Software Testing Syllabus

Contents

Introduction of Software

Introduction of Software Testing

Verification and Validation

Quality Assurance and Quality Control

Verification Strategies

Review

Walkthrough

Inspection

Kickoff

Validation Strategies

White Box Techniques

Statement Coverage

Branch / Decision Coverage

Condition / Path coverage

Black Box Techniques

Equivalence Portioning

Boundary value analysis

Error Guessing

Level of Testing

Software development life cycle

Software testing life cycle

Software Development Models

Test Design and deliverables

Software Requirement Specification

Test Plan

Use Case

Test Scenario

Test Case

Requirement Traceability Matrix

Test Metrics

Test Summary Report

Defect Life Cycle

Risk Based Testing

Regression Testing / Retesting

Smoke / Sanity Testing

Web application Testing Checklist

**Introduction of Software Testing:**

**What Is Testing:**

* The process of executing the program with the intention of finding defects.
* Software testing is the process of testing the functionality and correctness of the software by running it out.
* It is the process of excesing or evaluating the a system means to verify that it satisfies the specified requirement.
* It is process of analyzing a system to detect the difference between existing and required conditions and to evaluate the features of the system.

**Importance of Testing:**

* Ensure that product is usable.
* Ensure that customer objectives are met.
* Early detection of errors to prevent breakdown at later stage.
* Ensure that software is reliable.
* Build confidence in software.
* Increases customer satisfaction.
* Ensure effective execution in the given software.
* Reduces overall cost of software.

**Testing and Quality:**

* Testing help to measure the quality of software in terms of number of defect found.
* Test also reveals important information regarding the nonfunctional attributes like reliability, security, performance etc.
* Testing can give confidence in the quality of the software if it finds few or no defects. When testing does find defects, the quality of the software system increases when those defects are fixed.

**How much Testing is enough?**

* Deciding how much testing is enough should take account of the level of risk, including technical, safety, and business risks, and project constraints such as time and budget.
* Testing should provide sufficient information to stakeholders to make informed decisions about the release of the software or system being tested, for the next development step or handover to customers

**Attributes of Good Tester**

* A good test engineer has “test to break” attitude.
* Ability to understand the point of view of the customer.
* A passion for quality and attentio to details.
* Notice little things that others miss / ignore.
* Ability to communicate fault information to the technical ans non technical.
* Tact and diplomacy for maintaining a cooperative relationaship with developers.
* Work under the worst time pressure.
* Patience.

**Seven Testing Principles:**

**Principle 1 – Testing shows presence of defects**

Testing can show that defects are present, but cannot prove that there are no defects. Testing reduces the probability of undiscovered defects remaining in the software but, even if no defects are found, it is not a proof of correctness.

**Principle 2 – Exhaustive testing is impossible**

Testing everything (all combinations of inputs and preconditions) is not feasible except for trivial cases. Instead of exhaustive testing, risk analysis and priorities should be used to focus testing efforts.

**Principle 3 – Early testing**

To find defects early, testing activities shall be started as early as possible in the software or system development life cycle, and shall be focused on defined objectives.

**Principle 4 – Defect clustering**

Testing effort shall be focused proportionally to the expected and later observed defect density of modules. A small number of modules usually contains most of the defects discovered during pre- release testing, or is responsible for most of the operational failures.

**Principle 5 – Pesticide paradox**

If the same tests are repeated over and over again, eventually the same set of test cases will no longer find any new defects. To overcome this “pesticide paradox”, test cases need to be regularly reviewed and revised, and new and different tests need to be written to exercise different parts of the software or system to find potentially more defects.

**Principle 6 – Testing is context dependent**

Testing is done differently in different contexts. For example, safety-critical software is tested differently from an e-commerce site.

**Principle 7 – Absence-of-errors fallacy**

Finding and fixing defects does not help if the system built is unusable and does not fulfill the users’ needs and expectations.

**Verification and Validation / Quality Assuarance and Quality Control:**

|  |  |
| --- | --- |
| **Verification / Quality Assurance** | **Validation / Quality Control** |
| It is the process to ensure whether we are developing the product accordinglyor not | It is the process to validate the product which we have developed is right or not |
| In simple word, verifcation is verifying the documents | In simple words, validation is to validate the actual and expected output of the software |
| Activities involved here are inspection, review, walkthrough | Activities involved in this is testing the software application |
| Verification is a static method of checking the documents and files | Validation is a dynamic process of testing the real product. |

**Verification Strategies / Static Testing Technique:**

**Review:** Static Review provides a powerful way to improve the quality and productivity of software development to recognize and fix their own defects early in the software development process.

**Activities in Formal Review:**

1. **Planning:**

* Defining the review criteria
* Allocating the roles
* Defining the entry and exit criteria for more formal review (EG: Inspection)
* Selecting which parts of the document to review
* Checking entry criteria (for more formal review types)

1. **Kick-off:**

* Distributing documents
* Explaining the objectives, process and documents to the participants.

1. **Individual Preparation:**

* Preparing for the review meetings by reviewing documents.
* Noting potential defects, questions and comments

1. **Examination / Evaluation / Recording of result (Review Meeting)**

* Discussing or logging with documented result or minutes (For more formal review types)
* Noting defects, making recommendations regarding handling the defects, making decisions about the defects
* Examining / evaluating and recording issues during any physical or group electroninc communications

1. **Rework:**

* Fixing defect found (typically done by developers)
* Recording updated status of defects (in formal review)

1. **Follow up:**

* Checking that defects have been addressed
* Gathering metrics
* Checking on exit criteria (For more formal review types)

**Roles and Responsibilities in a Review**

There are various roles and responsibilities defined for a review process. Within a review team, five types of participants can be distinguished: moderator, author, scribe,reviewer and  
manager.Lets discuss their roles one by one:-

**1. The manager :-** Manager is involved in the reviews as he or she decides on the execution of reviews, allocates time in project schedules and determines whether review process objectives have been met or not.

**2. The moderator**:- The moderator (or review leader) leads the review process. His role is to determine the type of review, approach and the composition of the review team.

The moderator also schedules the meeting, disseminates documents before the meeting, coaches other team members, paces the meeting, leads possible discussions and stores the data that is collected.

**3. The author:-** As the writer of the ‘document under review’, the author’s basic goal should be to learn as much as possible with regard to improving the quality of the document.The author’s  
task is to illuminate unclear areas and to understand the defects found.

**4. The reviewer:-**  Individuals with a technical or businessbackground (also called checkers or inspectors) who, after the necessary preparations, identify and describe the finding (EG: defects) in product under review. Reviewers should take part in any review meetings

**5. The scribe/ recorder :–** The scribe (or recorder) has to record each defect found and open points that were identified and any suggestions or feedback given in the meeting for process improvement.

**Types of Review:**

**1. Walkthrough:**A walkthrough is typically led by the author of a document, for the purpose of educating the participants about the content so that everyone understands the same thing. A walkthrough may include "dry runs" of business scenarios to show how the system would handle certain specific situations.

**2. Inspection:**  
An Inspection is the most formal review techniques. There are strict entry and exit criteria to the Inspection process, it is led by a trained Leader or moderator (not the author), there are defined roles for searching for faults based on defined rules and checklists. Metrics are a required part of the process.

* Led by trained moderator
* Usually conducted as peer examination
* Defined roles
* Includes metrics gathering
* Formals process based on rules and checklist
* Specified entry and exit criteria for the acceptance of software product
* Pre-meeting preparation
* Inspection report includes list of findings.
* Formal follow up process
* Main purpose: Finding defects

**Validation Strategies / Dynamic Testing Technique:**

**1. White Box Testing / Structural Testing / Glass Box / Clear Box / Open Box Testing / Code Based Testing:**

**A) Statement coverage:** Identify the testcase that every line of code is executed.

It involves execution of all the executable statements in the source code atleast once. It is used to calculate and measure the number of statements in the source code which can be executed given the requirements.

**B) Branch / Decision coverage:**Indentify the testcase such that every branch of code is executed in one test case or other.

Test coverage criteria requires enough test cases such that each condition in a decision takes on all possible outcomes at least once, and each point of entry to a program or subroutine is invoked at least once. That is, every branch (decision) taken each way, true and false. It helps in validating all the branches in the code making sure that no branch leads to abnormal behavior of the application.

**C) Condition / Path Testing:**In this the test case is executed in such a way that every path is executed at least once. All possible control paths taken, including all loop paths taken zero, once, and multiple (ideally, maximum) items in path coverage technique, the test cases are prepared based on the logical complexity measure of a procedural design. In this type of testing every statement in the program is guaranteed to be executed at least one time.

**D) Data flow Testing:** Testing in which test cases are designed based on variable usage within the code. Source code is tested to analyze “where the variable has been used” and “where the variable has been defined”

**2. Black box Testing / Behavioural Testing**

**A) Equivalence Partioning:** A portion of the component’s input or output domains for which the component’s behaviour is assumed to be same from the component specification.

* Divide the input, output etc in a range.
* All items in the range should be equivalent.
* Pick one item from each range to derive the test cases.
* Develp the testcases for both valid and invalid partitions.

**B) Boundary Value Analysis:** Analyzing the value which lie at the boundaries

**C) State Transition Testing:** It is an integration testing which includes testing the transition between two allowable states of a system or component.

**D) Random Testing:**

* A non systematic techniques.
* Should be used only after systematic techniques have been executed.
* Involves picking the set of test cases randomly from the present test.
* No set approach in selecting the test cases.
* Also known as Guerilla Testing and Monkey Testing

**E) Error Guessing:** Much ad hoc testing and it is based on intuition and guesswork. Test cases are mainly derived by domain expertise

**Level of Testing**

**1) Unit Testing**

* Component testing (also known as unit, module or program testing) searches for defects in, and verifies the functioning of, software modules, programs, objects, classes, etc., that are separately testable.
* Component testing occurs with access to the code being tested and with the support of a development environment, such as a unit test framework or debugging tool. In practice, component testing usually involves the programmer who wrote the code. Defects are typically fixed as soon as they are found, without formally managing these defects.

**2) Integration Testing:**

* Integration testing tests interfaces between components, interactions with different parts of a system, such as the operating system, file system and hardware, and interfaces between systems.
* Systematic integration strategies may be based on the system architecture (such as top-down and bottom-up), functional tasks, transaction processing sequences, or some other aspect of the system or components
* At each stage of integration, testers concentrate solely on the integration itself. For example, if they are integrating module A with module B they are interested in testing the communication between the modules, not the functionality of the individual module as that was done during component testing

**Stubs:** Stubs can replace the component which are not ready but require to be called during integration testing.

**Drivers / Test Harness:** Drivers replace the component which call the component being tested during the integration testing.

**3) System Integration Testing:**

* System integration testing (SIT) tests the interactions between different systems and may be done after system testing.
* It verifies the proper execution of software components and proper interfacing between components within the solution.
* The objective of SIT Testing is to validate that all software module dependencies are functionally correct and that data integrity is maintained between separate modules for the entire solution.

**4) User Acceptance Testing:**

* The**User Acceptance test**focuses mainly on the functionality thereby validating the fitness-for-use of the system by the business user. The user acceptance test is performed by the users and application managers
* Acceptance testing is basically done by the user or customer although other stakeholders may be involved as well.
* The goal of acceptance testing is to establish confidence in the system.

**Software Development Life Cycle: Waterfall Model**

**1) Requirement gathering and analysis:** Business requirements are gathered in this phase. This phase is the main focus of the project managers and stake holders. Meetings with managers, stake holders and users are held in order to determine the requirements like; Who is going to use the system? How will they use the system? What data should be input into the system? What data should be output by the system? These are general questions that get answered during a requirements gathering phase. After requirement gathering these requirements are analyzed for their validity and the possibility of incorporating the requirements in the system to be development is also studied.

Finally, a Requirement Specification document is created which serves the purpose of guideline for the next phase of the model. The testing team follows the Software Testing Life Cycle and starts the Test Planning phase after the requirements analysis is completed.

**2) Design:** In this phase the system and software design is prepared from the requirement specifications which were studied in the first phase. System Design helps in specifying hardware and system requirements and also helps in defining overall system architecture. The system design specifications serve as input for the next phase of the model.

In this phase the testers comes up with the Test strategy, where they mention what to test, how to test.

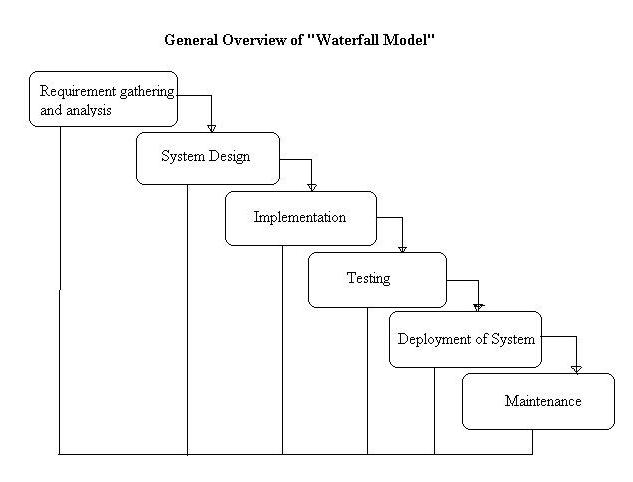
**3) Implementation / Coding**: On receiving system design documents, the work is divided in modules/units and actual coding is started. Since, in this phase the code is produced so it is the main focus for the developer. This is the longest phase of the software development life cycle.

**4) Testing:** After the code is developed it is tested against the requirements to make sure that the product is actually solving the needs addressed and gathered during the requirements phase. During this phase all types of functional testing like integration testing, system testing, acceptance testing are done as well as non-functional testing are also done

**5) Deployment:** After successful testing the product is delivered / deployed to the customer for their use.

As soon as the product is given to the customers they will first do the beta testing. If any changes are required or if any bugs are caught, then they will report it to the engineering team. Once those changes are made or the bugs are fixed then the final deployment will happen.

**6) Maintenance:** Once when the customers starts using the developed system then the actual problems comes up and needs to be solved from time to time. This process where the care is taken for the developed product is known as maintenance.



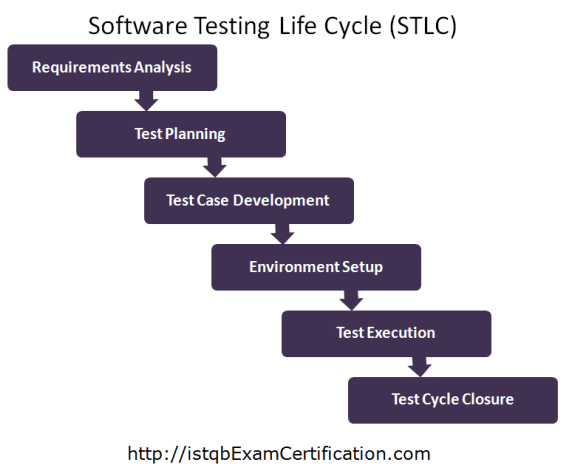
**Advantages of waterfall model:**

* This model is simple and easy to understand and use.
* It is easy to manage due to the rigidity of the model – each phase has specific deliverables and a review process.
* In this model phases are processed and completed one at a time. Phases do not overlap.
* Waterfall model works well for smaller projects where requirements are very well understood.

**Disadvantages of waterfall model:**

* Once an application is in the testing stage, it is very difficult to go back and change something that was not well-thought out in the concept stage.
* No working software is produced until late during the life cycle.
* High amounts of risk and uncertainty.
* Not a good model for complex and object-oriented projects.
* Poor model for long and ongoing projects.
* Not suitable for the projects where requirements are at a moderate to high risk of changing.

**Software Testing Life Cycle:**



Software Testing Life Cycle is a testing process which is executed in a sequence, in order to meet the quality goals. It is not a single activity but it consists of many different activities which are executed to achieve a good quality product. There are different phases in STLC which are given below:

* Requirement analysis
* Test Planning
* Test case development
* Environment Setup
* Test Execution
* Test Cycle Closure

Software testing life cycle (STLC)Each of the step mentioned above has some Entry Criteria (it is a minimum set of conditions that should be met before starting the software testing) as well as Exit Criteria (it is a minimum set of conditions that should be completed in order to stop the software testing) on the basis of which it can be decided whether we can move to the next phase of Testing Life cycle or not.

Let us discuss about each phase in detail:

**Requirement Analysis:** This is the very first phase of Software testing Life cycle (STLC). In this phase testing team goes through the Requirement document with both Functional and non-functional details in order to identify the testable requirements.

In case of any confusion the QA team may setup a meeting with the clients and the stakeholders (Technical Leads, Business Analyst, System Architects and Client etc.) in order to clarify their doubts.

Once the QA team is clear with the requirements they will document the acceptance Criteria and get it approved by the Customers.

**Activities to be done in Requirement analysis phase are given below:**

* Analyzing the System Requirement specifications from the testing point of view
* Preparation of RTM that is Requirement Traceability Matrix
* Identifying the testing techniques and testing types
* Prioritizing the feature which need focused testing
* Analyzing the Automation feasibility
* Identifying the details about the testing environment where actual testing will be done

**Deliverables (Outcome) of Requirement analysis phase are:**

* Requirement Traceability Matrix (RTM)
* Automation feasibility report

**Test Planning:** Test Planning phase starts soon after the completion of the Requirement Analysis phase. In this phase the QA manager or QA Lead will prepare the Test Plan and Test strategy documents. As per these documents they will also come up with the testing effort estimations.

**Activities to be done in Test Planning phase are given below:**

* Estimation of testing effort
* Selection of Testing Approach
* Preparation of Test Plan, Test strategy documents
* Resource planning and assigning roles and responsibility to them
* Selection of Testing tool

**Deliverables (Outcome) of Test Planning phase are:**

* Test Plan document
* Test Strategy document
* Best suited Testing Approach
* Number of Resources, skill required and their roles and responsibilities
* Testing tool to be used

**Test Case Development:** In this phase the QA team write test cases. They also write scripts for automation if required. Verification of both the test cases and test scripts are done by peers. Creation of Test Data is done in this phase.

**Activities to be done in Test Case Development phase are given below:**

* Creation of test cases
* Creation of test scripts if required
* Verification of test cases and automation scripts
* Creation of Test Data in testing environment

**Deliverables (Outcome) of Test Case Development phase are:**

* Test cases
* Test scripts (for automation if required)
* Test Data

**Test Environment setup:** This phase includes the setup or installation process of software and hardware which is required for testing the application. In this phase the integration of the third party application is also carried out if required in the project.

After setting up the required software and hardware the installation of build is tested. Once the installation of build is successful and complete then the Test Data is generated.

After the creation of Test data the Smoke testing is executed on the build in order to check whether the basic functionalities are working fine or not. This phase can be done in parallel with the Test Case Development phase.

**Activities to be done in Test Environment Setup phase are given below:**

* As per the Requirement and Architecture document the list of required software and hardware is prepared
* Setting up of test environment
* Creation of test data
* Installation of build and execution of Smoke testing on it

**Deliverables (Outcome) of Test Environment Setup phase are:**

* Test Environment setup is ready
* Test Data is created
* Results of Smoke testing

**Test Execution:** Before starting the Test Execution phase the Test Environment setup should be ready. In Test Execution phase the test cases are executed in the testing environment.

While execution of the test cases the QA team may find bugs which will be reported against that test case. This bug is fixed by the developer and is retested by the QA.

**Activities to be done in Test Execution phase are given below:**

* Execution of Test Cases
* Reporting test results
* Logging defects for the failed test cases
* Verification and retesting of the defect
* Closure of defects

**Deliverables (Outcome) of Test Execution phase are:**

* Test execution Report
* Updated test cases with results
* Bug Report

**Test Cycle Closure:** In order to start the Test Cycle Closure activity the Test Execution phase should be completed. In Test Cycle closure phase the QA team will meet and discuss about the testing artifacts.

The whole intent of this discussion is to learn lessons from the bad practices. This will help in future projects.

**Activities to be done in Test Cycle Closure phase are given below:**

* To evaluate the test completion on the basis of Test Coverage and Software Quality
* Documentation of the learning from the project
* Analyzing the test results to find out the distribution of severe defects
* Test Closure Report preparation

**Deliverables (Outcome) of Test Cycle Closure phase are:**

* Report of Test Closure

**Test Design and Deliverables**

**1. Software Requirement Specifiction:** SRS defines the following areas.

* Testing scope
* Testing options
* Test plan creation approach
* Data and region requirement
* Interface requirement
* Test scheduling approach

**2. Test Plan:** Test planning is a activity of defining the objectives of testing and the specification of the test activities in order to meet the objective and mission.

Planning is influenced by the test policy of the organization, scope of testing, objectives, risks, constraints, criticality, testability, and availability of resources.

**Test Planning Activities:**

* Determining the scope and the risks and identifying the objectives of testing.
* Defining the overall approach of testing , including the defination of the test levels and entry and exit criteria.
* Integration and coordinating the testing activities into the software life cycle activities ( development, operation, APP support team, Security team, Contact centre agents, ITAM team)
* Making decision about what to test, how the test activities should be done, and how the test result should be validated.
* Scheduling test analysis and design activities.
* **Scheduling** test implementation, execution and evaluation.
* **Selecting metrics** for monitering and controlling test preparation and execution , defect resolution and risk issues.

**3. Use Case:**

* Use cases are a sequence of steps that describe the interactions between the actor and the system. Use cases are defined in terms of the actor, not the system, describing what the actor does and what the actor sees rather than what inputs the system expects and what the system’s outputs.
* Use cases can uncover integration defects, that is, defects caused by the incorrect interaction between different components.

**4. Test Scenario:**

* Scenario is any functionality that need to be tested.
* Test scenario is what to be tested.

**5. Test Case:**

* A test case is a document which consists of a set of conditions or actions which are performed on the software application in order to verify the expected functionality of the feature.
* Test case describe how to be tested

**Layout Of Testcase Document:**

* **Test case ID:** The ID of the test case
* **Test case description**: The objective or summary of the test case
* **Prerequisites**: Any preconditions which need to be executed before starting the test
* **Teststeps**: Procedure to execute the test
* **Testdata**: Data required while executing the test
* **ExpectedResult**: The expected output of the test
* **ActualResult**: The actual output of the test
* **Status**: Pass, Failed, ‘No Run’ when test case is not executed and ‘Blocked’ when high severity bug is found
* **CreatedBy**: Name of the person who wrote the test case
* **Date of creation**: The date on which the test case was authored
* **Executed By**: Name of the person who ran or executed the test case
* **Date of execution**: The date on which the test case was executed

**6. Requirement Traceability Matrix:**

* It is a document that maps and traces user requirement with test cases.
* The purpose of the Requirements Traceability Matrix is to ensure that for all requirements test case has been written.
* It is used to ensure that all the requirements in scope will be tested.

**7. Test Metrics:** In software testing process, Metrics is the measure of a particular characteristics of testing’s performance or efficiency

* It is used for measuring test progress and evaluating the quality of testing and the product.
* It is used for monitoring and controlling test preparation and execution, defect resolution and risk issue.

**a. Test Scripting Productivity Metrics:** The ratio of total raw step written and total man hour efforts required to script them is called test scripting productivity metrics.

* It is used to measure test case/script written productivity and implement proactive/corrective measures to improve productivity further.
* Formula: Test Scripting productivity = Total raw steps written / Efforts in man hour.
* Low producivity may indicate root cause like high rework, low reusability, requirement for training associate etc
* Higher number indicates efficient scripting.

**b. Test Execution Productivity Metrics:** The ratio of total raw test step executed and total man hour efforts required to execute them is called test execution productivity metrics.

* It is used to measure test case/script execution productivity and implement proactive/corrective measures to improve productivity further.
* Formula: Test executio productivity = ((Total raw steps executed) + (Rested raw test steps) + (staging raw test steps)) / Efforts in man hour.
* Low producivity may indicate root cause like high rework, low reusability, requirement for training associate, high number of issues etc
* Higher number indicates efficient execution.

**c. Testing Defect Density:**

* The total number of valid defects per raw test steps (LOC) is called defect density metrics.
* This metrics measures average number of defects for every 1000 raw test step.
* Formula: Testing Defect Density = (Total number of valid defect \*1000) / Total raw steps executed
* High density indicate poor quality of the product.
* Higher the number lower the product quality.

**d. Test Case Effectiveness:** The percentage of testcase that are usefull to reveal the defects is called as the test case effectiveness metric.

* This metric signifies the percentage of the test cases that are useful to reveal the defect.
* Provide indication for implementingcorrective actions to improves the capability of test case and make them more effective in discovering the defects.
* Formula: Test Case Effectiveness: (Number of defects mapped to testcases / Total number of valid defects) \* 100
* Higher the number better is the test case effectiveness
* Low effectiveness indicate requirement of improvement in test cases, need to train associate etc.

**e. Defect Rejection Metrics:** The percentage of rejected defects to the total number of defects that are opened is called defect rejection metrics

* This provide measure for number of defect rejected during the execution phase.
* Formula: Defect Rejection = (Total number of rejected defect / Total number of defects) \* 100
* Higher rejection rate indicate development and testing are not in syns, team needs training etc.
* Higher number signifies poor understanding of functionality / application knowledge.

**Defect:**

When actual result deviates from the expected result while testing a software application or product then it results into a defect.

* A software engineer makes an **error**
* Error creates **defect** in the software
* That can cause **failure** in operation

**Causes Of Software Defects**: Defects occur because human beings are fallible and because there is time pressure, complex code, complexity of infrastructure, changing technologies, and/or many system interactions.

**Defect Life Cycle:**

New

Assigned

Open

Duplicate

Rejected

Deferred

Fixed

Reopened

Retest

Closed

**Defect Status**

* **New**: When a new defect is logged and posted for the first time. It is assigned a status NEW.
* **Assigned**: Once the bug is posted by the tester, the lead of the tester approves the bug and assigns the bug to developer team
* **Open**: The developer starts analyzing and works on the defect fix
* **Fixed**: When developer makes necessary code change and verifies the change, he or she can make bug status as "Fixed."
* **Retest**: Tester does the retesting of the code at this stage to check whether the defect is fixed by the developer or not and change the status to "Re-test."
* **Reopen**: If the bug persists even after the developer has fixed the bug, the tester changes the status to "reopened". Once again the bug goes through the life cycle.
* **Closed**: If the bug is no longer exists then tester assigns the status "Closed."
* **Duplicate**: If the defect is repeated twice or the defect corresponds the same concept of the bug, the status is changed to "duplicate."
* **Rejected**: If the developer feels the defect is not a genuine defect then it changes the defect to "rejected."
* **Deferred**: If the present bug is not of a prime priority and if it is expected to get fixed in the next release, then status "Deferred" is assigned to such bugs

**Defect are categorized based on below parameters:**

* Defect severity
* Defect priority
* Defect types

**1) Severity:**

It is the extent to which the defect can affect the software. In other words it defines the impact that a given defect has on the system. For example: If an application or web page crashes when a remote link is clicked, in this case clicking the remote link by an user is rare but the impact of application crashing is severe. So the severity is high but priority is low.

Severity can be of following types:

**Critical:** The defect that results in the termination of the complete system or one or more component of the system and causes extensive corruption of the data. The failed function is unusable and there is no acceptable alternative method to achieve the required results then the severity will be stated as critical.

**Major:** The defect that results in the termination of the complete system or one or more component of the system and causes extensive corruption of the data. The failed function is unusable but there exists an acceptable alternative method to achieve the required results then the severity will be stated as major.

**Moderate:** The defect that does not result in the termination, but causes the system to produce incorrect, incomplete or inconsistent results then the severity will be stated as moderate.

**Minor:** The defect that does not result in the termination and does not damage the usability of the system and the desired results can be easily obtained by working around the defects then the severity is stated as minor.

**Cosmetic:** The defect that is related to the enhancement of the system where the changes are related to the look and field of the application then the severity is stated as cosmetic.

**2) Priority:**

Priority defines the order in which we should resolve a defect. Should we fix it now, or can it wait? This priority status is set by the tester to the developer mentioning the time frame to fix the defect. If high priority is mentioned then the developer has to fix it at the earliest. The priority status is set based on the customer requirements. For example: If the company name is misspelled in the home page of the website, then the priority is high and severity is low to fix it.

Priority can be of following types:

**Low:** The defect is an irritant which should be repaired, but repair can be deferred until after more serious defect have been fixed.

**Medium**: The defect should be resolved in the normal course of development activities.

**High**: The defect must be resolved as soon as possible because the defect is affecting the application or the product severely. The system cannot be used until the repair has been done.

Few very important scenarios related to the severity and priority which are asked during the interview:

**High Priority & High Severity:** An error which occurs on the basic functionality of the application and will not allow the user to use the system. (Eg. A site maintaining the student details, on saving record if it, doesn’t allow to save the record then this is high priority and high severity bug.)

**High Priority & Low Severity:** The spelling mistakes that happens on the cover page or heading or title of an application.

**High Severity & Low Priority:** An error which occurs on the functionality of the application (for which there is no workaround) and will not allow the user to use the system but on click of link which is rarely used by the end user.

**Low Priority and Low Severity:** Any cosmetic or spelling issues which is within a paragraph or in the report (Not on cover page, heading, title).

**Defect Types:**

* **DO** = Documentation related defect. This include all defect related to missing or misstated requirement in functional specification
* **FN / LO (Code changes:):** Functional / Program logic
* **EN =** Envirenment (Design, compile, test or other supported system problem)
* **Production Behaviour**
* **Data Error / Parameter Related**

**Defect Logging:**

* **Defect ID**– Every bug or defect has it’s unique identification number
* **Defect Description** – This includes the abstract of the issue.
* **Product Version** – This includes the product version of the application in which the defect is found.
* **Detail Steps** – This includes the detailed steps of the issue with the screenshots attached so that developers can recreate it.
* **Date Raised** – This includes the Date when the bug is reported
* **Reported By** – This includes the details of the tester who reported the bug like Name and ID
* **Status**– This field includes the Status of the defect like New, Assigned, Open, [Retest](http://istqbexamcertification.com/what-is-retesting/), [Verification](http://istqbexamcertification.com/what-is-verification-in-software-testing-or-what-is-software-verification/), Closed, Failed, Deferred, etc.
* **Expected date of closure:**
* **Fixed by** – This field includes the details of the developer who fixed it like Name and ID
* **Date Closed** – This includes the Date when the bug is closed
* **Severity**– Based on the severity (Critical, Major or Minor) it tells us about impact of the defect or bug in the software application
* **Priority** – Based on the Priority set (High/Medium/Low) the order of fixing the defect can be made. (Know more about [Severity and Priority](http://istqbexamcertification.com/what-is-the-difference-between-severity-and-priority/))

|  |  |
| --- | --- |
| **Regression testing** | **Retesting** |
| Regression testing is done to find out the issues which may get introduced because of any change or modification in the application. | Retesting is done to confirm whether the failed test cases in the final execution are working fine or not after the issues have been fixed. |
| The purpose of regression testing is that any new change in the application should NOT introduce any new bug in existing functionality. | The purpose of retesting is to ensure that the particular [bug](http://istqbexamcertification.com/what-is-defect-or-bugs-or-faults-in-software-testing/)or issue is resolved and the functionality is working as expected. |
| [Verification](http://istqbexamcertification.com/what-is-verification-in-software-testing-or-what-is-software-verification/)of bugs are not included in the regression testing. | Verification of bugs are included in the retesting. |
| Regression testing can be done in parallel with retesting. | Retesting is of high [priority](http://istqbexamcertification.com/what-is-the-difference-between-severity-and-priority/)so it’s done before the regression testing. |
| For regression testing test cases can be automated. | For retesting the test cases cannot be automated. |

**What is Smoke Testing?**

Smoke Testing is a kind of Software Testing performed after software build to ascertain that the critical functionalities of the program is working fine. It is executed "before" any detailed functional or regression tests are executed on the software build. The purpose is to reject a badly broken application, so that the QA team does not waste time installing and testing the software application.

In Smoke Testing, the test cases chosen cover the most important functionality or component of the system. The objective is not to perform exhaustive testing, but to verify that the critical functionalities of the system is working fine.

For Example a typical smoke test would be - Verify that the application launches successfully, Check that the GUI is responsive ... etc.

**What is Sanity Testing?**

Sanity testing is a kind of Software Testing performed after receiving a software build, with minor changes in code, or functionality, to ascertain that the bugs have been fixed and no further issues are introduced due to these changes. The goal is to determine that the proposed functionality works roughly as expected. If sanity test fails, the build is rejected to save the time and costs involved in a more rigorous testing.

**What Is Risk:** Risk can be defined as the chance of an event, hazard, threat or situation occurring and resulting in undesirable consequences or a potential problem. The level of risk will be determined by the likelihood of an adverse event happening and the impact.

**Risk Based Testing:**

In this approach of testing a risk assessment of the changes/functionalities is performed. The risk assessment would access the risk based o the below areas.

* Customer Impact
* Monetory impact to business
* Compliance and regulatory
* Functional complexity
* Technical complexity
* New product functionalities

The scope of testing is based on risk assessment. The high and medium risk areas will be covered in the testing scope and low risk areas would be considered as optional areas within the scope, and are optional to business to decide on whether should be included. The estimate would be based on agreed scope of teesting.

**Types Of Risk:**

* Project Risk
* Product Risk
* **Project Risk:** Project risks are the risks that surround the project’s capability to deliver its objectives, such as:
  + Organizational factors:
    - Skill, training and staff shortages
    - Personnel issues
    - Political issues, such as: Problems with testers communicating their needs and test results and Failure by the team to follow up on information found in testing and reviews (e.g., not improving development and testing practices)
    - Improper attitude to ard or expectations of testing (e.g., not appreciating the value of finding defects during testing)
  + Technical issues:
    - Problems in defining the right requirements
    - The extent to which requirements cannot be met given existing constraints
    - Test environment not ready on time
    - Late data conversion, migration planning and development and testing data conversion/migration tools
    - Low quality of the design, code, configuration data, test data and tests
  + Supplier issues:
    - Failure of a third party
* **Product Risk:** Potential failure areas (adverse future events or hazards) in the software or system are known as product risks. These include
* Failure-prone software delivered
* The potential that the software/hardware could cause harm to an individual or company
* Poor software characteristics (e.g., functionality, reliability, usability and performance)
* Poor data integrity and quality (e.g., data migration issues, data conversion problems, data transport problems, violation of data standards)
* Software that does not perform its intended functions

**Product Risks** are used to decide where to start testing and wher to test more; testing is used to reduce the risk of an adverse effect occurring, or to reduce the impact of an adverse effect.

A risk-based approach to testing provides proactive opportunities to reduce the levels of product risk, starting in the initial stages of a project. In a risk- based approach the risks identified may be used to:

* Determine the test techniques to be employed
* Determine the extent of testing to be carried out
* Prioritize testing in an attempt to find the critical defects as early as possible
* Determine whether any non-testing activities could be employed to reduce risk (e.g., providing training to inexperienced designers)

**Alpha Testing:**

Alpha testing is carried out at the developer’s site and the testers are internal employees of the organization. Alpha testing is a type of acceptance testing; performed to identify all possible issues/bugs before releasing the product to everyday users or public.

|  |  |
| --- | --- |
| **Load Testing** | **Stress Testing** |
| Load testing studies the behaviour of the application under various loads | Identify the breaking point of the server |
| In this type of testing, application are simultaneously hit by thousand of users | In this type of testing, we try to break the system to determine how the system recover after crash |
| In this type of testing, we determine the reliability of the application | Here we determine the stability of the application |
| The main parameter to focus is response time | The main parameter to focus is throughput |

**Web Application Testing Checklist:**

* Functionality Testing
* Usability Testing
* Interface Testing
* Database Testing
* Compatibilty Testing
* Performance Testing
* Security Testing

**Functionality Testing:** Here we test end to end business flow/Functional flow of the application is as per client requirement or not.

Ex: Test all links in your webpages are working correctly and make sure there are no broken links.

## Usability testing: Here we test Menus, buttons or Links to different pages on your site should be easily visible and consistent on all webpages.

## Content should be legible with no spelling or grammatical errors

**Interface Testing:** Three areas to be tested here are - Application, Web and Database Server

* Application: Test requests are sent correctly to the Database and output at the client side is displayed correctly. Errors if any must be caught by the application and must be only shown to the administrator and not the end user.
* Web Server: Here we test whether Web server is handling all application requests without any service denial.
* Database Server: Make sure queries sent to the database give expected results.

Here we test how the system respond when connection between the three layers (Application, Web and Database) cannot be established and appropriate message is shown to the end user.

## Database Testing: Database is one critical component of your web application. Testing activities will include:

* Test if any errors are shown while executing queries
* Data Integrity is maintained while creating, updating or deleting data in database.
* Test data retrieved from your database is shown accurately in your web application

## Compatibility testing

## Compatibility testing make sure that website works fine across various Operating systems such as Windows, Linux, Mac.

## It also make sure that web applications are working correctly across various browser like Firefox, Internet Explorer, Safari etc

## Performance Testing:

This will ensure your site works under all loads. Testing activities will include but not limited to –

* Website application response times at different connection speeds
* Load test your web application to determine its behavior under normal and peak loads
* Stress test your web site to determine its break point when pushed to beyond normal loads at peak time.
* Test if a crash occurs due to peak load, how does the site recover from such an event

## Security Testing

[Security Testing](https://www.guru99.com/what-is-security-testing.html) is vital for e-commerce website that store sensitive customer information like credit cards. Testing Activities will include following-

* Test unauthorized access to secure pages should not be permitted
* Restricted files should not be downloadable without appropriate access
* Check sessions are automatically killed after prolonged user inactivity
* On use of SSL certificates, website should re-direct to encrypted SSL pages