

Process Improvement through Software Testing

1. Introduction

Since several decades countless strategies have been developed to fight software quality problems, and numerous methodologies have been deployed using several standards and quality improvement models. The global IT industry has lived through countless process improvement exercises, but still software testing remains a challenge.

With time, business and systems requirement changes and thus the process of Requirements Management (RM) assumes relevance. A well-defined RM defines customer's requirements and guides the project towards customer satisfaction. It is only through implementation of appropriate standards that software can be tested for its functionality, speed, performance, user-friendliness and compatibility. It is also necessary to test requirement with respect to objectivity, feasibility, redundancy and clarity.

Quality assumes different definitions and is interpreted differently by different roles. Quality for developers means meeting specified requirements of an acceptable product, process and standards. Whereas the Customer defines quality as fitness for purpose, available at right time, place, form, price. The process adds to the third dimension of quality as the totality of characteristics of an entity (product or service) that bear on its ability to satisfy stated or implied needs.

Testing and debugging has been verified as the two important parts of software quality assurance. In testing the conditions are known. There are predefined procedures, standards and predictable outcomes that identify the debugging activity. Testing also needs detailed planning, scheduling and designing of activities for unearth things as many defects as possible so that defects can be eliminated during the debugging phase. Today testing starts right from the requirements stage to the analysis stage, design stage, coding, implementation and also post implementation activities in the entire life cycle of the product. Testing has presumed over a period of time a very procedural, systematic, sometimes called ruthless approach for finding defects, fixing them and verifying their removal. Still testing can be subcontracted to third party but debugging cannot be subcontracted and it has to be done by the development team. Testing can be automated and there are several automation tools that help this activity. Whereas automating debugging is still a dream.

Testing fails when it happens in the paradigm of optimism. Testing is also plagued by the negative attitude of software developers. It is a virtual challenge to their hard work. With CMMI taking over the lead in assessing the capability and maturity of the software industry, testing now can be segregated into activities for each KPA. The CMMI calls for V&V (Verification & Validation), with such a focus on testing as a continuous process throughout the life cycle; third party software testing has become a very big business in India and abroad.

With so many test procedures being executed throughout the lifecycle, there is no guarantee that the product does not carry any defects. It becomes more imperative to understand and find out various defects that cannot be unearthed by collecting and executing dozens of tests.

A test strategy should be formulated by the SEPG in the organization as a general guideline for the projects. At CMMI level 2, the test strategy may specify guidelines specific to the projects, whereas at CMMI level 3, the test strategy may specify guidelines specific to the organization. It is also imperative to test the testing technology and assess the risks involved in case of over testing.

Irrespective of the process, training, standards or the CMMI level, it is the maturity of every individual in the organization to use his others experience and test everything that he contributes in the overall software development process to create an international quality product. Peer Reviews, an important KPA at CMMI level 3 plays a major role in identifying and fixing the defects.

2. Software Testing Types: "70 Tests"

Software testing job was supposed to be a backend job in a traditional software development life cycle. But since last few years Software Testing is spanning over the entire life cycle right from the requirement stage to the design, code, testing, implementation and post implementations and product update. There are several types of tests, which are required to ensure a superior quality product.

The list of 70 tests that can be done at various points in the life cycle of the software development process is shown below. All these tests may not be required for each project, as every project needs a specific set of test depending upon platform, client requirements, geographical conditions and the CMMI level of the organization.

1. User Requirements Test: This is the test done at the time of preparing requirements document URS (User Requirement Specification). Here the requirements are tested for clarity, redundancy, feasibility, and objectivity. As soon as the requirements are tested the URS can be helpful in preparing the acceptance test plan so that user can test the software according to the URS.

2. System Requirements Test: From the URS we produce SRD or SRS (System Requirements Document) or (System Requirements Specification) which tests the validity of the system at the client end. As soon as the document is ready they can make a plan for a systems test conducted by the development organization before coordinating the plan for the Acceptance test.

3. Design Test: This is done on the basis of the design and the validity of the design where one needs System Design Document (SDD). It specifies various modules and the coupling between the modules. This in turn helps to plan the integration test, which is done after the unit test.

4. Static Test: A walkthrough or code reviews or design review, which does not need a machine, can do This test. It is a dry test.

5. Dynamic Test: This test needs to be done dynamically on the computer.

6. Black Box Test: Black Box Testing is not a specific test but Category of test and lot of the other tests fall in this category. This is based on external specifications without knowledge of how the system is constructed.

7. White Box: White Box test is not a specific test but a category of test and lot of other test is included in this category. White Box testing is based on knowledge of the internal structure and logic. What is Black Box to one layer is white box to another layer.

8. Functional Test: A Black Box test that validates functional requirements i.e. what the system is supposed to do. A function test matrix helps in testing various functions of the software.

9. Structure Test: Test that validates the system architecture. There are two type of structural test a) Internal Design Structure and b) Application Navigational Structure.

10. Internal Design Structure Test: This test checks the coupling and cohesion in the different modules of the software and it can use an impact analysis chart which helps in software maintenance as well as system understanding.

11. Navigation Structure Test: This is front- end design of the systems and checks how the user can navigate through various menus and submenus items to reach the functionality that he wants.

12. Unit Test: Unit is one small module of the software this is White Box test. It can be successfully implemented using unit Test plan.

13. Integration Test: Done after the Unit Test and it combines nearly successfully tested units into bigger modules and assisted by an Integration Test Plan.

14. System Test: Focuses on a complete integrated system to evaluate the functionality's of any user requirements and system requirements. It is a Black Box Test that can be facilitated by system test plan.

15. Acceptance Test: This is done by user or similar roles and it is specifically based on functionality and user requirements specification.

16. Alpha Test: It is done in the development environment in the presence of a client and entire testing is controlled by the developers.

17. Pre Beta Test: This test is done after the Alpha test and before Beta test. The test is done by the agencies or client similar to the current client and feedback is taken and all the defects are checked by the subject matter experts (SME). For e.g., the software is tested by three travel agents before actually being tested by travel agent client.

18. Beta Test: This is test in which the software is installed at the client environment for few days and client exercise full freedom to test according to the specification mention in the URS.

19. Regression Testing: Regression testing is done after debugging software to verify that no other errors are inserted due to the changes made to the software while debugging. Regression test is done for the first time after the unit test, integration test, system test and acceptance test.

20. The Statement Coverage Test: It ensures that every statement in software is executed at least once while testing.

21. Branch Coverage Test: It ensures that every branch in software is executed at least once. This takes care of all branching, looping or goto's, function calls, procedures etc.

22. Condition Coverage Test: It ensures that every condition (if, else, while, do, switch, case) is executed at least once to ensure that all the conditions are covered exhaustively.

23. Multiple Condition Coverage: This is a subset of condition Coverage test that ensures multiple condition coverage during execution.

24. Full Path Coverage: It ensures that all paths of the entire software be covered at least once. The Mc Gabe technique can be used to ensure full path coverage.

25. Boundary Value Analysis: It checks all the critical boundary value for every condition and each data values for the variable are tested properly to take care of truncating, rounding, approximation, etc.

26. Equivalence Partitioning Test: Ensures that all inputs and outputs are identified with the tests module and represents samples from all the partitions of possible input and output data.

27. Usage Test: This test checks the case with which the user can operate the program effectively.

28. API Test: This test is with reference to various API calls (Application Programming Interface) and their appropriateness of the use in a particular coding situation.

29. ODBC Test: This test checks the various combination of connectivity of fronted and backend in the client server environment decide by tester.

30. Bit Test: It checks the performance and validity of code and design for applications running on 32 bit operating system.

31. Gorilla Test: This is by random piano playing on the keyboard or random mouse clicks done by the tester to check how robust is the program against / illegal inputs and untrained users.

32. User Interface Test: This checks for how efficient are the interfaces for various forms, reports, menus, dialogue boxes that can help the user.

33. Stress Test: This checks whether the modules of application can work under various stress conditions and upto what time they can survive and operate efficiently. This is done with the help of data generation tools or burning test.

34. NT Test: These are the specific test that are done to test the application operation with respect to specific features of Windows NT or Unix operating system.

35. Applets Test: These tests check the functionality of Applets and its code.

36. Load Test: The Racing conditions and artificial load is created at a particular point of operation to test how the application behaves when there are multiple demands of resources like printing, reading from a device, port handling, communication with loading and Html page are happening at same time.

37. Printer Test: Here the various types of printers and the models and device drivers are tested for report generation, graph and charts.

38. Configuration Tests: Nearly overall integration of hardware integration and all the other collected elements like printer devices, device drivers, networks etc. Some terminals, mark terminus, memory and hardware configuration aspects are being tested for overall configuration.

39. Memory Test: The usage and the release of memory along with optimum usage of memory is tested. Memory allocation, de allocation and reallocation with respect to optimum use of memory.

40. Simulation Test: A real time situation is created within the control of developing organization with simulated user roles and real conditions to see how the software performance in real life situation. For e.g. A railway ticket booking system with 20 users standing in queue for booking of tickets for various trains.

41. Random Data Test: This helps in creating lakhs of data or records elements per second and tested for its validity in the software which helps you in identifying productive test cases of data, it is normally used in automated tools.

42. History Data Test: Past or history data is tested with the application where their results are already known in advance and are compared with the results generated by the software.

43. OS Test (Operating System): Every software needs the power; facilities and features provided by the operating system. Operating Systems test clearly means configuring the Kernel parameters to see or to match the requirements of application software to generate the best performance.

44. Network Test: This checks the speed on the network, connectivity, time, traffic and transfer of messages in a network. This test will help to determine the overall performance and the competence of the software in the networking environment.

45. Reusability Test: This test discloses how much amount of code can be reused in the future applications and also gives you idea of ROI (Return On Investments) for the reusability in the current product. Also if you are already reusing the existing code, then how effective is the reusability.

46. Object Test: It checks how comprehensive and complete is the object and what are the possibilities of inheritance, overloading and polymorphism with the object and how high is the reusability.

47. COTS Test: (Commercial Off The Shelf): This test allows you to test the functionality of COTS product.

48. Data Storage Testing: This test will identify the actual data volumes that are going to be handled by the software and checks the software against the data warehousing management of that volume of data.

49. Mutation Test: Mutation helps in finding more defects by introducing defects. If artificially ten defects are inserted into the program then there is possibility in debugging of ten defects, you may find twenty more defects. This is an abstract way of testing program, which is not always successful.

50. Conversion Test: This test is important for converting the program from one version to another version. For example Word 97 documents should be able to convert all the files of Word 6.0 into Word 97 and visa versa.

51. Release Test: You gather all the things that will go to the customer or to a manufacturer, and check for manuals, packaging, delivery etc.

52. Binary Test: Binary comparison between all files on these disks and those on the version you declared "good" during the final round of testing.

53. Virus Test: Test the product for viruses.

54. Integrity Test: It anticipates every major criticism that will appear in product reviews, or, for contract work. The product must live upto all claims made in the advertisements.

55. State transitions: Does the program switch correctly from state to state? State tables are made to checks the values assumed by variables.

56. Volume Tests: Study the largest tasks the program can deal with. You might feed huge programs to a compiler and huge test files to a word processing program.

57. Stress Test: Study the program's response to peak bursts of activity. Word processor's response when a person types 120 words per minute.

58. Security: How easy would it be for an unauthorized user to gain access to this program.

59. Compatibility Test: It checks how one product works with another, efficiently share the same data files simultaneously that resides in the same computer's memory.

60. Portability Test: The program must work on a range of computers. Model will differ in their printers, other peripherals, memory, and internal logic cards.

61. Insatiability and Serviceability Test: An installation utility lets you customize the product to match your system configuration. Does the installation program work? It is easy to use? How long does the average user take to install the product? How long does an expert take?

62. Port Testing: It checks for the completeness of the serial and parallel port functioning.

63. Keyboard Test: Checks for the unnecessary keyboard inputs and the configuration of function keys on Unix, NT and other operating environments.

64. Terminal handling: This is specifically carried in Unix where there are several terminals like vt100, vt220, ansi, ct100 etc. and they need to be configured for data transmission, keyboard keys, screen etc.

65. Documentation test: Checks for the User, system and design documentation version control, configuration management and system identification.

66. Operating System Error Handling: Does the operating system let your program handle the problem or does it halts your program and report a system level error? Insulate the user from bad operating system error handling and other system quirks.

67. Web Server Performance: It tests the server response time and has severe other sub tests within itself.

68. IIS Loading Test: It is dependent upon no of CPUs, RAM, length of ASP request, number of users etc.

69. Skills Tests: This ensures that the capability and maturity of the team is adequate and competent to complete the project successful.

70. Software Configuration Tests: Each and every component of the product and process should have unique identification and tractability, version trol and change control.