CHAPTER

3

FUNDAMENTALS OF SOFTWARE TESTING

OBJECTIVES

This chapter aims to provide a basic knowledge of testing. It clearly highlights the difference between 'Total Quality Management' and 'Big Bang' approaches to testing. It also defines different methodologies used in testing such as 'Black Box Testing', 'White Box Testing' and 'Gray Box Testing'. The chapter concludes with test processes including process of defining test policy, test strategy, and test plan.

3.1 INTRODUCTION

Software development activities during a life cycle have corresponding verification and validation activities at each stage of software development. Software verification involves comparing a work product with processes, standards, and guidelines. Software validation activities are associated with checking the outcome of developed product and the processes used with respect to standards and expectations of a customer. It is considered as a subset of software quality assurance activities though there is a huge difference between quality assurance and quality control. Cost of software verification as well as validation comes under appraisal cost, when one is doing it for the first time. When repeat verification/validation (such as retesting or regression testing) is done, it is defined as cost of failure. Software testing involves verification as well as validation activities such as checking the compliance of the artifacts and activities with respect to defined processes and standards, and executing the software program to ensure that it performs correctly as desired by the customer and expressed in requirement specification agreed between development team and customer. Testing involves finding the difference between actual behaviors with respect to the expected behaviors of an application. There are many stages of software testing as per software development life cycle. It begins with feasibility testing at the start of the project, followed by contract testing and requirements testing, then goes through design testing and coding testing till final acceptance testing, which is performed by customer/user.

E-next THE NEXT LEVEL OF EDUCATION
https://E-next.in
Scanned by CamScanner

3.2 HISTORICAL PERSPECTIVE OF TESTING

The concept of independent testing did not prevail during the initial days of software development. It was believed that whatever the developers were doing was the best way of producing the product and the customer was expected to use it as it is. If there were any problems reported by customer/users, they would be fixed by the developers, and the application would be given again to customer/users. The primary responsibility of testing was with customer/users (and not with developers) during the production phase.

Glenford Myer introduced software testing as a separate phase in software development life cycle. According to him, software testers were expected to test software with all the possible combinations. The main intention was to create a software which would never fail in production. Testers were expected to have an attitude to break the software so that it would be eventually corrected and would never fail during use. This approach separated debugging from testing, and an independent testing community was created. Different phases of software testing evolution as a separate discipline in software development activities were followed in a cycle.

3.2.1 DEBUGGING-ORIENTED TESTING

During the initial phase, software testing was considered as a part of software development. Developers were expected to perform debugging on the application, which they were building. Tests were not documented, and were mainly done in a heuristic way. Generally, testing was completely 'positive testing' to see whether the implementation was working correctly or not.

3.2.2 DEMONSTRATION-ORIENTED TESTING

In this phase, there was an introduction of software testers independent of development activity. Their main aim was to show to customer/users that the software really works. Although, this phase was much advanced than the initial phase of debugging, yet the approach was still positive testing only. The approach was oriented towards demonstration that the software could do something which was expected by the customer/users. Test cases were generally derived from the requirement statements which were oriented towards successful demonstration.

3.2.3 DESTRUCTION-ORIENTED TESTING

This approach was the basis of Glenford Myer's theory of software testing. As per this approach, it was not sufficient to only test the software positively but users must also be protected from any conceivable failure of application. The tester's responsibility changed from 'proving that software works under normal conditions' to 'proving that software does not fail at some abnormal instances'. Often, the testers were too imaginative in breaking the software, and defects for which there were no feasibility of happening were reported as defects. This phase was quiet frustrating to software developers—testers were considered as demons and testing was considered as a hurdle to be passed before delivering the application to the customer.

3.2.4 EVALUATION-ORIENTED TESTING

Evaluation-oriented testing is executed nowadays at many places of software development where the product as well as process of software development is evaluated. It corresponds to the testing process definition where software is evaluated against some fixed parameters derived from quality factors (test factors). Quality factors/test factors are introduced in this phase as a part of customer requirements. It is believed that there

https://E-next.in

Scanned by CamScanner

is no possibility of software without any defect, but some level of defect may be acceptable to the customer. This approach also refers to level of confidence given to customer that application will work as expected by the user. The confidence level is determined and application is evaluated against it. Confidence-level expectations are linked with cost of testing.

3.2.5 PREVENTION-ORIENTED TESTING

Prevention-based testing is done in some highly matured organisations while for many others, the concept is still utopia. Testing is considered as a prevention activity where process problems are used to improve it so that defect-free products can be produced. Every defect found in testing is considered as process lacunae, and efforts are initiated to improve the processes of development. This reduces dependency on testing as a way to improve the quality of software. This helps in reducing the cost by producing right product at the first time.

3.3 DEFINITION OF TESTING

Testing is defined as 'execution of a work product with intent to find a defect'. The primary role of software testing is not to demonstrate the correctness of software product, but to expose hidden defects so that they can be fixed. Testing is done to protect the common users from any failure of system during usage.

This approach is based on the assumption that any amount of testing cannot show that software product is defect free. If there is no defect found during testing, it can only show that the scenario and test cases used for testing did not discover any defect. From user's point of view, it is not sufficient to demonstrate that software is doing what it is supposed to do. This is already done by system architects in system architecture design and testing, and by developers in code reviews and unit testing. Testers are involved mainly to ensure that the system is not doing what it is not supposed to do. Their work includes assurance that the system will not be exposed to any major risks of failure when a normal user is using it. Some people call this approach as negative approach of testing. This negative approach is built upon few assumptions and risks for the software being developed and tested. These assumptions and risks must be documented in the test plan while deciding test strategy or test approach.

3.3.1 WHY TESTING IS NECESSARY?

One may challenge testing activities by asking this question—'If any level of testing cannot declare that there is no defect in the product, then why is it required at all?' In normal life, we find highly qualified and experienced people involved in each stage of development from requirement gathering till acceptance of software. Finding a defect in software is sometimes considered as challenging the capabilities of these people involved in development phases. Testing is necessary due to the following reasons.

- Understanding of customer requirements may differ from person to person. One must challenge the understanding at each stage of development, and there must be some analysis of customer expectations. Approach-related problems may not be found when there is no detail analysis by another person not involved emotionally with development. Everything is considered as 'OK' unless there is an independent view of a system.
- Development people assume that whatever they have developed is as per customer requirements and will always work. But, it is imperative to create real-life scenario and undertake actual execution of a product at each level of software building (including system level) to assess whether it really works or not.

https://E-next.in

Scanned by CamScanner

- Different entities are involved in different phases of software development. Their work may not be matching exactly with each other or with the requirement statements. Gaps between requirements, design, and coding may not be traceable unless testing is performed in relation to requirements.
- Developers may have excellent skills of coding but integration issues can be present when different units do not work together, even though they work independently. One must bring individual units together and make the final product, as some defects may be possible when the sources are developed by people sitting at different places.
- There is a possibility of blindfold and somebody has to work as the devil's representative. Every person feels that what he/she has done is perfect and there is no chance of improvement. Testers have to challenge each assumption and decision taken during development.

3.4 APPROACHES TO TESTING

There are many approaches to software testing defined by the experts in software quality and testing. The approaches may differ significantly as per customer requirements, type of the system being developed as well as management thinking about software development life cycle followed by software, type of project, type of customer, and maturity of development team. These approaches form the part of testing strategy. Few of them are discussed below.

3.4.1 BIG BANG APPROACH OF TESTING

Characteristics of 'Big bang' approach involve testing software system after development work is completed. This is also termed 'system testing' or final testing done before releasing software to the customer for acceptance testing. This testing is the last part of software development as per waterfall methodology. Big bang approach has main thrust on black box testing of software to ensure that the requirements as defined and documented in requirement specifications and design specifications are met successfully. Testing done at the end of development cycle may show the defects pertaining to any phase of development such as requirements, design, and coding. Roughly saying, the phase-wise defect origination follows the trend shown in Table 3.1.

Table 3.1 Phase-wise defect distribution

Development phases	Percentage of defects
Requirements	58
Design	35
Coding	5
Other	2

In case of big bang approach, software is tested before delivery using the executable or final product. It may not be able to detect all defects as all permutations and combinations cannot be tested in system testing due to various constraints like time. In such type of testing, one may find a cascading effect or camouflage effect, and all defects may not be detected. It may discover the failures but cannot find the problems effectively. Sometimes, defects found may not be fixed correctly as analysis and defect fixing can be a problem.

https://E-next.in

Scanned by CamScanner

3.4.2 TOTAL QUALITY MANAGEMENT APPROACH

If there is a process definition for testing software, and these processes are optimised and capable, no (less) defects are produced and no (few) undetected defects are left in the software when it is delivered to the customer. Defect removal costs are approximately 10 times more after coding than before coding. This is a cost associated with fixing the problems belonging to requirements, design, coding, etc. If the defect is not detected earlier but found in acceptance testing or further down the line during warranty, it may be much more costly. Defect removal cost would be 100 times more during production (at user site) than before coding. This may involve deploying people at customer site, loss of goodwill, etc, which is a part of failure cost.

3.4.3 TOTAL QUALITY MANAGEMENT (TQM) AS AGAINST BIG BANG APPROACH

Figure 3.1 is a very famous cost triangle defined by TQM. If the organisation has very good processes defined which are optimised and capable, and can produce consistent results again and again, then it gives advantage in productivity and effectiveness in development. It can reduce cost of production significantly.

Stage 0 There may be a stage of maturity in an organisation where no verification/validation is required to certify the product quality. The cost involved is 'zero' or the benefits derived by investment in process definition, optimisation and deployment make quality free. There is a famous saying 'quality is free'. Theoretically, if the processes are optimised, there is no need of verification/validation as the defects are not produced at all.

E-next
THE NEXT LEVEL OF EDUCATION

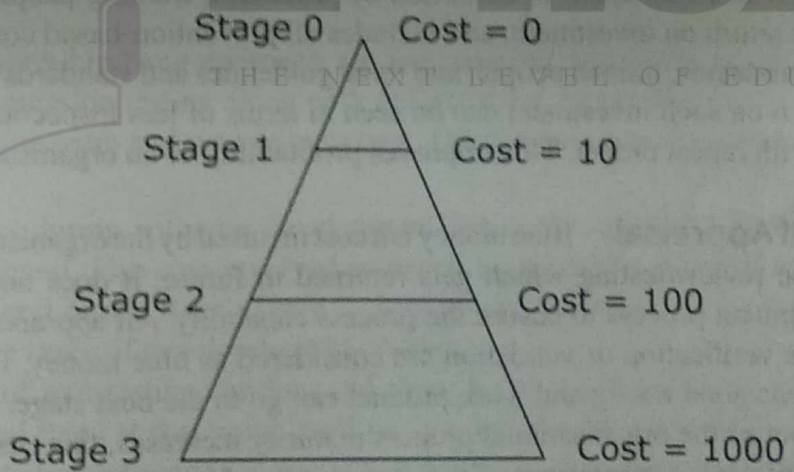

Fig. 3.1 TQM cost triangle

Stage 1 Even if some defects are produced during any stage of development in such quality environment, then an organisation may have very good verification processes which may detect the defects at the earliest possible stage and prevent defect percolation. There will be a small cost of verification and fixing, but stage contamination can be saved which helps in finding and fixing the defects very fast. This is an appraisal cost represented by '10'. This is the cost which reduces the profitability of the organisation due to scrap, rework and reverification.

https://E-next.in
Scanned by CamScanner

Stage 2 If some defects escape the verification process, still there are capable validation processes for filtering the defects before the product goes to a customer. Cost of validation and subsequent defect fixing is much higher than verification. This cost is represented by '100'. One may have to go to the stage where defect was introduced in the product and correct all the stages from that point onward till the defect-detection point. The cost is much higher, but till that point of time the defect has not reached the customer, it may not affect customers feelings or goodwill.

Stage 3 At the bottom of the pyramid, there is the highest cost associated with the defects found by customer during production or acceptance testing. This is represented by '1000' showing that cost paid for fixing such defect is huge. There may be customer complaints, selling under concession, sending people onsite for fixing defects in front of the customer, loss of goodwill, etc. This may result into premature closure of relationship and bad advertisement by the customer.

3.4.4 TQM IN COST PERSPECTIVE

Total quality management (TQM) aims at reducing the cost of development and cost of quality through continual improvement. Often, it is termed 'Quality is free'. It means that the cost of quality must repay much more than what has been invested. TQM defines the cost incurred in development and quality into three parts as follows.

Green Money/Cost of Prevention Green money is considered as an investment by the organisation in doing quality work. It is a cost spent in definition of processes, training people, developing foundation for quality, etc. It gives return on investment, and includes all prevention-based costs. If an organisation has defined and optimised processes, trained people, and fixed guidelines and standards for doing work which are followed, then the return on such investment can be seen in terms of less inspection and testing, and higher customer satisfaction with repeat orders. This improves profitability of an organisation.

THE NEXT LEVEL OF EDUCATION

Blue Money/Cost of Appraisal Blue money is a cost incurred by the organisation during development, in the form of first-time review/testing which gets returned in future. It does not earn profit, but it is an essential part of development process to ensure the process capability. All appraisal techniques used during SDLC such as first-time verification or validation are considered as blue money. First-time testing helps in certifying that nothing has gone wrong and work product can go to the next stage. In initial phases, the cost of appraisal increases, but as the organisational process maturity increases, this cost must go down as fewer samples are required to prove the correctness of the process of making software.

Red Money/Cost of Failure Red money is a pure loss for the organisation. It involves money lost in scrap, rework, sorting, etc. It also represents loss due to various wastes produced during development life cycle, and directly reduces the profit for the organisation and the customer may not pay for it. As the investment or green money increases, failure cost must go down. All cost incurred in reinspection, retesting, and regression testing represent cost of failure.

3.4.5 CHARACTERISTICS OF BIG BANG APPROACH

Big bang approach talks about testing as the last phase of development. All the defects are found in the last phase and cost of rework can be huge.

https://E-next.in

Scanned by CamScanner

- Testing is the last phase of development life cycle when everything is finalised. Heavy costs and efforts of testing are seen at the end of software development life cycle representing 'Big bang' approach. Most of the testing is concentrated in this phase only, as there are no previous verification or validation activities spread during the development phases.
- Big bang approach is characterised by huge rework, retesting, scrap, sorting of software components, and programs after the complete software is built. There may be special teams created to fix defects found in final testing. Big bang works only on correction, and there are no corrective and preventive actions and process improvements arising from these defects. The processes remain immature and every time, defect fixing becomes an invention of the wheel.
- Regression testing reveals many issues, as correction may not be correct and may introduce some defects in the product. There are many interdependencies between various entities while building software product, and these may get affected adversely. When some areas in the software units are touched for correction or fixing of defect, dependent components may get affected in a negative way. These may be termed 'regression defects'.
- All requirements and designs cannot be covered in testing. As the schedule of delivery is fixed and development generally lags behind the schedule, thus huge adhoc, exploratory, monkey, and random testing is done in this part of testing. Testing is done in a hurry, and many iterations of defect fixing and testing are done in the shortest possible time. Some defects may flow to customer as 'known defects' or sometimes they are not declared at all.
- The major part of software build never gets tested as coverage cannot be guaranteed in random testing. Software is generally tested by adhoc methods and intuition of testers. Generally, positive testing is done to prove that software is correct which represents second level of maturity.

Organisations following big bang approach are less matured and pay a huge cost of failure because of several retesting and regression testing along with defect-fixing cycles. Success is completely dependent on good development, testing team and type of customer. The following can be observed.

E-next
THE NEXT LEVEL OF EDUCATION

- Verification activities during software development life cycle can find out about two-third of the total number of defects found. The cost involved in fixing such defects is much less than any other way of finding the defects and fixing them. There is less stage contamination as defects are prevented from progressing from one stage of development to another.
- Validation in terms of unit testing can find out about three-fourth of the remaining defects. Defects are found in the units and fixed at that point itself so that they do not occur further down the line. It reduces the chances of defects being found in system testing. Unit testing validates an individual unit.
- Validation in terms of system testing can find out about 10% of the total number of defects. System testing must be intended to validate system-level requirements along with some aspects of design. Some people term system testing as certification testing while some people term acceptance testing as certification testing. If the exit criteria of system testing (acceptance criteria by customer) are met, system may be released to the customer.

Remaining defects (about 5–10%) go to the customer, unless the organisation makes some deliberate efforts to prevent them from leaking to production phase. Big bang approach may not be useful in preventing defects from going to the customer, as it can find only 5% of the total defects present in the product. In other terms, theoretically, to achieve the effectiveness of life-cycle testing, one may need about 18 cycles of system testing. This can prove to be a costly affair.

https://E-next.in
Scanned by CamScanner

3.5 POPULAR DEFINITIONS OF TESTING

Let us try to define "software testing" keeping the background of Big bang approach in mind. All definitions of testing indicate that testing is a life-cycle operation, and not the activity at the end of development phase. No definition of testing can really cover all aspects of testing. Hence, no definition is complete but indicates a part of what software testing is.

3.5.1 TRADITIONAL DEFINITION OF TESTING

There can be many definitions of testing pertaining to different instances. Few of them are as follows.

- Testing is done to establish confidence that the program does what it is supposed to do. It generally covers the functionalities and features expected in the software under testing. It covers only positive testing.
- Testing is considered as any activity aimed at evaluating an attribute or capability of a program or system with respect to user requirements. To some extent, this definition may be considered as correct, as number of defects found in testing is directly proportional to number of defects remaining in the system.
- Testing is a process of demonstrating that errors are not present in the product. This approach is used in acceptance testing where if the application meets acceptance criteria, then it must be accepted by the customer.
- Testing gives number of defects present which indirectly gives a measurement of software quality. More number of defects indicate bad software and bad processes of development.
- Testing is done to evaluate the program or system used for making software. As we consider that defects are introduced due to incapable processes, testing may be used to measure process capability to some extent.
- Testing is used to confirm that a program performs its intended functions correctly. Intended functionality may be defined from requirement specifications.

If testing is defined as a process, then it is designed to,

- Prove that the program is error free or there is no defect present
- Establish that the software performs its functions correctly and is fit for use
- Establish that all expectations of functionalities are available

Testing may not be any of these certification activities. If the goal of testing is to prove that an application works correctly, then the tester should subconsciously work towards this goal, choosing test data that would prove that the system is working correctly. The reverse would be true if the goal of testing is to locate defects so that eventually these would be corrected. Test data should be selected with an eye towards providing the test cases that are likely to cause product failure.

3.5.2 WHAT IS TESTING?

Let us try to define what is meant by testing with this background. Testing is completely guided by software requirements specifications and design specifications, and supported by test strategy and test approach depending on assumptions and risks of development, testing and usage. Testing process may include the following.

E-next THE NEXT LEVEL OF EDUCATION

https://E-next.in Scanned by CamScanner

- An activity of identification of the differences between expected results and actual results produced, during execution of software application. Difference between these two results suggests that there is a possibility of defect in the process and/or work product. One must note that this may or may not be a defect.
- Process of executing a program with the intention of finding defects. It is expected that these defects may be fixed by the development team during correction, and the root causes of the defects are also found and closed during corrective actions. This can improve development process.
- Detecting specification-related errors and deviations of working application with respect to the specifications. Requirement mismatches and misinterpretation must be detected by testing.
- Establish confidence that a program does what it is supposed to do. This defines the confidence level imparted by software testing to a customer that the software will work under normal conditions. The expectation of confidence level is a function of depth and width of software testing.
- Any activity aimed at evaluating an attribute of a program or software. Acceptance testing is an activity defining whether the software has been accepted or not.
- Measurement of software quality in terms of coverage of testing (in terms of requirements, functionality, features, and number of defects found) can give information about confidence level imparted to a customer
- Process of evaluating processes used in software development. Every failure/defect indicates a process failure. This can be used to improve development processes.
- Verifying that the system satisfies its specified requirements as defined, and is fit for normal use. Requirements may be elicited with the help of the customer.
- Confirming that program performs its intended functions correctly
- Testing is the process of operating a system or component under specified conditions, observing and recording the results of such processing, and evaluating some aspect of system or component on the basis of testing
- Software testing is the process of analysing a software item to detect the difference between existing and required conditions, and to evaluate the feature of the software item.

We have previously discussed about different stakeholders and their interests in software development and testing. Let us try to analyse the expectations or views of different stakeholders about testing.

3.5.3 MANAGER'S VIEW OF SOFTWARE TESTING

The senior management from development organisation and customer organisation have the following views about testing the software product being developed.

- The product must be safe and reliable during use, and must work under normal as well as adverse conditions when it is actually used by the intended users.
- The product must exactly meet the user's requirements. These may include implied as well as defined requirements.
- The processes used for development and testing must be capable of finding defects, and must impart the required confidence to the customer.

3.5.4 TESTER'S VIEW OF SOFTWARE TESTING

Testers have different definitions about software testing, as mentioned below.

- The purpose of testing is to discover defects in the product and the process related to development and testing. This may be used to improve the product and processes used to make it.

https://E-next.in
Scanned by CamScanner

- Testing is a process of trying to discover every conceivable fault or weakness in a work product so that they will be corrected eventually. Random testing sometimes become too imaginative, and unconceivable defects may be found.

3.5.5 CUSTOMER'S VIEW OF SOFTWARE TESTING

Customer is the person or entity who will be receiving/using the product and will be paying for it. Testers are considered as the representatives of the customer in system development.

- Testing must be able to find all possible defects in the software, alongwith related documentation so that these defects can be removed. Customer must be given a product which does not have defects (or has minimum defects).
- Testing must give a confidence that software users are protected from any unreasonable failure of a product. Mean time between failures must be very large so that failures will not occur, or will occur very rarely.
- Testing must ensure that any legal or regulatory requirements are complied during development.

Testing is an activity which is expected to reduce the risk of software's failure in production. All stakeholders have many expectations from testing. Let us try to analyse the meaning of a successful tester.

3.5.6 OBJECTIVES OF TESTING

To satisfy the definition of testing given earlier, testing must accomplish the following things.

- Find a scenario where the product does not do what it is supposed to do. This is deviation from requirement specifications
- Find a scenario where the product does things it is not supposed to do. This includes risk

The first part refers to specifications which were not satisfied by the product while the second part refers to unwanted side effects while using the product.

3.5.7 BASIC PRINCIPLES OF TESTING

The basic principles on which testing is based are given below.

- Define the expected output or result for each test case executed, to understand if expected and actual output matches or not. Mismatches may indicate possible defects. Defects may be in product or test cases or test plan.
- Developers must not test their own programs. No defects would be found in such kind of testing as approach-related defects will be difficult to find. Development teams must not test their own products. Blindfolds cannot be removed in self testing.
- Inspect the results of each test completely and carefully. It would help in root cause analysis and can be used to find weak processes. This will help in building processes rightly and improving their capability.
- Include test cases for invalid or unexpected conditions which are feasible during production. Testers need to protect the users from any unreasonable failure so that one can ensure that the system works properly.
- Test the program to see if it does what it is not supposed to do as well as what it is supposed to do.
- Avoid disposable test cases unless the program itself is disposable. Reusability of test case is important for regression. Test cases must be used repetitively so that they remain applicable. Test data may be changed in different iterations.

https://E-next.in

Scanned by CamScanner

- Do not plan tests assuming that no errors will be found. There must be targeted number of defects for testing. Testing process must be capable of finding the targeted number of defects.
- The probability of locating more errors in any one module is directly proportional to the number of errors already found in that module.

3.5.8 SUCCESSFUL TESTERS

The definition by testers about testing talks about finding defects as the main intention of testing. Testers who find more and more number of defects are considered as successful. This gives some individuality to testing process which talks about ability of a tester to find a defect. There is a difference between executing a test case and finding the defect. This needs an ability to look for detailing, problem areas, and selection of test data accordingly.

- Testers must give confidence about the coverage of requirements and functionalities as defined in test plan.
- Testers must ensure that user risks are identified before deploying the software in production.
- Testers must conduct SWOT analysis of the software and processes used to make it. This can help in strengthening the weaker areas and the processes responsible for defects so that the same problems do not recur.

3.5.9 SUCCESSFUL TEST CASE

Testing is a big investment and justify its existence, if it catches a defect before going to the customer. Every defect caught before delivery means the probability of finding a defect by a customer is reduced. If testing does not catch any defect, it is a failure of testing and a waste for the organisation as well as customer.

Testing involved in software development life cycle starts from requirements, goes through design, coding, and testing till the application is formally accepted by user/customer.

3.6 TESTING DURING DEVELOPMENT LIFE CYCLE

Let us discuss life cycle phase and testing associated with it. This discussion is based on the consideration that development methodology follows waterfall cycle/model.

Requirement Testing Requirement testing involves mock running of future application using the requirement statements to ensure that requirements meet their acceptance criteria. This type of testing is used to evaluate whether all requirements are covered in requirement statement or not.

This type of testing is similar to building use cases from the requirement statement. If the use case can be built without making any assumption about the requirements, by referring to the requirements defined and documented in requirement specification documents, they are considered to be good. The gaps in the requirements may generate queries or assumptions which may possibly lead to risks that the application may not perform correctly. Gaps also indicate something as an implied requirement where the customer may be contacted to get the insight into business processes. It is a responsibility of business analyst to convert (as many as possible), implied requirements to expressed requirements. Target is 100%, though it is difficult to achieve.

Requirement testing differs from verification of requirements. Verification talks about review of the statement containing requirements for using some standards and guidelines, while testing talks about dummy

https://E-next.in
Scanned by CamScanner

execution of requirements to find the consistency between them, i.e., achievement of expected results must be possible by requirements without any assumption. Verification of requirements may talk about the compliance of an output with defined standards or guidelines.

The characteristics of requirements verification or review may include the following.

- Completeness of requirement statement as per organisation standards and formats. It must cover all standards like performance and user interface expected by customer organisation.
- Clarity about what is expected by the users at each step of working while using an application. It must include the expected output by the customer. It may be in the form of error messaging, screen outputs, printer outputs, etc.
- Measurability of expected results, possibly in numerals, so that these results can be tested. Test case will have expected results which must satisfy measurement criteria defined. 'User friendliness' or 'fairly fast' are words which can create confusion about requirements.
- Testability of the scenario defined in requirement statement is must. Some requirements like application must work 24 × 7 for 10 years may not be directly testable.
- Traceability of requirements further down the development life cycle must be ensured. Requirement traceability starts at requirement phase and gets populated as one goes down the life cycle.

Theoretically, each statement in requirement document must give atleast one functional/non-functional test scenario which may result into test cases. Requirements must be prioritised as 'must', 'should be' and 'could be' requirements. The customer is an entity to confirm requirement priority.

Requirement validation must define end-to-end scenario completely so that there is no gap. It must talk about various actors, transactions involved and information transfer from one system to another.

Design Testing Design testing involves testing of high-level design (system architecture) as well as low-level design (detail design). High-level design testing covers mock running of future application with other prerequisites, as if it is being executed by the targeted user in production environment. This testing is similar to developing flow diagrams from the designs, where flow of information is tracked from start to finish. When the flow is complete, the design may be considered as good. Wherever the flow is not defined, or not clear about where it will lead to, there are defects with design which must be corrected. For low-level design, system requirements and technical requirements are mapped with the entities created in design to ensure adequacy of detail design.

Design verification talks about reviewing the design, generally by the experts who may be termed as subject-matter experts. It involves usage of standards, templates, and guidelines defined for creating these designs. Design verification ensures that designs meet their exit criteria.

- Completeness of design, in terms of covering all possible outcomes of processing and handling of various controls as defined by requirements
- Clarity of flow of data within an application and between different applications which are supposed to work together in production environment
- Testability of a design which talks about software structure and structural testing
- Traceability with requirements
- Design must cover all requirements

Code Testing Code files, Tables, Stored procedures etc are written by developers as per guidelines, standards, and detail design specifications. In reality, developers do not implement requirements directly but

https://E-next.in
Scanned by CamScanner

they implement detail design as defined by the designer. Code testing (unit testing) is done by using stubs/drivers as required. Code review is done to ensure that code files written are,

- Readable and maintainable in future. There are adequate comments available.
- Testable in unit testing.
- Traceable with requirements and designs. Anything extra as well as anything missing can be considered as a defect.
- Testable in integration and system testing.
- Optimised to ensure better working of software. Reusability creates a lighter system.

Test Scenario and Test Case Testing Test scenarios are written by testers to address testing needs of a software application. Test cases are derived from test scenarios which are related to requirements and designs. Test scenarios can be functional as well as structural, depending upon the type of requirement and design they are addressing.

- Test scenario should be clear and complete, representing end-to-end relationship of what is going to happen and also, the possible outcomes of such processing.
- Test scenarios should cover all requirements. Test scenarios may be prioritised as per requirement priorities.
- Scenarios should be feasible so that they can be constructed during testing.
- Test cases should cover all scenarios completely.
- Test scenarios and test cases must be prioritised so that in case of less time availability, the major part of the system (where priority is higher) is tested.

E-next
THE NEXT LEVEL OF EDUCATION

3.7 REQUIREMENT TRACEABILITY MATRIX

Some quality management models and standards prescribe complete traceability of a software application from requirements through designs and code files upto test scenario, test data, test cases and test results. Requirement traceability matrix is one way of doing the complete mapping for the software. One can expect a blueprint of an entire application using requirement traceability matrix.

Typical requirement traceability matrix is as shown in Table 3.2.

Table 3.2 Requirement traceability matrix

Require-ments	High-level Design	Low-level design	Code files/ Stored Proce-dures/TBLs	Test scenario	Test cases	Test results

https://E-next.in
Scanned by CamScanner

3.7.1 ADVANTAGES OF REQUIREMENT TRACEABILITY MATRIX

As discussed earlier, requirement traceability matrix is a blueprint of software under development. All the agencies concerned with software can use it to understand the software in a better way. It may answer questions about what is being developed and how it will be implemented. It helps in tracing if any software requirement is not implemented, or if there is a gap between requirements and design further down the line. It also helps to understand if any redundancy has been created in the application.

- Entire software development can be tracked completely through requirement traceability matrix.
- Any test case failure can be tracked through requirements, designs, coding, etc.
- Any changes in requirements can be affected through entire work product upto test cases and vis-à-vis any test case failure can be traced back to requirements.
- The application becomes maintainable as one has complete relationship from requirement till test results available.

3.7.2 PROBLEMS WITH REQUIREMENT TRACEABILITY MATRIX

Theoretically, all softwares must have requirement traceability matrix, but in reality, most of the softwares do not have it. The reasons are numerous; some of the prominent ones are listed below.

- Number of requirements is huge. It is very difficult to create requirement traceability matrix manually. For using some tools, one needs to invest money. Also, people may need to be trained for using tools.
- There may be one-to-many, many-to-one and many-to-many relationships between various elements of traceability matrix, when we are trying to connect columns and rows of traceability matrix, and maintaining these relationships need huge efforts
- Requirements change frequently, and one needs to update the requirement traceability matrix whenever there is a change. Similarly designs, code and test cases may also change which will affect traceability matrix.
- Developing teams may not understand the importance of requirement traceability matrix, if development follows waterfall model, and during maintenance, it may be too late to create it. Incremental and iterative developments are the major challenges for maintaining traceability.
- A customer may not find value in it and may not pay for it

3.7.3 HORIZONTAL TRACEABILITY

When an application can be traced from requirement through design and coding till test scenario and test cases upto test results, it is termed as horizontal traceability. On failure of any test case, we must be able to find which requirements have not been met. Any design which does not have requirement, introduces an extra feature which may be considered as defect. Similarly, when any requirement is not traceable to design, that requirement is not implemented at all. Same thing can happen in the relationship between design and coding, coding and test scenario, and also, test scenario and test case. When any entity can't be traced in forward direction, horizontal traceability is lost.

3.7.4 BIDIRECTIONAL TRACEABILITY

One must be able to go from requirements, designs, coding, and testing to reach the test result. Reverse must also be possible, where one may start from the result and go to requirements. One must be able to go in any

https://E-next.in
Scanned by CamScanner

direction from any point in traceability matrix. This is referred to as 'bidirectional traceability'. CMMi model mandates bidirectional traceability for all products.

3.7.5 VERTICAL TRACEABILITY

Traceability explained above is called 'horizontal traceability' as it goes in horizontal direction, either forward or backward. Traceability may exist in individual column as the requirements may have some interdependencies between them, or these may be child and parent relationships. For achieving a requirement, the other child requirement must be achieved. One requirement may have several child requirements, while some child requirements may have several parent requirements. If these requirements are traced completely, it ensures vertical traceability. Similarly design, coding, and testing may have a vertical traceability where there may be parent-child relationship and interdependence on different parts. Designs may have parent-child relationships, and coding may have 'called functions' and 'calling functions' traceability relationships.

3.7.6 RISK TRACEABILITY

Some application development organisations also add references about the risks of failure faced by the application in Failure Mode Effect Analysis (FMEA). The risks are traced to requirements and mainly with design which defines control mechanism to reduce probability or impact or improves detection ability of a risk. This helps the customer to identify which accident-prone zones are in the application and where the user is completely/partially protected from failures. It also helps in identifying various types of controls that are designed and used. Typical risk traceability matrix is as shown in Table 3.3.

Table 3.3 Risk traceability matrix

Risk	High-level Design/Control	Low-level design/Control	Code files/ Stored procedures/TBLs	Test scenario	Test cases	Test results

3.8 ESSENTIALS OF SOFTWARE TESTING

Software testing is a disciplined approach. It executes software work products and finds defects in it. The intention of software testing is to find all possible failures, so that eventually these are eliminated, and a good product is given to the customer. It intends to find all possible defects and/or identify risks which final user may face in real life while using the software. It works on the principle that no software is defect free, but less risky software is better and more acceptable to users. The tester's job is to find out defects so that they will be eventually fixed by developers before the product goes to a customer. Completion of testing must yield number of defects which can be analysed to find the weaker areas in the process of software development. No amount of testing can show that a product is defect free as nobody can test all permutations and combinations possible in the given software. Software testing is also viewed as an exercise of doing a SWOT analysis of software product where we can build the software on the basis of strengths of the process of development and testing, and overcome weakness in the processes to the maximum extent possible.

https://E-next.in
Scanned by CamScanner

Strengths Some areas of software are very strong, and no (very less) defects are found during testing of such areas. The areas may be in terms of some modules, screens, and algorithms, or processes like requirement definition, designs, coding, and testing. This represents strong processes present in these areas supporting development of a good product. We can always rely on these processes and try to deploy them in other areas.

Weakness The areas of software where requirement compliance is on the verge of failure may represent weak areas. It may not be a failure at that moment, but it may be on the boundary condition of compliance, and if something goes wrong in production environment, it will result into defect or failure of software product. The processes in these areas represent some problems. An organisation needs to analyse such processes and define the root causes of problems leading to these possible failures. It may be attributed to some aspects in the organisation such as training, communication, etc.

Opportunity Some areas of the software which satisfy requirements as defined by the customer, or implied requirements but with enough space available for improving it further. This improvement can lead to customer delight (it must not surprise the customer). These improvements represent ability of the developing organisation to help the customer and give competitive advantage. It decides the capability of the developing organisation to provide expert advice and help to the customer for doing something better.

Threats Threats are the problems or defects with the software which result into failures. They represent the problems associated with some processes in the organisation such as requirement clarity, knowledge base and expertise. An organisation must invest in making these processes stronger. Threats clearly indicate the failure of an application, and eventually may lead to customer dissatisfaction.

3.9 WORKBENCH

Workbench is a term derived from the engineering set-up of mass production. Every workbench has a distinct identity as it takes part in the entire development life cycle. It receives something as an input from previous workbench, and gives output to the next workbench. This can be viewed as a huge conveyor belt where people are working their part while the belt is moving forward. The complete production and testing process is defined as set of interrelated activities where input of one is obtained from the output of previous activity, and output of that activity acts as an input to the next. Each activity represents a workbench. A workbench comprises some procedures defined for doing a work, and some procedures defined to check the outcome of the work done. The work may be anything during software development life cycle such as collecting the requirements, making designs, coding, testing, etc. Organisational process database refers to the methods, procedures, processes, standards, and guidelines to be followed for doing work in the workbench as well as for checking whether the processes are effective and capable of satisfying what customer is looking for in the outcome. There are standards and tools available for doing the work and checking the work in the work bench. While checking/testing a work product in a workbench, if one finds deviations between expected result and actual result, it may be considered as defect, and the work product and the process used for development needs to be reworked.

3.9.1 TESTER'S WORKBENCH

Tester's workbench is made of testing process, standards, guidelines and tools used for conducting tests, and checking whether the test processes applied are effective or not. For every workbench, there should be

E-next
https://E-next.in
Scanned by CamScanner

a definition of entry criteria, process of doing/checking the work, and exit criteria. For testers, there must be a definition of all things that enter the testers workbench. These may be defined in a test plan. Let us discuss with an example of test-case execution as one activity represented by a workbench.

Examples of Tester's Workbench Considering a typical system testing life cycle for a product/project, the different work benches for a tester may be defined as follows. Kindly note that it is not an exhaustive list but a representative one. As one goes into finer details, there may be many more workbenches in each of these defined below.

- Workbench for creating test strategy
- Workbench for creating a test plan
- Work bench for writing test scenario
- Workbench for writing test cases
- Workbench for test execution
- Workbench for defect management
- Workbench for retesting
- Workbench for regression testing

The following is a typical workbench described for system testing execution.

Inputs to Tester's Workbench Inputs may be test scenario, test cases, work products, documentation associated with a work product, test environment, or test plan depending upon the location of workbench in life cycle. The software work product is delivered to the tester as described in delivery note supplied by development team. Delivery note must contain any known issue which tester needs to know before performing testing.

Do Process The software undergoes testing as per defined test case and test procedure. This may be guided by organisational process database defining testing process. 'Do process' must guide the normal tester while doing the process.

Check Process Evaluation of testing process to compare the achievements as defined in test objectives is done by 'check process'. Check process helps in finding whether 'do processes' have worked correctly or not.

Output Output must be available as required in form of test report and test log from the test process. Output of the tester's workbench needs to have an exit criteria definition.

Standards and Tools During testing, the tester may have to use several standards and tools. Standards may include how to install the application, which steps are to be followed while doing testing, how to capture defects, etc. There may be several tools used in testing like defect management tools, configuration management tools, regression testing tools, etc.

Rework If 'check processes' find that 'do processes' are not able to achieve the objectives defined for them, it must follow the route of rework. This is a rework of 'do process' and not of work product under testing. This ensures that all incapable processes are captured so that these can be taken for improvement.

There may be two more criteria in the work bench, viz. suspension criteria for the workbench, and resumption criteria for the workbench guided by organisational policies and standards. Suspension criteria

https://E-next.in

Scanned by CamScanner

defines when the testing process needs to be suspended or halted, whereas resumption criteria defines when it can be restarted after such halt or suspension. If there are some major problems in inputs or standards and tools required by the workbench, 'do/check process' may be suspended. When such problems are resolved, the processes may be restarted.

Testing process may be defined as a process used to verify and validate that the system structurally and functionally behaves correctly as defined by expected result. Components of testing process may include the following.

- Giving inputs (program code) to tester from previous work bench
- Performing work (execute testing) using tools and standards
- Following a process of doing and checking whether test process is capable or not
- Produce output (test results and test log) which may act as an input to the next work bench

Check process must work to ensure that results meet specifications and standards, and also that the test process is followed correctly. If no problem is found in the test process, one can release the output in terms of test results. If problems are found in test process, it may need rework. Figure 3.2 shows a schematic diagram of a workbench.

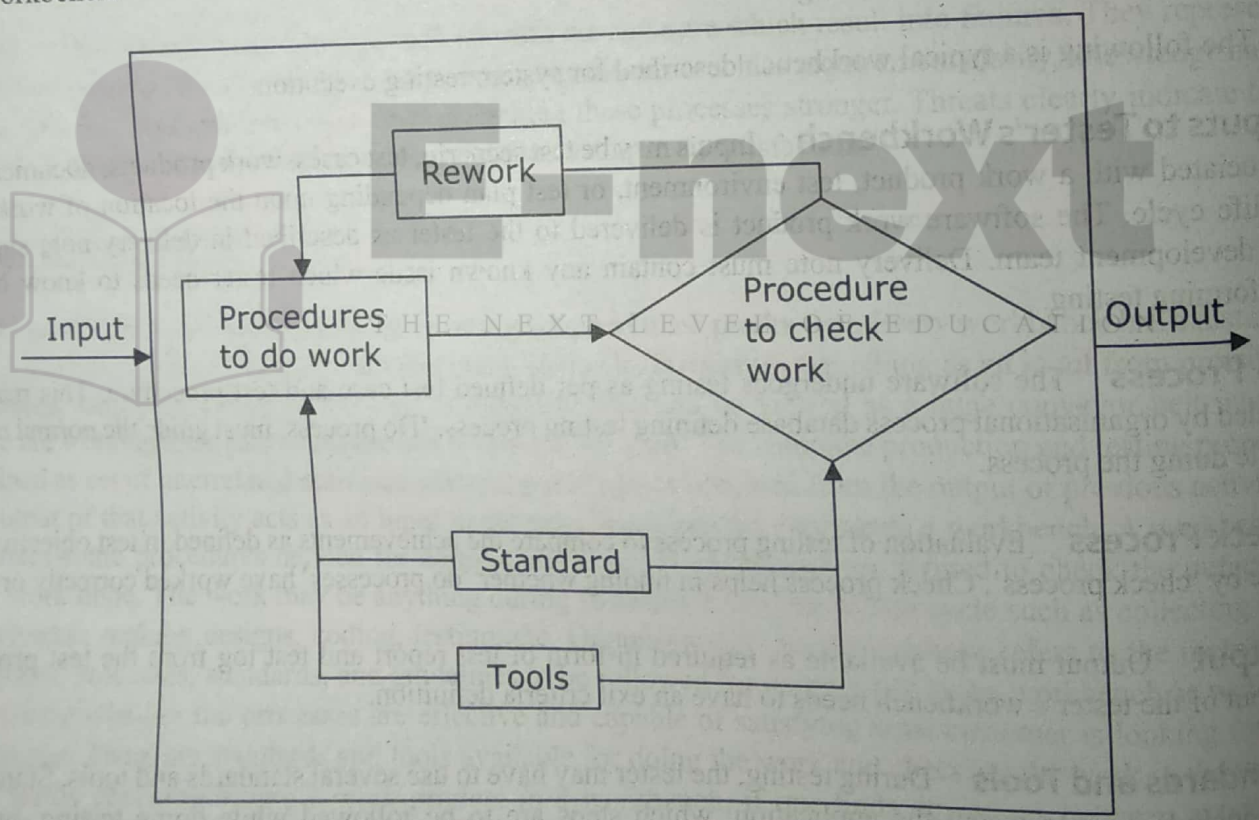

Fig. 3.2 Typical workbench

3.10 IMPORTANT FEATURES OF TESTING PROCESS

Testing is characterised by some special features, as given below.

Testing is a Destructive Process, But it is Constructive Destruction Testing involves a systematic destruction of a product with the intent to find the defects so that these would be fixed before the

https://E-next.in
Scanned by CamScanner

product is given to the customer. While executing tests, a tester goes about testing an application using some valid/invalid inputs to find the response of software to each of these conditions. The ability of the tester to break the application can make him successful. For devising negative testing, one needs to build the scenario with destructive mentality.

Testing Needs a Sadistic Approach with a Consideration That There is a Defect A tester cannot certify that a work product is defect free. He needs to go through the software work product and hunt for defects. If the defect is not found, it directly means that there is some problem with testing process. Defect-free product does not exist. Testers are expected to identify the risks to the final users and test the product accordingly.

If the Test Does Not Detect a Defect Present in the System, it is an Unsuccessful Test Success of testing lies in its ability to find defects or threats and weaker areas of software work product and the processes supporting these areas. Testers who cannot find defects are unsuccessful testers. Testing is considered as an investment only when it reduces the probability that software may fail at customer end.

A Test That Detects a Defect is a Valuable Investment for Development As Well As Customer, it Helps in Improving a Product The root cause analysis of defects can show where the application and process can be and needs to be improved. It also helps in identifying the weaker areas of the processes used for developing software. Weaker areas are analysed to find the process lacunae and take actions on these areas and strengthen them. Thus, defect finding helps in improving processes which can result in increasing customer satisfaction. Testing with an intention to find and fix the problems in processes can be considered as an investment by customer/organisation.

Some organisations use final black box testing as an acceptance testing for the application. If no defect is found in system testing, the software is delivered to the customer. If the sole purpose of testing is to validate specifications implemented then,

- Testing is an unnecessary/unproductive activity as it does not consider invalid scenarios. No amount of testing can certify that there is no defect in the product. There must be a probability of finding the defect in testing.
- Testing is designed to compensate for ineffective software development process, it cannot give results. The defect indicates a process deficiency and it must be fixed by improving the processes. Testing cannot improve the quality of product. Defect is a symptom of something failing and one must try to fix the root cause and not only the symptom.
- If development methodology designed and implemented by a team is not correct, testing cannot compensate for it. Testing is not meant to certify the work products but to find the defects.
- Testing is a separate discipline and may get affected by as well as affects software development processes. Better processes of development and testing can reduce the chances of failure of product at customer place and subsequent customer complaints.

It is Risky to Develop Software and Not to Test it Before Delivery Not providing sufficient resources, time and support for testing activities is a common scenario across the industry. There is a belief that less-tested software means less defects, less rework, less scrap, and less-corrective actions which means higher profits. But this may result into customer dissatisfaction as the defects will get exposed at customer site. Reducing the coverage of testing is another risk associated with software. Software testing must give a desired level of confidence to users that system will not fail. Less testing reduces this confidence level and increases a probability of failure at customer site.

https://E-next.in
Scanned by CamScanner

With High Pressure to Deliver Software As Quickly As Possible, Test Process Must Provide Maximum Value in Shortest Timeframe This approach is adopted in test strategy designing where the efficiency and effectiveness of testing is defined. Generally, test cases are categorised into installation testing, smoke testing, and sanity testing before going into further detailed testing. Test cases which represent scenarios that may occur with higher probability can help in reducing probabilities of failure in production environment. An organisation may define such test cases with probability aspect such as high, medium, and low or allocate the numbers indicating priority of execution. Probability of occurrence of a defect may not have any relationship with severity as severity talks about type of failures.

Testing is no longer an after-programming evaluation to certify that the software works, but supposed to indicate the confidence level that product will work at customer site. Testing can give the SWOT analysis of development process. Thus, testing is adjunct to software development life cycle.

Testing starts at the proposal level and ends only when software application is finally accepted by user/customer. Every stage of development and every work product must go through stages of review and formal approval to ensure that it can go to the next stage. But it is a key to ensure quality at each work product and each phase of software development life cycle.

Highest Payback Comes from Detecting Defect Early in Software Development Life Cycle and Preventing Defect Leakage/Defect Migration from One Phase to Another Defects are the problems or something wrong happening in software as well as the development process used for making software. Every defect indicates failure of the process at some place or another. It is always economical to fix the defects as and when they appear, and conduct an analysis to find the root causes of the defects rather than waiting till it hits the software product and user, again and again. It is always beneficial to do a root cause analysis and fix the problem areas at the earliest possible time. The investment in testing can be worthwhile only if it is capable of finding defects as soon as it is introduced in the work product and also can prevent any potential defect from getting introduced. Defect, if not corrected in the phase where it is introduced, leaks to the next stage and creates a larger problem. This is termed 'phase contamination'. Defect keeps on migrating from one phase to next phase, till it hits back the users at some point of time.

Organisation's Aim Must Be Defect Prevention Rather That Finding and Fixing a Defect The major misconception about testing is that it is considered as a fault-finding mission. Instead, it must be viewed as defect-prevention mission to avoid critical problems in software development process by initiating actions to prevent any recurrence of problems. Finding and fixing the problem is not a good approach as the basic cause of defect is never addressed. It needs analysis of root causes, and defect prevention mechanism—that is installed and operational—to prevent recurrence as well as removal of potential problems.

3.11 MISCONCEPTIONS ABOUT TESTING

At many places, software testing is termed 'quality assurance (QA)' activity. In reality testing is a quality control (QC) activity. There are many other misconceptions about software testing, as listed below.

Anyone Can Do Testing, and No Special Skills Are Required for Testing Many organisations have an approach that anyone can be put in testing. They give the task of testing to developers on the bench or people asking for 'light duty'. Many people involved in testing do not have any experience in testing or in the domain in which testing is done. Test planning, test case writing, and test data definition using

https://E-next.in
Scanned by CamScanner

different methodologies may not be possible with unskilled people. If the organisation considers investment in special skills as a waste of time and money, it may result in disaster for customer/user.

Testers Can Test Quality of Product at the End of Development Process This is a typical approach where system testing or acceptance testing is considered as qualification testing for software. Few test cases out of infinite set of possibilities are used for certifying whether the software application works or not. The customer may be dissatisfied as the application does not perform well as per his expectations. Sometimes, the defects remain hidden for an entire life cycle of software without anybody knowing that there was a defect.

Defects Found in Testing Are Blamed on Developers Another common misconception regarding defects found in testing is blaming developers for defects. Though two-third of defects are due to wrong requirements, yet developers are mostly blamed for defects in software development. Also, some surveys indicate that most of the defects can be attributed to faulty development processes. Developers are responsible for converting the design into code by using the standards or guidelines available. Ensuring good inputs to developer's workbench is a responsibility of the management.

Defects Found By Customer Are Blamed on Tester Testers perform testing by executing few test cases and try to cover some part of software program to check whether the program performs as intended or not. No one can say that testing can ensure 100% coverage. If no defect is found during testing, it does not indicate that the software program is defect free. There may be few defects left in the product which can be found only in real life. One must do a root cause analysis of the defects found, and try to learn from the experiences to ensure that a better product is produced and similar defects do not recur next time. But no one can blame testers for defects reported by customer.

3.12 PRINCIPLES OF SOFTWARE TESTING

Testing needs to be performed according to processes defined for it. It needs skilled and trained people to break the application and demonstrate the problems or defects in the software product. Some key points in software testing are as follows.

Programmers/Team Must Avoid Testing Their Own Work Products Everybody is in love with the work product he/she has made. Also, the approach of an individual remains the same and hence, approach-related defects cannot be found in self review or self testing. A second opinion is essential which can add value to a work product. Though self review is a good tool for retrospection, yet it has many limitations.

Thoroughly Inspect Results of Each Test to Find Potential Improvements Test results show possibilities of weaker areas in the work product and the problems associated with the processes used for developing a work product. The defects found in test log do not form an exclusive list of all problems with the application, but indicate the areas where development team and management must perform a root cause analysis. Corrective actions are to be planned and executed to prevent any possible recurrence of similar defect and make software better. Defects indicate process failures.

Initiate Actions for Correction, Corrective Action and Preventive Actions Defect identification, fixing and initiation of action to prevent further problems are the natural ways of making better

https://E-next.in
Scanned by CamScanner

products and improve processes. Corrections of the defect are done by the developers. But one must ensure that corrective and preventive actions are initiated for making better products again and again.

Establishing that a program does what it is supposed to do, is not even half of the battle and rather easier one than establishing that program does not do what it is not supposed to do. This is negative testing driven by risk assessment for the final users. Roughly, testing may involve 5% positive testing and 95% negative testing.

3.13 SALIENT FEATURES OF GOOD TESTING

Defects indicate the quality of software under testing, development and test processes used for making it. Testing is a life-cycle activity where the testers take part in testing right from proposal stage till the application is finally accepted by the customer/user. Good software testing involves testing of the following.

Capturing User Requirements The requirements defined by the users or customer as well as some implied requirements (which are intended by the users but not put in words) represent the foundation on which software is built. Intended requirements are to be analysed and documented by testers so that they can write the test scenario and test cases for these requirements. User requirements involve technical, economical, legal, operational, and system requirements. Generally, a user is able to specify only functional and non-functional requirements which form a part of operational requirements and legal requirements but definition of other requirements is a responsibility of development organisation.

Capturing User Needs User needs may be different from user requirements specified in software requirement specifications. User needs may include present and future requirements, and other requirements which may include process requirements (including definition of deliverables) and implied requirements. Elicitation of requirements is to be done by the development organisation to understand and interpret the requirements.

Design Objectives Design objectives state why a particular approach has been selected for building software. The selection process indicates the reasons and criteria framework used for development and testing. How an applications functional requirements, user interface requirements, performance requirements and other requirements are interpreted in design, and how they can be achieved in further development must be defined in an approach document.

User Interfaces User interfaces are the ways in which the user interacts with the system. This includes screens and other-ways of communication with the system as well as displays and reports generated by the system. User interfaces should be simple, so that the user can understand what he is supposed to do and what the system is doing. Users ability to interact with the system, receive error messages, and act according to instructions given is defined in the user interfaces.

Internal Structures Internal structures are mainly guided by software designs and guidelines or standards used for designing and development. Internal structures may be defined by development organisation or sometimes defined by customer. It may talk about reusability, nesting, etc. to analyse the software product as per standards or guidelines. It may include commenting standards to be used for better maintenance. Every approach may have some advantages/disadvantages, and one needs to weigh the benefits and costs associated with them to get a better solution.

Execution of Code Testing is execution of a work product to ensure that it works as intended by customer or user, and is prevented from any probable misuse or risk of failure. Execution can only prove that application, module, and program work correctly as defined in requirements and interpreted in design. Negative testing shows that application does not do anything which is detrimental to the usage of a software product.

https://E-next.in

Scanned by CamScanner

3.14 TEST POLICY

Test policy is generally defined by the senior management covering all aspects of testing. It decides the framework of testing and its status in overall mission of achieving customer satisfaction. For project organisations, test policy may be defined by the client while for product organisation, it is decided by the senior management.

3.15 TEST STRATEGY OR TEST APPROACH

Test strategy defines the action part of test policy. It defines the ways and means to achieve the test policy. Generally, there is a single test policy at organisation level for product organisations while test strategy may differ from product to product, customer to customer and time to time. Some of the examples of test strategy may be as follows.

- Definition of coverage like requirement coverage or functional coverage or feature coverage defined for particular product, project and customer.
- Level of testing, starting from requirements and going upto acceptance phases of the product.
- How much testing would be done manually and what can be automated?
- Number of developers to testers.

3.16 TEST PLANNING

Test planning is the first activity of test team. If one does not plan for testing, then he/she is planning for failure. Test plans are intended to plan for testing throughout software development life cycle. Test plans are defined in the framework created by test strategy and established by test policy. Test plans are made for execution which involves various stages of software testing associated with software development life cycle. Test plan should be realistic and talk about the limitations and constraints of testing. It should talk about the risks and assumptions done during testing.

Plan Testing Efforts Adequately with an Assumption That Defects Are There All software products have defects. Test planning should know the number of defects it is intending to find by executing the given test plan. Test plan should cover the number of iterations required for software testing to give adequate confidence required by customer (to show that software will be usable). Defect found in testing is an investment in terms of process improvement opportunity. Test plan is successful if intended number of defects are found.

If Defects Are Not Found, It Is Failure of Testing Activity There are many defects in software. If no (less) defects are found in testing, then it does not mean that there are no (less) defects in the product. It may mean that the test cases are not complete or adequate, or the test data is not effective in locating defects in the software product. Testing is intended to find defects. If defects are not found, the testing process may be considered as defective. Every defect found is an investment as it reduces a probability of any defect which customer may find. If more number of defects are found, it means the development process is problematic.

Successful Tester Is Not One Who Appreciates Development But One Who Finds Defect in the Product Success of testing is in finding a defect and not certifying that application or development process is good. Successful testers can find more defects with higher probabilities of occurrences

https://E-next.in
Scanned by CamScanner

and higher severities of failure. Tester should find defects, which have a probability of affecting common users and thus contribute to a successful application.

Testing Is Not a Formality to Be Completed at the End of Development Cycle Testing is not a certifying process. It is a life-cycle activity and should not be the last part of a development life cycle before giving the application to customer/user. Acceptance testing and system testing are the integral parts of software development where the certification of application is done by the customer but complete test cycle is much more than black box testing.

Software testing includes the following.

- Verification or checking whether a right process is followed or not during development life cycle.
- Validation or checking whether a right product is made or not as per customer's need.

Some differences between verification and validation are shown in Table 3.4.

Table 3.4 Difference between Verification and Validation

Verification	Validation
Verification is an activity where we check the work products with reference to standards, guidelines, and procedures.	Validation is an activity to find whether the software achieves whatever is defined by requirements.
Verification is prevention based. It tries to check the process adherence.	Validation is detection based. It checks the product attributes.
Verification talks about a process, standard and guidelines.	Validation talks about the product.
Verification is also termed 'white box testing' or 'static testing' as the work product undergoes a review.	Validation is also termed 'black box testing' or 'dynamic testing' as work product is executed.
Verification may be based on opinion of reviewer and may change from person to person.	Validation is based on facts and is generally independent of a person.
Verification can find about 60% of the defects.	Validation can find about 30% of the defects.
Verification involves the following. • reviews • walkthroughs • inspection • audits	Validation involves all kinds of testing. • system testing • user interface testing • stress testing
Verification can give the following. • statement coverage • decision coverage • path coverage	Validation can give the following. • requirement coverage • feature coverage • functionality coverage
Some parts of software can undergo verification only, as given below. • coding guidelines • nesting • commenting • declaration of variables	Some characteristics of software can be proven by validation only, as given below. • stress testing • performance testing • volume testing • security testing

https://E-next.in
Scanned by CamScanner

3.17 TESTING PROCESS AND NUMBER OF DEFECTS FOUND IN TESTING

Testing is intended to find more number of defects. Generally, it is believed that there are fixed number of defects in a product and as testing finds more defects, chances of the customer finding the defect will reduce. Actually the scenario is reverse. As we find more and more defects in a product, there is a probability of finding some more defects. This is based on the principle that every application has defects and every test team has some efficiency of finding defects. It is governed by the test team's defect-finding ability. Let us say, the organisational statistics shows that after considerable testing and use of application by a user, number of defects found is three per KLOC; test planning must intend to find three defects per KLOC for the program under testing. The number of defects found after considerable testing will always indicate possibilities of existing number of defects. Figure 3.3 shows a relationship between number of defects found and probability of finding more defects.

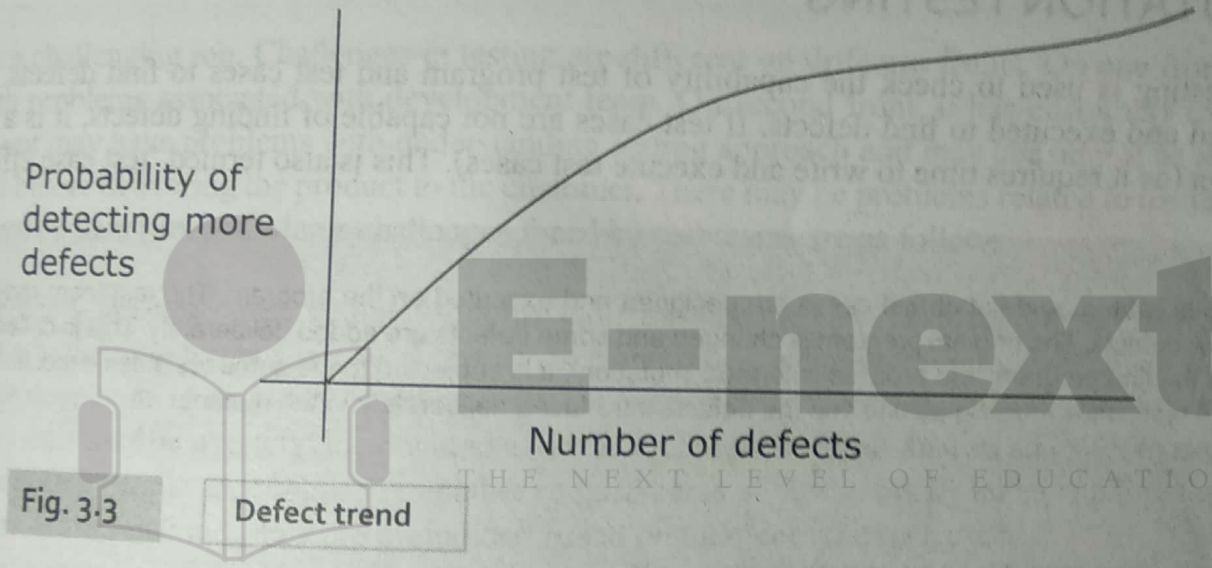

Fig. 3.3 Defect trend

E-next
THE NEXT LEVEL OF EDUCATION

3.18 TEST TEAM EFFICIENCY

Test team efficiency is a very important aspect for development team and management. If test team is very efficient in finding defects, less iterations of testing are required. On the other hand, if development team is less efficient in fixing defects, more iterations of testing and defect fixing may be required.

The test team has some level of efficiency of finding defects. Suppose the application has 100 defects and the test team has an efficiency of 90%, then it will be able to find 90 defects. Thus, if the test team finds 180 defects (considering same efficiency), it means that there are 200 defects in the software product.

Every test manager must be aware of the efficiency of a test team that he/she is working with. Often, test managers and project managers try to assess the test team efficiency at some frequency. The process may be as defined below.

Ideally, test team efficiency must be 100% but in reality, it may not be possible to have test teams with efficiency of 100%. It must be very close to 100% in order to represent a good test team. As it deviates away from 100%, the test team becomes more and more unreliable. Test team efficiency is dependent on organisation culture and may not be improved easily unless organisation makes some deliberate efforts.

https://E-next.in
Scanned by CamScanner

Solved Example 3.1

The development team introduces some defects in a software product and gives it to the test team. The test team completes testing iterations as planned and gives the number of defects found. The development team then analyses the defects found by the test team to understand how many defects have been found by the test team. The ratio gives test team efficiency.

Suppose,

Defects deliberately introduced by development = X
Total defects found by testing team = Y
Defects found by testing team but not belonging to defects deliberately introduced by development = Z

The ratio of $(Y - Z)/X$ will give the test team efficiency.

3.19 MUTATION TESTING

Mutation testing is used to check the capability of test program and test cases to find defects. Test cases are designed and executed to find defects. If test cases are not capable of finding defects, it is a loss for an organisation (as it requires time to write and execute test cases). This is also termed 'test case efficiency'.

Solved Example 3.2

A program is written, and set of test cases are designed and executed on the program. The test team may find out few defects. The original program is changed and some defects are added deliberately. This is called 'mutant of the first program' and process is termed 'mutation'. It is subjected to the same test case execution again. The test cases must be able to find the defects introduced deliberately in the mutant.

Suppose,

Defects deliberately introduced by development = X
Defects found by test cases in original program = Y
Defects found by test cases in mutant = Z

The ratio of $(Z - Y)/X$ will give the test case efficiency. Theoretically, it must be 100%. But it may not be exactly 100% due to the following reasons.

3.19.1 REASONS FOR DEVIATION OF TEST TEAM EFFICIENCY FROM 100% FOR TEST TEAM AS WELL AS MUTATION ANALYSIS

Though desirable, it is very difficult to get a test team with 100% efficiency of finding defects and test cases with 100% efficiency of finding defects. Some of the reasons for deviation are listed below.

- *Camouflage Effect* It may be possible that one defect may camouflage another defect, and the tester may not be able to see that defect, or test case may not be able to locate the hidden defect. It is called 'camouflage effect' or 'compensating defects' as two defects compensate each other. Thus, defect introduced by developer may not be seen by the tester while executing a test case.

https://E-next.in
Scanned by CamScanner

- *Cascading Effect* It may be possible that due to existence of a certain defect, few more defects are introduced or seen by the tester. Though there is no problem in the modification, defects are seen due to cascading effect of one defect. Thus, defects not introduced by developer may be seen by tester while executing a test case.

- *Coverage Effect* It is understood that nobody can test 100%, and there may be few lines of code or few combinations which are not tested at all due to some reasons. If defect is introduced in such a part which is not executed by given set of test cases, then tester may not be able to find the defect.

- *Redundant Code* There may be parts of code, which may not get executed under any condition, as the conditions may be impossible to occur, or some other conditions may take precedence over it. If developer introduces a defect in such parts, testers will not be able to find the defect as that part of code will never get executed.

3.20 CHALLENGES IN TESTING

Testing is a challenging job. Challenges in testing are different on different fronts. On one front, it needs to tackle with problems associated with development team. On second front, it has customers to tackle with. Management may have problems with understanding testing approach and may consider it as an obstacle to be crossed before delivering the product to the customer. There may be problems related to testing process as well as development process. Major challenges faced by test teams are as follows.

- Requirements are not clear, complete, consistent, measurable and testable. These may create some problems in defining test scenario and test cases. Sometimes, a configuration management issue is faced when the development team makes changes in requirements but test team is not aware of these changes.
- Requirements may be wrongly documented and interpreted by business analyst and system analyst. These knowledgeable people are supposed to gather requirements of customers by understanding their business workflow. But sometimes, they are prejudiced based on their earlier experiences.
- Code logic may be difficult to capture. Often, testers are not able to understand the code due to lack of technical knowledge. On the other hand, sometimes, testers do not have access to code files.
- Error handling may be difficult to capture. There are many combinations of errors, and various error messages and controls are required such as detective controls, corrective controls, suggestive controls, and preventive controls.

3.20.1 OTHER CHALLENGES IN TESTING

More bugs found in software introduce additional iterations of fixing defects, retesting, and regression testing of a product. It means more efforts, delayed shipment to customer and late payment receipt by organisation. Often, testers are blamed for delays in delivery.

- Badly written code introduces many defects. Code may not be readable, maintainable, optimisable, and may create problems in future. Defects may not be fixed correctly, and fixing of defects may introduce more defects called 'regression defects'.
- Bad architecture of software cannot implement good requirement statement. What developers do in reality is that they implement the design and not the requirements. Bad architecture creates complex code and adds many defects to the software product.

https://E-next.in
Scanned by CamScanner

- Testing is considered as a negative activity. Often, testers need to reject builds if problems are found in it which does not satisfy exit criteria. It is a difficult situation as organisational fund flow may be depending upon successful delivery of system, and it gets affected due to such rejection.
- Testers find themselves in lose-lose situation in testing. If more defects are found, application delivery is delayed and testers are blamed for such delay. On the other hand, if testing is not done properly, customer complaints are possible and again testers are held responsible for it.

3.21 TEST TEAM APPROACH

Type of the organisation and type of the product being developed define a test team. There may or may not be a separate team doing testing if management does not recognise its importance, or the application under development demands this scenario. There are four approaches of software testing team.

3.21.1 LOCATION OF TEST TEAMS IN AN ORGANISATION

Generally, test team is located in an organisation as per testing policy. It may vary from organisation to organisation, project to project and customer to customer. Following are some of the approaches used for locating test team organisationally.

Independent Test Team Independent test team may not be reporting to development group at all, and are independent of development activities. They may be reporting independently to senior management or customer. Presence of test manager is essential to lead the test team. Such test teams may be shown as in Fig. 3.4.

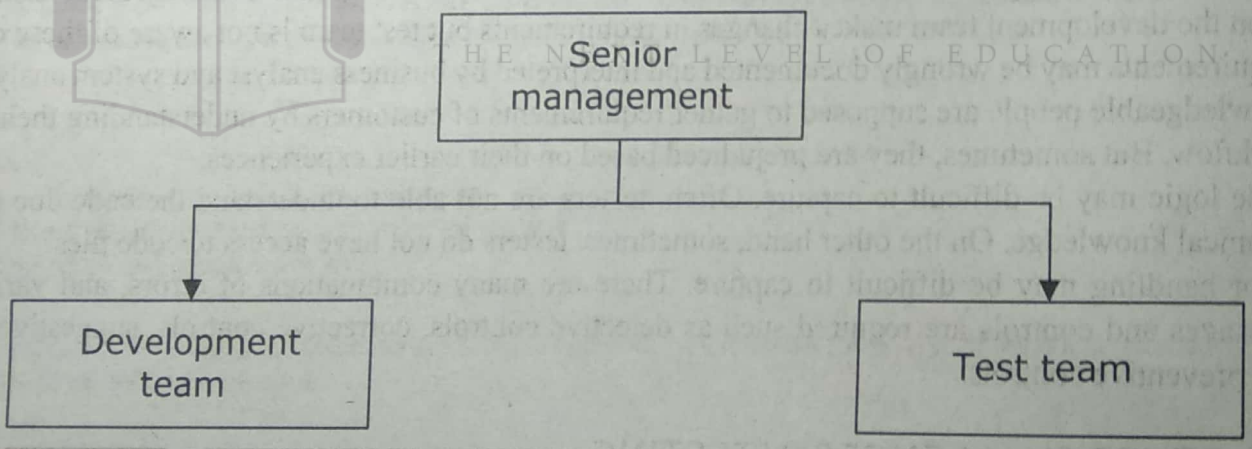

Fig. 3.4 Organisational structure of test team independent of development team

Advantages of Independent Test Team

- The test team is not under delivery pressure. They can take sufficient time to execute complete testing as per definitions of iterations, coverage, etc
- Test team is not under pressure of 'not finding' a defect. They are considered as the certifiers of a product and must be able to find every conceivable fault in the product before delivery.
- Independent view about a product is obtained as thought process of developers and testers may be completely different.

E-next
https://E-next.in
Scanned by CamScanner

- Expert guidance and mentoring required by test team for doing effective testing may be available in the form of a test manager.

Disadvantages of Independent Test Team

- There is always 'us' vs 'them' mentality between development team and test team. Team synergy can be lost as developers take pride in what they develop while testers try to break the system.
- Testers may not get a good understanding of development process as development team tries to hide the process lacunae from them. Testers are treated as outsiders.
- Sometimes, management may be inclined excessively towards development team or test team, and the other team may feel that they have no value in an organisation.

Test Team Reporting to Development Manager If the test team is reporting to development manager, then they can be involved from the start of project till the project is finally closed. Such test team may be as shown in Fig. 3.5.

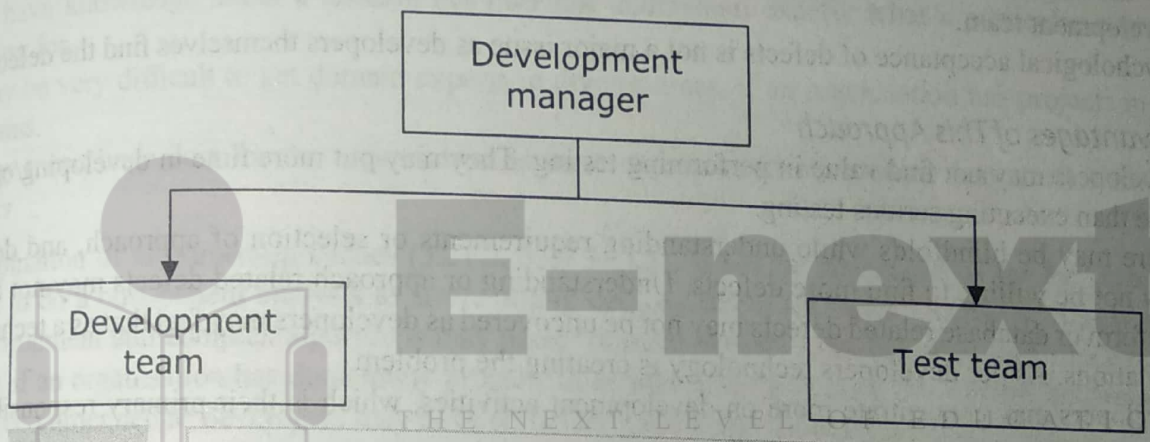

Fig. 3.5 Organisational structure of test team reporting to development manager

Advantages of Test Team Reporting to Development Manager

- There is a better cooperation between development team and test team as both are part of the same team.
- Test team can be involved in development and verification/validation activities from the start of the project. It gives them good understanding of requirements.
- Testers may get a good understanding of development process and can help in process improvement.

Disadvantages of Test Team Reporting to Development Manager

- Expert advice in the form of test manager may not be available to testers. In case testers need some guidance or mentoring, they may have to rely on an external person.
- Sometimes, development managers are more inclined towards development team. Defects found by test team are considered as hurdles in delivery process.
- Often, testers start perceiving the product from developers angle and their defect finding ability reduces.

Matrix Organisation In case of matrix organisation, an effort is made to achieve the advantages of both approaches, and get rid of disadvantages of both approaches. Test team may be reporting functionally to the development manager while administratively it reports to the test manager.

E-next THE NEXT LEVEL
https://E-next.in
Scanned by CamScanner

3.21.2 DEVELOPERS BECOMING TESTERS

Sometimes, those who work as developers in initial stages of development life cycle take the role of testers when the latter stages of life cycle are executed. This is a common practice while working with simple software which does not need very structured testing. Developers becoming testers can be suitable when the application is technologically heavy.

Advantages of This Approach

- Developers do not need another knowledge transfer while working as a tester. The knowledge transfer that they received at initial stages of development can be used by them in both roles.
- Developers have better understanding of detail design and coding, and can test the application easily.
- For automation, some amount of development skill is required in writing the automation scripts. Developers can adapt themselves in automation testing better as they have the ability to create code.
- It is less costly as there is no separate test team. It provides staff balancing to some extent. Initially, the development team is large but as SDLC comes to an end, the test team becomes larger than the development team.
- Psychological acceptance of defects is not a major issue as developers themselves find the defects.

Disadvantages of This Approach

- Developers may not find value in performing testing. They may put more time in developing/optimising code than executing serious testing.
- There may be blindfolds while understanding requirements or selection of approach, and developers may not be willing to find more defects. Understanding or approach related defects may not be found. Platform or database related defects may not be uncovered as developers may feel that as a technological limitations. As per developers technology is creating the problem.
- Developers may concentrate more on development activities, which is their primary responsibility and may neglect testing activities.
- Development needs more of a creation skill while testing needs more of a destruction skill. It is difficult to have a team having both skills at a time.

3.21.3 INDEPENDENT TESTING TEAM

An organisation may create a separate testing team with independent responsibility of testing. The team would have people having sufficient knowledge and ability to test the software.

Advantages of This Approach

- Separate test team is supposed to concentrate more on test planning, test strategies and approach, creating test artifacts, etc.
- There is independent view about the work products derived from requirement statement.
- Special skills required for doing special tests may be available in such independent teams.
- 'Testers working for customer' can be seen in such environment.

Disadvantages of This Approach

- Separate team means additional cost for an organisation.
- Test team needs ramping up and knowledge transfer, similar to a development team.
- An organisation may have to check for rivalries between development team and test team.

https://E-next.in
Scanned by CamScanner

3.21.4 DOMAIN EXPERTS DOING SOFTWARE TESTING

An organisation may employ domain experts for doing testing. Generally, this approach is very successful in system testing and acceptance testing where domain specific testing is required. Domain experts may use their expertise on the subject matter for performing such type of testing.

Advantages of This Approach

- 'Fitness for use' can be tested in this approach where actual user's perspective may be obtained. Domain experts will be testing software from user's perspective.
- Domain experts may provide facilitation to developers about defects and customer expectations, and may be able to interpret requirements in the correct context.
- Domain experts understand the scenario faced by actual users and hence, their testing is realistic.

Disadvantages of This Approach

- Domain experts may have prejudices about the domain which may reflect in testing. Domain experts may have knowledge about a domain but may not understand exactly what a particular customer is looking for.
- It may be very difficult to get domain experts in diverse areas, if an organisation has projects in diverse domains.
- It may mean huge cost for the organisation as these experts cost much more than normal developers/ testers.

Combination of all three approaches (3.21.2, 3.21.3, 3.21.4) can be advantageous for the organisation. One has to do a cost-benefit analysis to arrive at any decision about test team formation. Highly complex user environment and complex algorithms may make 'domain experts doing testing' more effective. On the contrary, if an organisation has done many projects in similar domain in past, 'developers becoming tester' may be recommended as developers may have sufficient knowledge about the subject.

In addition to a test team, there are many other agencies involved in software testing as per phases of software development.

Customer/User Customer or users generally do acceptance testing to declare formal acceptance/rejection/ changes in requirements for the product. Customer perspective is most important in software acceptance. Organisations creating prototype may rely on customer approval of prototype.

Developers Developers do unit testing before the units are integrated. Generally, units require stubs and drivers for testing, and developers can create the same. Sometimes, integration testing is also done by developers if it needs stubs and drivers.

Tester Testers perform module, integration, and system testing as independent testing. They may oversee acceptance testing. Tester's view of system testing is very close to user's view while accepting software.

Information System Management Information system management may do testing related to security and operability of system.They would provide the specialised skills needed for this type of testing.

Senior Management/Auditors Senior management or auditors appointed by senior management such as Software Quality Assurance (SQA) perform predelivery audit, smoke testing, and sample testing to ensure that proper product is delivered to the customer.

https://E-next.in
Scanned by CamScanner

3.22 PROCESS PROBLEMS FACED BY TESTING

'Q' organisations consider that defects in the product are due to incorrect processes. In general, it is believed that incorrect processes cause majority (about 90%) of the working problems. Defects are introduced in software due to incapable processes of development and testing. Software testing is also a process, and prone to introduce defects in the system. If the process of software testing is faulty, it gives problems in terms of defects not found during testing but found by customer, or wrong defects found which are either 'not a defect', 'duplicate', 'cannot be reproduced' or 'out of scope' type of defects. The basic constituents of processes are people, material, machines and methods.

People Many people are involved in software development and testing, such as customer/user specifying requirements; business analysts/system analysts documenting requirements; test managers or test leads defining test plans and test artifacts; and testers defining test scenarios, test cases, and test data. There is a possibility that at few instances some personal attributes and capabilities may create problems in development and testing. Proper skill sets such as domain knowledge and knowledge about development and testing process may not be available.

Material Testers needs requirement documents, development standards and test standards, guidelines, and other material which add to their knowledge about a prospective system. These documents may not be available, or may not be clear and complete. Similarly, other documents which act as a framework for testing such as test plans, project plan, and organisational process documents may be faulty. Test tools and defect tracking tools may not be available. All of these may be responsible for introducing defects in the product.

Machines Testers try to build real-life scenarios using various machines, simulators and environmental factors. These may include computers, hardware, software, and printers. The scenarios may or may not represent real-life conditions. There may be problems induced due to wrong environmental configurations, usage of wrong tool, etc.

Methods Methods for doing test planning, risk analysis, defining test scenarios, test cases, and test data may not be proper. These methods undergo revisions and updations as the organisation matures.

Economics of Testing As one progresses in testing, more and more defects are uncovered, and probability of customer facing a problem reduces while the cost of testing goes up. The cost of customer dissatisfaction is inversely proportional to testing efforts. It means more investment in testing efforts reduces the cost of customer unhappiness. On the other hand, the cost of testing increases exponentially. If the first ten defects are found in one hour, for finding the next ten defects, it may need two hours, and so on. If we plot both curves, then at some point, the two curves intersect each other. This point shows optimum testing point. Area before this point represents an area under testing where defective product goes to customer and customer dissatisfaction cost is higher than cost of testing, while area after this point represents an area of over testing where cost of customer dissatisfaction is less than cost of testing.

Cost of testing curve is guided by the following.

- Defect finding ability of testing team (test team efficiency)
- Defect fixing ability of development team (defect fixing efficiency)
- Defect introduction index of development team which talks about regression defects getting introduced due to fixation of some defects discovered during testing

https://E-next.in
Scanned by CamScanner

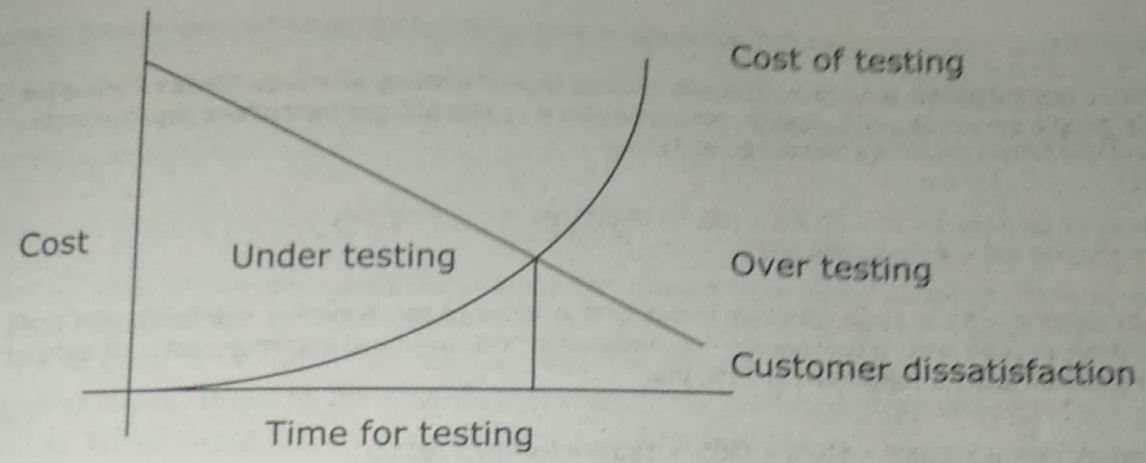

Fig. 3.6 Cost of testing and cost of customer dissatisfaction

- If test team efficiency and defect fixing efficiency are 100%, and defect introduction index is zero, we need only two iterations of testing. As it goes away from ideal numbers, number of iterations increase exponentially.

Cost of customer dissatisfaction is guided by the following aspects.

- Cost of customer dissatisfaction mainly depends upon customer-supplier relationship. If an organisation has done several projects which were very successful, then the customer may not mind if some defects are there in the current project. Customer dissatisfaction curve tends to be parallel to 'Y' axis. On the other hand, if the customer is very finicky about defects and past performance of an organisation is not good, it may tend to be parallel to 'X' axis.
- Cost of customer dissatisfaction also depends on the type of product and its mission criticality to the customer. Customer may not like any problem in high mission-critical software while he may accept certain problems in other types of software.

3.23 COST ASPECT OF TESTING

As seen earlier, cost of quality includes cost of prevention, cost of appraisal and cost of failure. Testing may take some portion of each of these costs. It is believed that cost of quality is about 50% of the cost of product. Testing is a costly affair and an organisation must try to reduce the cost of testing to the maximum extent possible.

There is a famous concept of efforts conversion into cost in case of software development, as effort is the major component of total cost. This may be done by standard costing method or marginal costing method as per organisation's process definition. Efforts spent by the organisation in developing and testing of an application are converted at some predefined rates to arrive at the total cost of a product. Sometimes, the cost of resource varies as per the role played by a person. For example, a project manager may get more rate than a developer, or an architect may get more billing than a tester.

https://E-next.in
Scanned by CamScanner

Solved Example 3.3

Let us suppose that the project duration is 10 months with 22 days of working per month, 8 hours working per day and 100 people are working on it. Also, if conversion rate is say Rs 500 per person hour, then the cost of development and testing taken together will be as follows.

Total efforts spent on project = 10 × 22 × 8 × 100 = 176,000 hrs
Total cost = 176,000 × 500 = Rs 88,000,000

An organisation may have some additions to this cost in terms of contingencies, overheads and profit expected to arrive at sales price. If contingency is considered at 10%, overhead apportionment is considered at 10% and expected profit is considered at 20%, then

Sales price would be = 8,800,000 × 110% × 110% × 120% = Rs 127,776,000

It is a very rare scenario that a project will have 100 people working for all 10 months for the development project. Generally, development projects never have the same number of resources throughout the life cycle. Initially, it may need less number of people and as one passes through different phases of development, number of resources required increases exponentially. Once the peak activities are over, number of resources required goes down.

The same cycle is followed by testing resource requirements. As the testing phases progress, number of resources required increases exponentially. Once the peak activities are over, number of test resources goes down. Thus, for development project, costing is always dynamic.

In case of maintenance or production support type of work, number of resources remains fairly constant over a long-time horizon. Figure 3.7 indicates resources required for a development project.

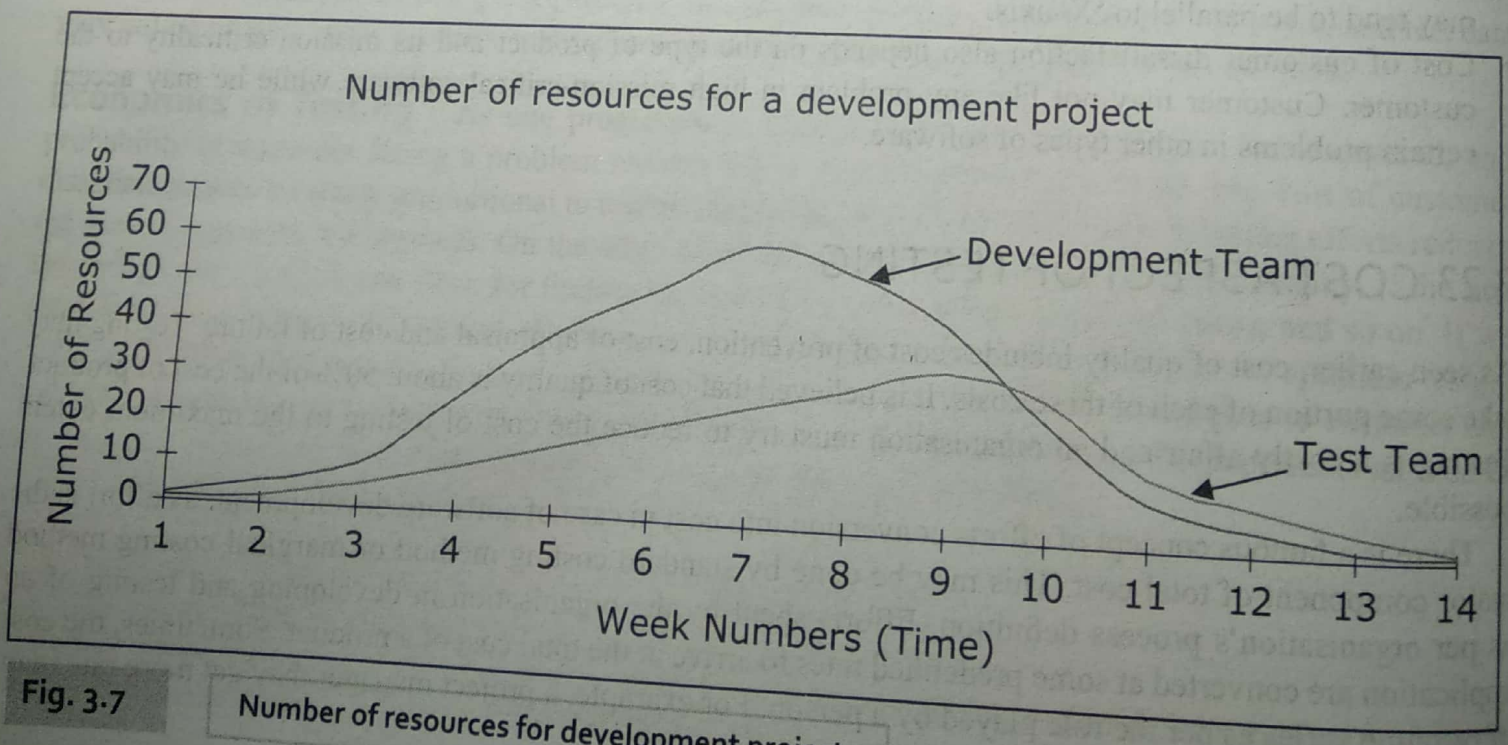

Fig. 3.7 Number of resources for development project

https://E-next.in
Scanned by CamScanner

Cost of development manufacturing includes the cost spent in the following.

- Capturing the requirements, conducting analysis, asking queries, and elicitation of requirements
- Cost spent in designing the application including high-level designing and low-level designing
- Cost spent in writing code, integrating and creating the final product

3.23.1 ASSESSMENT OF COST OF TESTING

Cost of testing may be considered as the cost of the project activities on and above normal phases of development. Various phases of development are associated with phases of verification and validation. Cost of testing is a function of two processes, viz. development process maturity and test process maturity. It has also a very close relationship with the type of application being produced, methodology of development and testing, domain, technical parameters of development, and testing.

Type of Application and Cost of Testing Depth and breadth of testing has direct relationship with the size and importance of an application to a user, and test efforts that the customer is ready to pay for also varies accordingly. As far as size of an application is concerned, testing follows 'Rayleigh Putnam' curve. As the size of application increases, the testing efforts increase exponentially. An organisation must evaluate its own curve on the basis of historical data, or may adapt to the existing baselines in initial phases when historical data is not available.

$$\text{Cost of testing} = k\, f(x)$$

where 'k' is a constant and $f(x)$ is a function of size of an application while 'x' indicates the size of the application.

An application may be defined as per its importance to the user. If user is completely dependant upon the application, he may want more confidence and thus, invest more in testing. On the other hand, if software does not have business criticality, the customer may not be ready to pay that much cost.

Development and Test Methodology Development and test methodologies also affect testing efforts and costs. It is generally believed that waterfall development methodology can produce robust software and testing efforts required are much lesser. Agile methodologies may need huge testing because there is more stress on regression testing, as each iteration of development is over. Iterative development has more testing costs as there is a change in requirements, design, coding, etc.

Test methodologies have an impact on test efforts and costs. An organisation conducting phase-wise testing has lesser cost than the organisation facing testing at the last phase of development. Life cycle testing is very cheap in comparison to testing at the last phase of development. Usage of regression testing tools may increase the cost of testing in initial phases while in the long term, overall cost of testing is reduced due to automation.

Domain and Technological Aspects Application domain plays an important role, along with criticality of the application, to the users. If the application is going to affect human life, there may be many regulations coming into the picture. The customer will be more cautious about testing and test results. There may be more legal implications if sufficient and adequate testing is not done. Some other types of software affecting huge sum of money are also regulated by the rules, regulations and laws of the place where it is used.

It is believed that technology also plays an important role in deciding the extent of testing. Object-oriented development faces different challenges in testing compared to normal development without any object usage.

https://E-next.in
Scanned by CamScanner

There may be more thrust on integration testing rather than unit testing. Old development languages and databases sometimes make testing difficult.

Maturity of Development and Testing Processes Development and testing process maturity is an important issue in deciding the cost of testing. Theoretically, testing is not at all required if development can achieve the defined output. If development process is highly matured and can achieve the expected output, then testing may be considered as wastage. Many engineering organisations have reached the phase of 'zero' defect and 'zero' inspection. But, software application development may not have achieved a level of maturity to produce defect-free products. Some amount of testing is required to find all conceivable faults so that development team can fix them.

Many organisations have a development phase followed by one complete iteration of testing. It must find all defects so that these are eventually fixed in rework phase. Second iteration of testing must achieve the required level of confidence that application will not fail.

Testing involves all three components of cost of quality mentioned below.

3.23.2 COST OF PREVENTION IN TESTING

Cost of prevention is the cost incurred in preventing defects from entering into a system. In various phases of verification and validation, cost of prevention is distributed. Some of these are discussed below.

Cost Spent in Creation of Verification and Validation Artifacts This part of cost is spent when the project team and test team create various artifacts related to verification and validation at different phases of software development. Cost of planning includes creating test plans and quality plans so that quality can be appraised correctly. This phase may also include the time and effort spent in creation of guidelines for testing, reviews, creation of checklists, writing of test cases and test data along with test scenarios. In case of test automation, cost of test-script creation may be a part of prevention cost.

Cost Spent in Training Testers may need training on the domain under testing. They may also need training on test process and various tools used for testing. This may include organisation-level training as well as project-specific training.

Cost of prevention is calculated as follows.

- Cost spent in creating various plans related to software verification and validation. An organisation must have a baseline definition of the effort required to create plans.
- Cost spent in writing test scenarios, test cases, creating guidelines for verification and validation, and creating checklists for doing verification and validation activities. Organisation baselines must define the time and effort required for this activity.
- Training requirements may be separately assessed depending upon the competency of test teams, type of application, and level of tool usage for testing. There may not be baseline data available in this case as it may change from project to project.

3.23.3 COST OF APPRAISAL IN TESTING

Cost of appraisal includes cost spent in actually doing verification and validation activities. Generally, first-time verification/validation is considered under cost of appraisal. Test artifacts also need reviews and testing to confirm that they are correct. Checklists needed for the reviews must also be reviewed before they are used.

https://E-next.in
Scanned by CamScanner

Irrespective of whether time and efforts are spent by developers or testers, all efforts spent on conducting reviews, walkthroughs, inspection, unit testing, integration testing, and system testing are considered under cost of appraisal.

Cost of appraisal is calculated as follows.

- Cost required for conducting first-time verification and validation activities can be assessed from the size of an application and organisation baseline data available.
- Time and effort required to conduct the given number of test cases may be derived from the productivity numbers, and number of test cases required can be derived from the size of an application.

3.23.4 COST OF FAILURE IN TESTING

Cost of failure in testing accounts for all retesting, regression testing, and rereviews conducted as the defects are found in earlier iterations. Any cost spent on and above first-time verification and validation makes a cost of failure for testing. The organisation must have a target to reduce the cost of failure continuously, as this directly affects its profitability.

Cost of failure is calculated as follows.

- On the basis of historical data available in baseline studies, one may plan number of iterations required to achieve predetermined exit criteria.
- Number of test cases per iteration and number of iterations required can give the efforts required to perform retesting and regression testing.

3.24 ESTABLISHING TESTING POLICY

Good testing is a deliberate planned effort by the organisation. It does not happen on its own, but detailed planning is required. Testing efforts need to be driven by test policy, test strategy or approach, test planning, etc. Test policy is an intent of test management about how an organisation perceives testing and customer satisfaction. It should define test objectives and test deliverables.

Test strategy or approach must define what steps are required for performing an effective testing. How the test environment will be created, what tools will be used for testing, defect capturing, defect reporting, and number of test cycles required will be a part of test strategy. It must talk about the depth and breadth of testing to ensure adequate confidence levels for users.

Test objectives define what testing will be targeting to achieve. It is better to have test objectives expressed in numbers in place of qualitative definitions. Some of the test objectives may be about code coverage, scenario coverage, and requirement coverage, whereas others may define the targeted number of defects.

Testing must be planned and implemented as per plan. Test plan should contain test objectives and methods applied for defining test scenario, test cases, and test data. It should also explain how the results will be declared and how retesting will be done. It is a general expectation that automation can solve all problems regarding testing process. One thing to be noted is that—automation can increase the speed and repeatability of testing but test planning cannot be done automatically. Rather, it needs involvement of people doing testing.

3.25 METHODS

Generally, methods applied for testing efforts are defined at organisational levels. They are generic in nature and hence, need customisation. They are customised into a test plan, and any tailoring required

https://E-next.in
Scanned by CamScanner

to suite a specific project may be done. Management directives establish methods applied for testing. It includes what part will be tested/not tested, and how it will be tested. Which tools will be used for testing, defect-tracking mechanism, communication methods and if there is any decision of automation, how it will be undertaken—all this must be covered by these processes. Management directives are defined in test strategy.

Testing strategy may be discussed with users/customer to get their views/buy-in about testing. It may be accomplished through meetings and memorandums. User/customer must be made aware of cost of finding and fixing defects. All stakeholders for the project must be made aware that 'zero defects' is an impossible condition and acceptance criteria for the project must be defined well in advance (possibly at the time of contract). Methods of using data or inputs provided by a customer must be analysed for sufficiency and correctness.

3.26 STRUCTURED APPROACH TO TESTING

Testing that is concentrated to a single phase at the end of development cycle, just before deployment, is costly. Testing is a life cycle activity and must be a part of entire software development life cycle. If testing is done only in the last phase before delivery to customer, the results obtained may not be accurate and defect fixing may be very costly. Here, it cannot show development process problems and similar defects can be found again and again. Four components of wastes involved in this type of testing are given below.

Waste in Wrong Development Wrong specifications used for development or testing will result into a wrong product and wrong testing. Even if specifications are correct, it may be wrongly interpreted in design, code, documentation, etc. The defects not found during reviews or white box testing will be discovered only at the customer's end. This may lead to high customer dissatisfaction, huge rework, retesting, etc.

Waste in Testing to Detect Defects If testing is intended to find all defects in product, then cost of testing will be very high. Effective reviews can reduce the cost of software testing and development. If entire responsibility for software quality is left to black box testing or final system testing, then the cost of testing and cost of customer dissatisfaction may be very high.

Wastage as Wrong Specifications, Designs, Codes and Documents Must Be Replaced By Correct Specifications, Designs, Codes and Documents The cost of fixing defects may be very high in the last part of testing as there are more number of phases between defect introduction phase and defect detection phase, and defect may percolate through the development phases. Correcting the specifications, designs, codes and documents, and respective retesting/regression testing is a costly process. It can lead to schedule variance, effort variance and customer dissatisfaction. One defect fix can introduce another defect in the system.

Wastage as System Must Be Retested to Ensure That the Corrections Are Correct For every fixing of defect, there is a possibility of some other part of software getting affected in a negative manner. One needs to test software again to ensure that fixing of software has been correct, and it has not affected the other parts in a negative manner. Regression and retesting are essential when defects are found and fixed.

https://E-next.in
Scanned by CamScanner

3.27 CATEGORIES OF DEFECT

Software defects may be categorised under different criteria. The categories of defects must be defined in the test plan. The definition may differ from organisation to organisation, project to project and customer to customer.

3.27.1 ON BASIS OF REQUIREMENT/DESIGN SPECIFICATION

- Variance from product specifications as documented in requirement specifications or design specifications represents specification related defects. These defects are responsible for 'Producer's gap'.
- Variance from user/customer expectations as business analyst/system analyst is not able to identify customer needs correctly. These variances may be in the form of implied requirements. These are responsible for 'Users gap'.

3.27.2 TYPES OF DEFECTS

- Wrongly implemented specifications are relate to the specifications as understood by developers differing significantly from what the customer wants. These may be termed 'misinterpretation of specifications'.
- Missing specifications are the specifications that are present in requirement statements but not available in the final product. The requirements are missed, as there is no requirement tracing through product development.
- Features not supported by specifications but present in the product represent something extra. This is something added by developers though these features are not supported by specifications.

3.27.3 ROOT CAUSES OF DEFECTS

- Wrong requirements given by user/customer can be a basic cause of defect. This is due to the inability of the customer to put the requirements in words, or specifying requirements which are not required.
- Business analyst/system analyst interprets customer needs wrongly can be another major cause of defect. This is due to the inability of business analyst/system analyst to elicit requirements.
- System design architect does not understand requirements correctly and hence, the architecture is wrong. This may be due to communication gap or inability of the architect in understanding the requirements.
- Incorrect program specifications, guidelines, and standards are used by respective people. If the organisation processes are not capable, then defects are introduced in the product so produced.
- Errors in coding represent lack of developer's skills in understanding design and implementing it correctly.
- Data entry error caused by the users while using a product. This can be possible when users are not protected adequately. This indicates design problems.
- Errors in testing—false call/failure to detect an existing defect in the product. The first part introduces defects in a correct product while the second part allows defects to go to the customer.
- Mistake in error correction, where defect is introduced while correcting some identified defect.

3.28 DEFECT, ERROR, OR MISTAKE IN SOFTWARE

The problems with software work product may be put under different categories on the basis of who has found it and when it has been found (as shown in Table 3.5).

COLLEGE OF COMME LIBRARY

https://E-next.in

Scanned by CamScanner

Table 3.5 Comparison of mistake, error and defect

Mistake	Error	Defect
An issue identified while reviewing own documents, or peer review may be termed 'mistake'.	An issue identified internally or in unit testing may be termed 'error'.	An issue identified in black box testing or by customer is termed 'defect'.
Very low cost of finding mistakes and can be fixed immediately.	Slightly more cost of finding an error and needs some time for fixing.	Most costly and needs longer time for fixing defects.
Most of the time, problems and resolutions are not documented properly.	Sometimes, problems and resolutions are documented, but may not be used for process improvements.	Problems and resolutions are officially documented and used for process improvements.

3.29 DEVELOPING TEST STRATEGY

Test planning includes developing a strategy about how the test team will perform testing. Some key components of testing strategy are as follows.

- Test factors required in particular phase of development
- Test phase corresponding to development phase

Process of developing test strategy goes through the following stages.

Select and Rank Test Factors for the Given Application The test team must identify critical success factors/quality factors/test factors for the software product under testing. A software may have some specific requirements from user's point of view. Test factors must be analysed and prioritised or ranked. Some test factors may be related to each other (either direct or inverse relationship). The trade-off decisions may be taken after consulting with customer, if possible, when the relationship is inversed.

Identify System Development Phases and Related Test Factors The critical success factors may have varying importance as per development life cycle phases. One needs to consider the importance of these factors as per the life cycle phase, that one is going through. The test approach will change accordingly.

Identify Associated Risks with Each Selected Test Factor In Case if it is Not Achieved Trade-offs may lead to few risks of development and testing the software. Customer must be involved in doing trade-offs of test factors and the possible risks of not selecting proper test factor. The risks with probability and impact need to be used to arrive at the decision of trade-off.

Identify Phase in Which Risks of Not Meeting a Test Factor Need to Be Addressed The risks may be tackled in different ways during development life cycle phases. As the phase is over, one needs to assess the actual impact of the risks and effectiveness of devised countermeasures for the same.

3.30 DEVELOPING TESTING METHODOLOGIES (TEST PLAN)

Developing test tactics is the job of project-level test manager/test lead. Different projects may need different tactics as per type of product/customer. Designing and defining of test methodology may take the following route.

https://E-next.in
Scanned by CamScanner

3.30.1 ACQUIRE AND STUDY TEST STRATEGY AS DEFINED EARLIER

Test strategy is developed by a test team familiar with business risks associated with software usage. Testing must address the critical success factors for the project and the risk involved in not achieving it.

3.30.2 DETERMINE THE TYPE OF DEVELOPMENT PROJECT BEING EXECUTED

Development projects may be categorised differently by different organisation, as shown below.

- Traditionally developed software by following known methods of development such as waterfall development life cycle. An organisation has a history available, and the lessons learned in previous projects can be used to avoid contingencies in the given project.
- Commercially Off The Shelf (COTS) purchased software which may be integrated in the given software. Requirements and designs of such software may not be available, and integration in terms of parameters passing can be a major constraint.
- Maintenance activities such as bug fixing, enhancement, porting, and reengineering will have their own challenges, such as availability of design documents, compatibilities of various technologies, and code readability.
- Agile methodology of development has small iterations of development and heavy regression testing.
- Iterative method of development has continuously changing requirements and all other artifacts must be updated accordingly.
- Spiral development, where new things are added in system again and again. Generally followed methodology in spiral development is modular design, development and testing.

Another possible way of categorising software is on the basis of its criticality. It can be as follows.

- Life affecting software
- Huge money affecting software
- Software which cannot be tested in real life and requires simulators for testing
- Other software

E-next
THE NEXT LEVEL OF EDUCATION

3.30.3 DETERMINE THE TYPE OF SOFTWARE SYSTEM BEING MADE

Type of software system defines how data processing will be performed by the software. It may involve the following.

- Determine the project scope (whether it is multivendor or multisite development). Distributed development and integration of parts developed by different vendors can be a challenging task.
- New developments including scope of development and scope of testing, when the products are enhanced from existing levels.
- Changes to existing system such as bug fixing, enhancement, reengineering, and porting.

3.30.4 IDENTIFY TACTICAL RISKS RELATED TO DEVELOPMENT

Risks may be introduced in a software due to its nature, type of customer, type of developing organisation, development methodologies used, and skills of teams. Risk may differ from project to project.

- *Structural Risks (Refers to Methods Used to Build a Product)* If the projects are supposed to use existing libraries or designs which may need complex algorithms to be written, the structure of the software

https://E-next.in
Scanned by CamScanner

may pose the highest risk. Complex structures with interfaces in relation to many other systems can make the architecture fragile.

- *Technical Risks (Refers to Technology Used to Build and Operate the System)* If the organisation is new to a particular technology, or the technology itself is new to the world, then it can lead to this kind of risk. Unproven technology or inability to work with the latest technology are the problems faced by developers as well as users.

- *Size Risks (Refers to Size of All Aspects of Software)* As the software size increases, it becomes more complex. Maintaining integrity of a very big software itself is a challenge.

3.30.5 DETERMINE WHEN TESTING MUST OCCUR DURING LIFE CYCLE

- Testing phases starting from proposal, contract or requirement testing till acceptance testing, and their integration decide the test strategy for the project.
- Previously collected information, if available, is to be used to decide how much testing is to be done, at what time and in which phases. It defines the cost of testing, cost of customer dissatisfaction and any trade-offs.
- Build tactical test plan which will be used by the test team in execution of testing related activities
- To describe software being tested, test objectives, risks, business functions to be tested, and any specific tests to be performed.

3.30.6 STEPS TO DEVELOP CUSTOMISED TEST STRATEGY

- Select and rank quality factors/test factors as expected by the customer in the final product. Quality factors must be prioritised. Generally, the scale used is 1–99, where '1' indicates higher priority while '99' indicates lower priority. No two quality factors have the same ratings.
- Identify the system development phase where these factors must be controlled. As the defects may originate in different phases, quality factors may change their priority during each phase of development life cycle.
- Identify business risks associated with system under development. If quality factors are not met, these may induce some risk in a product.
- Place risks in a matrix so that one may be able to analyse them. This may be used to devise preventive, corrective and detective measures to control risks.

Table 3.6 shows how test strategy matrix can be developed

Table 3.6 Typical test strategy matrix

Quality factors	Test phases	Require-ment	Design	Coding	Testing	Deploy-ment	Mainte-nance
Factors	Compliance Accuracy Reliability	Risks associated at different phases for different factors.					

E-next
https://E-next.in
Scanned by CamScanner

3.30.7 TYPE OF DEVELOPMENT METHODOLOGY IMPACT TEST PLAN DECISIONS

Table 3.7 shows, in general, which test tactics can be used depending upon the type of development activity. This is an indicative list and may differ from situation to situation, product to product and customer to customer.

Table 3.7 Development model and testing tactics

Type	Characteristics	Test tactics
Traditional system development such as waterfall model of development	• Uses a system development methodology as defined • User knows requirements completely and these are defined • Development determines structure of application	• Test at end of each task/step/phase to ensure that exit criteria is achieved • Verify that specifications match user needs exactly • Test functions and structures as per test plan
Iterative development/prototyping development	• Complete set of requirements unknown to users and developers • Structure predefined by development • Refactoring may be done, if required	• Verify that case tools are used properly where required by testing • Test functionality first • Test other areas of product such as integration, system etc
System maintenance methodology	Modify structure if it represents a limiting factor for maintenance activities	• Test structure consistency as it may affect entire application • Works best with release methods of maintenance • Requires heavy regression testing as system is updated
Purchase/contracted software COTS	• System unknown to testers • May contain defects in hidden form • Functionality defined in user manual • Documentation may vary from software to software	• Verify that functionality matches needs of business • Test functionality as per business need • Test fitness for use into production environment

3.31 TESTING PROCESS

Testing is a process made of many milestones. Testers need to achieve them, one by one, to achieve the final goal of testing. Each milestone forms a basis on which the next stage is built. The milestones may vary from organisation to organisation and project to project. Following are few milestones commonly used by many organisations.

Defining Test Policy Test policies are defined by senior management of the organisation or test management, and may or may not form a part of the test plan. Test policy at organisation level defines the intent of test management. Test policy is dependent on the maturity of an organisation in development and test process, customer type, type of software, development methodology, etc. Test policy may be tailored at project level, depending upon the scope and historical information available from similar projects.

https://E-next.in

Scanned by CamScanner

Defining Test Strategy Definition of test strategy includes how the test team will be organised, and how it will work to achieve test objectives, decision about coverage, automation, etc. Test strategy provides the actions to the intents defined by test policy. Test strategy helps the test team to understand the approach of testing.

Preparing Test Plan Test planning is done by test managers, test leads or senior testers, as the case may be. Test policy sets a tone, whereas test strategy adds the actions required to complete test policy. Test plan tries to answer six basic questions—What, When, Where, Why, Which and How. Test plans are for individual product/project and customer. They are derived from test strategy and give details of execution of testing activity.

Establishing Testing Objectives to Be Achieved Test objectives measure the effectiveness and efficiency of a testing process. They also define test achievements that they plan for. Test objectives are also defined from the quality objectives for the project or product, and the critical success factors for testing. Testing objectives must be 'SMART' (specific, measurable, agreed upon, realistic, and time bound).

Designing Test Scenarios and Test Cases How the test scenarios and test cases will be defined should be explained by the test strategy. A test scenario represents user scenario which acts as a framework for defining test cases. It may have actors and transactions. Similarly, there may be few scenarios arising from system requirements. Theoretically, each transaction in a test scenario eventually becomes a test case.

Writing/Reviewing Test Cases Writing and reviewing test cases along with test scenarios and updating requirement traceability matrix accordingly are the tasks done by senior testers or test leads for the project. The traceability matrix gets completed by adding the test cases and finally, the test results. One more column in the traceability matrix would be the results of execution of test cases, i.e., test results.

Defining Test Data Test data may be defined on the basis of different techniques available for the purpose. It may include boundary value analysis, error guessing, equivalence partitioning, and state transition. Test data must include valid as well as invalid set of data. It must include some special values generated from error guessing. Test data definitions may be important from testing point of view as different iterations must have different test data sets though it may be used by same test cases.

Creation of Test Bed Testing needs creation of environment for testing. It may be a real-life environment or a simulated environment using some simulators. Test bed defines some of the assumptions in a test plan which may induce certain risks of testing. It must reflect real-life situations as closely as possible. Simulators may need definition of few risks, as testing is not done in real environment.

Executing Test Cases Execution of actual test cases with the test data defined for testing the software involves applying test cases as well as test data and trying to get the actual results. It sometimes involves updating test cases or test data, if some mistakes are found in initially defined test cases or test data.

Test Result Logging results of testing in test log is the last part of the testing iteration. There may be several iterations of testing planned and executed. The defect database may be populated, if expected results are not matching with the actual results. One needs to make sure that the expected results are traceable to requirements.

Test Result Analysis Testing is a process of SWOT analysis of software under development and testing. Examining test results and analysis may lead to interpretation of software in terms of capabilities and weakness. At the end of testing, the test team must recommend the next step after testing is completed

https://E-next.in
Scanned by CamScanner

to the project manager—whether the software is ready to go to the next stage or it needs further rework and retesting.

Performing Retesting/Regression Testing When Defects Are Resolved By Development Team When defects are given to a development team, they perform analysis and fix the defects. Retesting is done to find out whether the defects declared as fixed and verified by the development team are really fixed or not. Regression testing is done to confirm that the changed part has not affected (in a negative way) any other parts of software, which were working earlier.

Root Cause Analysis and Corrective/Preventive Actions Root cause analysis is required to initiate corrective actions. Development/test team performs post-mortem reviews to understand the weakness of development as well as test process, and initiates actions to improve them. Process improvement is the last activity after the project is closed and formally accepted by the customer. All defect prevention activities must lead to process improvements.

3.32 ATTITUDE TOWARDS TESTING (COMMON PEOPLE ISSUES)

Attitude of development team and senior management or project management towards a test team is a very important aspect to build morale of the test team. It may be initiated from test policy and may be percolated down to test strategy definition and test planning. Some of the views about test team are as follows.

- New members of development team are not accustomed to view testing as a discovery process where defects are found in the product. The defects found are taken as personal blames rather than system/process lacunae. Sometimes, people try to defend themselves, considering it as an attack on the individual.
- 'We take pride on what we developed' or 'we wish to prove that it is right' or 'it is not my fault' are very common responses. Developers may not accept the defect in the first place. If they accept the presence of a defect, then they try to put blame on somebody else. Root cause analysis is very difficult in such situations as people may attach personal ego to it.
- Conflict between developer and tester can create differences between project teams and test teams. In reality, the sole aim of development and testing must be a customer satisfaction, and defects must be considered as something which prevents achievement of this objective.

3.33 TEST METHODOLOGIES/APPROACHES

The two major disciplines in testing are given below.

Black Box Testing Black box testing is an attesting methodology where product is tested as per software specifications or requirement statement defined by business analysts/system analysts/customer. Black box testing mainly talks about the requirement specification given by customer, or intended requirements as perceived by testers. It deals with testing of an executable and is independent of platform, database, etc. This testing is with the view as if a user is testing the system.

White Box Testing White box testing is a testing methodology where software is tested for the structures. White box testing covers verification of work products as per structure, architecture, coding standards and guidelines of software. It mainly deals with the structure and design of the software product.

https://E-next.in
Scanned by CamScanner

White box testing requires that testers must have knowledge about development processes and artifacts including various platforms, databases, etc.

There is one more methodology covering both testing methodologies at the same time.

Gray Box Testing Gray box testing talks about a combination of both approaches, viz. black box testing and white box testing at the same time. There may be various shades of black box testing as well as white box testing in this type of testing, depending upon the requirements of product. Though not a rule, gray box testing mainly concentrates on integration testing part along with unit testing.

Broadly, all other testing techniques may be put in any of these three categories, viz. black box testing, white box testing and gray box testing.

3.33.1 BLACK BOX TESTING (DOMAIN TESTING/SPECIFICATION TESTING)

Black box testing involves testing system/components considering inputs, outputs and general functionalities as defined in requirement specifications. It does not consider any internal processing by the system. Black box testing is independent of platform, database, and system to make sure that the system works as per requirements defined as well as implied ones. Actual system (production environment) is simulated, if it is difficult to create a real-life scenario in test laboratory. It does not make any assumption about technicalities of development process, platform, tools, etc. It represents user scenario and actual user interactions.

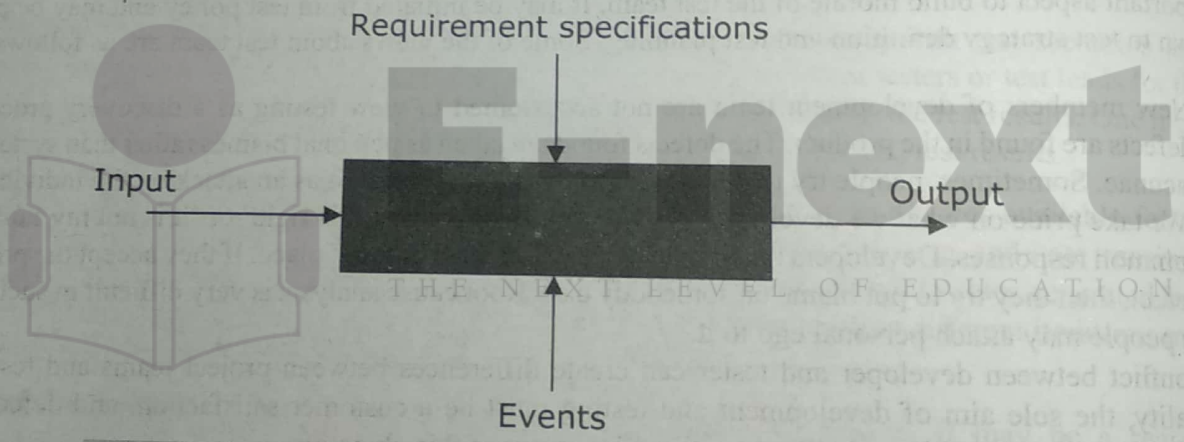

Fig. 3.8 Black box testing

Figure 3.8 shows a block box testing schematically. Black box functional testing is generally conducted for integration testing, system testing, and acceptance testing where the users/customers or the testers as representatives of the customer execute a system as if it is used by users in production environment.

Advantages of 'Black Box Testing' Black box testing is used primarily to test the behavior of an application with respect to expressed or implied requirements of the customer.

- Black box testing is the only method to prove that software does what it is supposed to do and it does not do something which can cause a problem to user/customer.
- It is the only method to show that software is living and it really works.
- Some types of testing can be done only by black box testing methodologies, for example, performance and security.

https://E-next.in
Scanned by CamScanner

Disadvantages of 'Black Box Testing' Black box testing has many disadvantages so far as software development methodology is concerned.

- Some logical errors in coding can be missed in black box testing as black box testing efforts are driven by requirements and not by the design. It uses boundary value analysis, equivalence partitioning, and some internal structure problems can be missed.
- Some redundant testing is possible as requirements may execute the same branch of code again and again. If an application calls common functions again and again, then it will be tested so many times that it leads to redundant testing.

Test Case Designing Methodologies Black box testing methodology defines how the user is going to interact with the system without any assumption about how the system is built. As there is no view of how software is build, defining test cases is very difficult. Test cases may be defined using the user scenario called 'test scenario'. Completeness of test scenario is essential for good testing. Theoretically, each sentence in the test scenario may become a test case. Scenario contains activities in terms of transactions and actors.

Test Data Definition Black box testing is mainly driven by the test data used during testing. It may not be feasible to test all possible data which user may be using while working with an application. Some special techniques are applied for defining test data which can give adequate coverage, and also limits the number of test cases and the risk of failure of an application during use. Some of these techniques are mentioned below.

- Equivalence partitioning
- Boundary value analysis
- Cause and effect graph
- State transition testing
- Use case based testing
- Error guessing

E-next
THE NEXT LEVEL OF EDUCATION

3.33.2 WHITE BOX TESTING

White box testing is done on the basis of internal structures of software as defined by requirements, designs, coding standards, and guidelines. It starts with reviews of requirements, designs, and codes. White box testing can ensure that relationship between the requirements, designs, and codes can be interpreted. White box testing is mainly a verification technique where one can ensure that software is built correctly. Figure 3.9 shows a white box testing schematically.

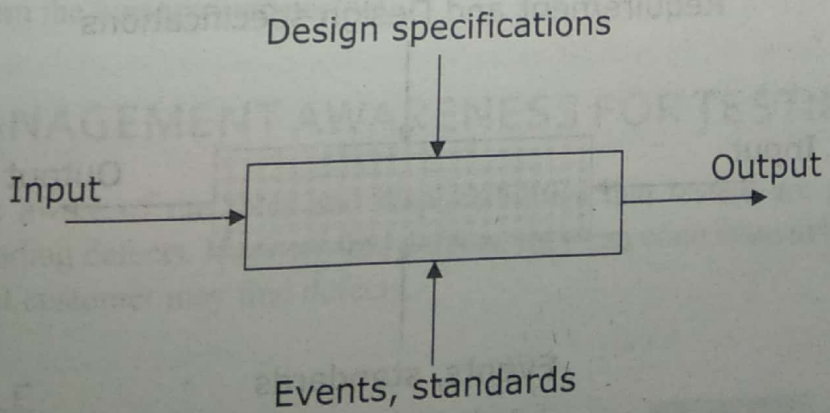

Fig. 3.9 White box testing

https://E-next.in
Scanned by CamScanner

Advantages of 'White Box Testing' White box testing is a primary method of verification.

- Only white box testing can ensure that defined processes, procedures, and methods of development have really been followed during software testing. It can check whether the coding standards, commenting and reuse have been followed or not.
- White box testing or verification can give early warnings, if something is not done properly. It is the most cost effective way of finding defects as it helps in reducing stage contamination.
- Some characteristics of software work product can be verified only. There is no chance of validating them. For example, code complexity, commenting styles, and reuse.

Disadvantages of 'White Box Testing' White box testing being a verification technique has few shortcomings.

- It does not ensure that user requirements are met correctly. There is no execution of code, and one does not know whether it will really work or not.
- It does not establish whether decisions, conditions, paths, and statements covered during reviews are sufficient or not for the given set of requirements.
- Sometimes, white box testing is dominated by the usage of checklists. Some defects in checklists may reflect directly in the work product. One must do a thorough analysis of all defects.

Test Case Designing Test case designing is based on how test artifacts are created and used during testing. It defines how documents are written and interpreted by each person involved in software development life cycle. Some of the techniques used for white box testing are as follows.

- Statement coverage
- Decision coverage
- Condition coverage
- Path coverage
- Logic coverage

E-next
THE NEXT LEVEL OF EDUCATION

3.33.3 GRAY BOX TESTING

Gray box testing is done on the basis of internal structures of software as defined by requirements, desi coding standards, and guidelines as well as the functional and non-functional requirement specificati Gray box testing combines verification techniques with validation techniques where one can ensure software is build correctly, and also works. Figure 3.10 shows a gray box testing schematically.

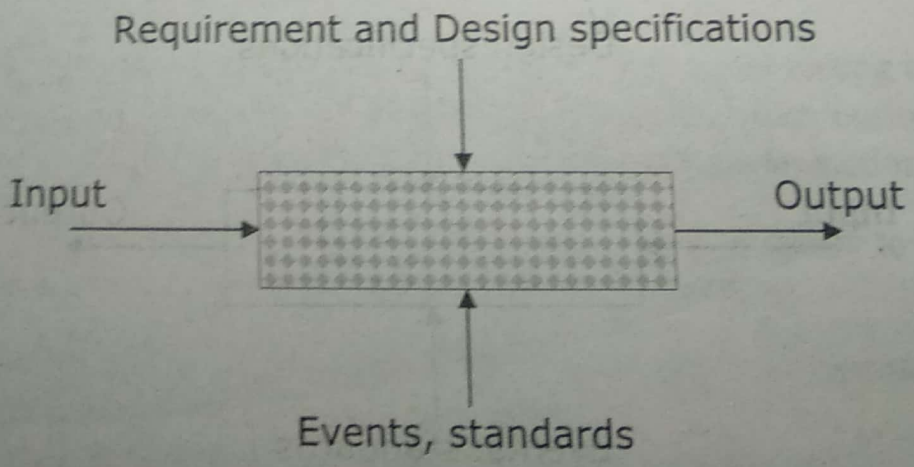

Fig. 3.10 Gray box testing

https://E-next.in
Scanned by CamScanner

Advantages of 'Gray Box Testing'

- Gray box testing tries to combine the advantages of white box testing and Black box testing. It checks whether the work product works in a correct manner, both functionally as well as structurally.

Disadvantages of 'Gray Box Testing'

- Generally, gray box testing is conducted with some automation tools. Knowledge of such tools along with their configuration is essential for performing gray box testing.

3.34 PEOPLE CHALLENGES IN SOFTWARE TESTING

Testing is a process and must be improved continuously. People need to analyse and take actions on the shortcomings found in the process, so that they can be improved continuously. Few expectations of software process improvement needs from testers are given below.

- The tester is responsible for improving testing process to ensure better products with less number of defects going to customer, thus enhancing customer satisfaction. All defects must be found and the confidence level must be built in the process that can give customer satisfaction. Proper coverage as required by test plan must be achieved.
- Testing needs trained and skilled people who can deliver products with minimum defects to the stakeholders. Testers have to improve their skills through continuous learning.
- The tester needs a positive team attitude for creative destruction of software. Defect in the software is an opportunity to improve the product and not to blame developers. Testers must be able to pinpoint the lacunae in software development process as the defects are found.
- Testing is creative work and a challenging task. Feasible test scenarios and test cases, as well as effective ways of looking for defects are essential to improve testing effectiveness.
- Programmers and testers work together to improve the quality of software developed and delivered to customer, and the process used for software development and testing. The ultimate aim is customer satisfaction.
- Testers hunt for defects—they pursue defects not people, including developers. Every defect is considered as a process shortcoming. Defect closure needs retesting and regression testing to find whether the defect is really fixed or not, and to ensure that there is no negative impact of a defect on existing functions.
- Testing needs patience, fairness, ambition, creditability, capability, and diligence on part of testers. Every defect must be seen from the business perspective.

3.35 RAISING MANAGEMENT AWARENESS FOR TESTING

The management must be aware of the roles and responsibilities that testers are performing to achieve customer satisfaction by finding defects. If testers find defects, they can contribute in building good software by reducing probability that customer may find defects.

3.35.1 TESTER'S ROLE

While establishing a test function in an organisation, the management has some objectives to be achieved. Test team needs to understand these objectives and fulfill them.

https://E-next.in

Scanned by CamScanner

- Calculate testing cost, effectiveness of testing and ensure that management understands the same. By doing good testing, number of customer complaints must reduce and cost of failure must go down.
- Demonstrate cost reduction and increase in effectiveness over a time span (as rework and scrap reduce over a long horizon). This can be shown by reduced customer complaints as well as less rework.
- Highlight needs and benefits of training—in test team as well as development team—on testing activities and skills, so that testers can perform better. Many developers need information about unit testing, integration testing and their role in such testing.
- Collect and distribute information on testing to all team members as well as development team/organisation which can be used for improvement.
- Get involved in test budgeting. Testing needs people, money, time, training and other resources. The organisation may have to develop budget to procure all these aspects.

3.36 SKILLS REQUIRED BY TESTER

Testing needs a disciplined approach. A tester is the person entrusted by an organisation to work as the devils agent. He/she is a person working for the client, finding the obvious defects in the processes and products. The main purpose of testing is to demonstrate that defects are present, and point towards the weaker areas in the software as well as processes used to build it, so that actions can be initiated in that direction. It must build confidence in management and customer that the application with which they will be working is usable and does not have defects. One must try to build maximum possible skills, and training is one of the effective methods to build a good testing team.

3.36.1 GENERAL SKILLS

Written and Verbal Presentation Skill Presenting test results, or discussing about an application or defects involves communication with many people. Testers are supposed to present test results and tell development team, customer and management about the present status of application and where further improvements can be done. Testers must be good in presentation skills.

Effective Listening Skill Testers need to listen to the customers voice as well as views of developers. Listening to customer as well as developer— to understand the needs and requirements correctly—is required to ensure that the scenarios and test cases can be written in a proper way. Tester's listening skills can convert testing into effective testing. Listening skills can give them complete information about the process, software application and also what the customer and management are looking for.

Facilitation Skill Facilitation of development team as well as customer is done by testers, so that defects are taken in the proper spirit. Testers must be able to tell the exact nature of a defect, how it is happening and how it will affect the users. Testers must contribute to improve development process and take part in building a better product.

Software Development, Operations and Maintenance Good knowledge of software development life cycle and software testing life cycle help testers in designing test scenarios and test cases accordingly. The defect age and cost of testing are important parameters to be controlled by testers.

Continuous Education Testers must undergo continuous education and training to build and enforce quality practices in development processes. They need to undergo training for test planning, test case definition, test data definition, methods and processes applied for testing, and reporting defects.

E-next
https://E-next.in
Scanned by CamScanner

3.36.2 TESTING SKILLS

Concepts of Testing A tester must have complete knowledge about testing as a discipline. He/she must understand methods, processes, and concepts of testing. He/she must be capable of doing test planning, designing test scenario, writing test cases, and defining test strategy, and defining test data.

Levels of Testing Testing is a multitier activity where the application goes from one level to another after successful completion of the previous level. Testers are involved in each phase of software development right from proposal and contract, followed by requirement till acceptance testing. Testers must ensure that each phase is passed successfully.

Techniques for Validation and Verification Techniques of verification/validation must be understood and facilitated by testers to the development team, customer and management. While writing test cases, he/she needs to define test case pass/fail criteria to validate the test case and product.

Selection and Use of Testing Tools Testing involves use of various tools including automation tools, defect tracking tools, configuration management tools, and simulators. A tester must understand and use the tools effectively.

Knowledge of Testing Standards Testing standards are defined by software quality management. There can be some international standards or organisation/customer defined standards. Testers need to understand these standards, and apply them effectively so that a common understanding can be achieved.

Risk Assessment and Management Testing is risk-driven activity. Test cases and test data must be defined to minimise risks to the final users in production environment. Testing efforts must be managed to improve their effectiveness and efficiency. Managing testing involves planning, organising, directing, coordinating, and controlling testing process.

Developing Test Plan Test plan development is generally done by test managers or test leads while implementation of these plans is done by testers. Individual testers must plan for their part in overall test plan for the project.

Defining Acceptance Criteria Definition of acceptance criteria is an important milestone for testing. Generally, acceptance criteria are defined by customer well before the project starts. Testers need to define acceptance criteria for the phases and iterations of testing. There are various forms of acceptance criteria which will be discussed later. The phase-end acceptance criteria may be defined by the testers in test plan.

Checking of Testing Processes Testers follow the processes as defined in the test plan. They need to audit the testing processes to check the compliance and effectiveness, and also initiate actions if deviations are observed. Testers must contribute in testing process improvements.

Execution of Test Plan Testers are given the responsibility of executing a test plan. It includes defining test scenario, test cases, and test data, and their execution. They put test results in test log and defects in defect logging tool. Resolved defects are taken for retesting. They must perform regression testing when

E-next
THE NEXT LEVEL OF EDUCATION
https://E-next.in
Scanned by CamScanner

planned. Testers must do analysis of test results to define weaker and stronger areas of software development and testing process. They must be able to define test coverage such as code coverage, statement coverage, branch coverage, requirement coverage, and function coverage.

Continuous Improvement of Testing Process Testers must plan for continuous improvement of testing process. Testing process must be subjected to improvements followed by phase of consolidations. Actions must be planned for improving process, and the results must be compared with expectations. If some deviations are observed, new actions can be initiated.

Testing tips

Customer pays for a product on the basis of the value he finds in acquiring such product. Cost of development and cost of testing define the profit available to an organisation by selling such product/project. Market forces define the sales price of the product/project. There is always a pressure on development and testing to reduce the cost to improve profitability. The following list may be considered as a generic guideline for reducing cost of testing or improving the value of a product.

Reduce Software Development Risk Development activities may introduce several risks starting from requirement capturing, through design and development, coding and so on. Software testing must be effective to locate the defects as early as possible so that stage contamination can be reduced.

Perform Testing Effectively Testing must be able to capture defects as effectively and efficiently as possible. It must give adequate confidence to users that application will not meet any accidental failures. It must be able to find as many defects as possible, so that they will be fixed eventually and probability of failure at customer place is minimised.

Uncover Maximum Number of Defects Each defect uncovered must reduce a chance of customer complaint. If testing can find 100% defects present in the given software, it will not be possible for customer to see any failure during use. Though it is very difficult, one must try to find all conceivable defects. Successful tester is one who finds maximum number of defects.

Use Business Logic Testers must have a good knowledge about the domain under testing. This is true for system testers where it is expected that they would be working as normal users. They must use business logic to improve efficiency and effectiveness of testing, and also testing must give enough confidence to users that there will not be any accidental failures.

Testing Must Occur Throughout SDLC Defect found as early as possible can reduce cost of failure by preventing stage contamination. Testing concentrating more on system testing may not be very effective as software developed may be very fragile.

Testing Must Cover Functional/Structural Parts Often, people consider functional testing as a complete testing. One must keep in mind that there are five types of requirements mentioned by 'TELOS' (Technical Economic Legal Operational System). Only operational requirements may cover functional as well as non-functional requirements. It must also cover the way software is built to give better screen designs, optimum performance, and security.

https://E-next.in

Scanned by CamScanner

Summary

This chapter establishes the basics of software testing. It starts with a historical perspective of testing and then explains how testing evolved from mere debugging to defect prevention technique. It then discusses the benefits of independent testing. 'TQM' concept of testing, 'Big Bang' approach of testing, and benefits of 'TQM' testing are elucidated in detail.

The chapter also presents the definitions of a successful tester and the basic principles of software testing. It offers a detailed exposition of the process of creating test policy, test strategy, and test plan. It also gives an indepth understanding of 'Black Box Testing', 'White Box Testing' and 'Gray Box Testing'. The chapter concludes with skills required by a good tester and challenges faced by a tester.

E-next

1) Explain the evolution of software testing from debugging to prevention based testing.
2) Explain why independent testing is required.
3) Explain big bang approach of software testing.
4) Explain total quality management approach of software testing.
5) Explain concept of TQM cost perspective.
6) Explain testing as a process of software certification.
7) Explain the basic principles on which testing is based.
8) Explain the concept of test team's defect finding efficiency.
9) Explain test case's defect finding efficiency.
10) What are the challenges faced by testers?
11) Explain the process of developing test strategy.
12) Explain the process of developing test methodology.
13) Which skills are expected in a good tester?

https://E-next.in

Scanned by CamScanner