; 31, pp. 103–108.

[4] C. Lovin and T. Yaptangco, Best Practices: Enterprise Testing Fundamentals, Dell Power Solusions, February 2006, pp. 52-54. www.dell.com/downloads/global/.../ps1q06-20050111-Lovin.pdf

[5] B. Hailpern and P. Santhanam, Software Debugging, Testing, and verification, IBM Systems Journal, Vol. 41, No. 1, 2002, pp. 4-12.

[6] L. Lazic and N. Mastorakis, Cost Effective Software Test Metrics, WSEAS TRANSACTIONS on COMPUTERS, Issue 6, Volume 7, June 2008, pp. 599-619.

DOI: 10.1145/1988997.1989019 http://doi.acm.org/10.1145/1988997.1989019