1. **SDLC:**

Requirement: Business analyst collects requirements from customer and makes a doc BRS.

Analysis: Here requirements are documented and analysed and got approved by the customer. Here SRS doc is used.

Design: *High level doc*: Components design needed t

Develop the overall software, *Low level doc*: what should be the design of each module(only designs and no codes).

Development: Developers start developing the product according to requirement .Output is src code and soln doc.

Testing: Code is tested by testers either manually or automation thoroughly and test report is produced.

Deployment and maintenance: Deployment of product for customer use and maintenance if any issues.

1. **STLC:**

Requirement analysis: Is carried by referring to BRD and here analysis is done regarding what is testable and what is not same is reported.

Test Planning: Leads estimate cost of testing effort , tool selection, training etc

Test design: Team starts preparing test cases, scripts, test data and gets them approved. Also prepares requirement traceability matrix.

Test environment setup: software and hardware for testing will be ready and testing team needs to prepare test cases for smoke testing to check the readiness of the build.

Test Execution: execute test cases , report bugs, prepare test failure doc. Log a defect and re test once fixed and close the defect.

Test Closure: Test Metrics are prepared, test closure reports, defect report, meeting is called for overall testing completion.

1. **Entry and Exit criteria:** Entry: pre-requisites to start testing or go to next phase of SDLC,

Exit- condition fulfilled to complete testing.

1. **Test Design Metrics:** We can measure the progress of the testing by using different test metrics. Test completion status, Test execution completion status**,** defect density, defect leakage, test effectiveness, defects by priority, defects by severity, productivity of testing, defect acceptance, defect rejection, effort variance,.
2. **Test Deliverables or Test artifacts:**

* Test Strategy: Static high level document which has information about how we go about testing .It refers BRD document and is shared with stakeholders.
* Test Plan: Document prepared to plan the type of testing to be done to deliver the software. It refers SRS doc.
* Effort estimation plan: Document which has actual effort and estimated effort o each task.
* Test Scenario: High level test cases to test particular functionality.
* Test Cases/Scripts: set of positive and negative steps needed to complete running a test referring the test scenario. It has pre-conditions description, actual and expected result post-Conditions.
* Test Data: data for positive and neg test cases in DB, excel sheet .csv files et.
* Requirement traceability matrix: Matrix in which we trace r\test cases to requirements.
* Bug Report: detailed description of defect.
* Test Execution report: defines test cases executed, failed passed etc.
* Metrics: Measure of test progress.
* Test Summary report: activities for testing summarized in the report.
* Test Incident report: Report of all the abnormal events that occurred during testing and needs to be reviewed.
* Test closure report: how many created, executed, defects closed blocked rejected etc.

1. **Testing**: Basically actual versus expected.
2. **Types of testing:** Functional and non-functional testing.
3. **Types of Functional testing:**

* System Testing: Functionality of whole system as one is tested.
* Regression Testing: After bug fix verifying the fixed bugs and also the functionality related to that bug .
* Re-Testing: Testing only the bugs that are fixed.
* Smoke Testing:
* Sanity Testing:
* Unit Testing: done by developer.
* Integration Testing: After combining two modules of software testing is done.
* Database testing: All the fields in table are populating and all.
* White Box Testing: code is visible.
* Black Box Testing: Source code is not visible. Only input and output to be taken care.
* Grey Box Testing: Testing components sometime are exposed, some are not.
* Monkey Testing: Have no time to test the test cases ,randomly play with the application and explore the features to test. Here you will have rough idea about application and diff use cases.
* Adhoc Testing: Random testing without knowing the different types of use cases and application.
* Component Testing: only testing the specific component of the application.
* GUI Testing: interface testing for proper display of application interface heading, colour, images etc.
* End to end Testing: complete business flow check.
* UAT: \**Alpha testing-* done by tester, one from dev, business analyst together.

\**Beta testing*- Done by end users after the code is deployed in the production. It mainly relies on the customer feedbacks.

* Acceptance Testing: Internal testing done to check whether the product is developed in the right way.
* Happy Path Scenarios: High level testing of only positive scenarios. Done usually in production.
* Negative Scenario Testing: done during system testing and also regression testing giving negative values and checking behaviour.
* Recovery Testing: Application got crashed then how easily will recovery.
* Static Testing: only when requirement is present but code not fully developed, doing some reviews. Three types: \*Walkthrough \*Review \*Inspection.
* Compatibility Testing: Cross browser kind of.

1. **Non-Functional Testing:**done when all the features are working fine and app is error free.

* Performance Testing: Testing with many users. \*Load testing: Give different number of inputs , increase it every time and see the response time \*Stress Testing: threshold for the application. Define the max value \*Volume Testing: with different volumes check the response of application.
* Security Testing: Penetration, SQl injection, network layers etc. Authentication taken care.

**Types of testing**

1. **Exploratory testing:**

Type of software testing where application is being tested without any requirement document or reference to any type of test scripts. Just done to explore the application better. There should be some experienced testers for this. Done when there is critical application. Simultaneous test, design and execution. Learning test results and creating a new solution.

Disadvantages: sometimes a feature is mistaken as a defect, sometime a defect is mistaken as a feature(OTP should generate both in phone and mail).

1. **Adhoc Testing:**

Type of an unplanned testing, usually used for negative test case execution to break the system. It uses a technique called error guessing where the experts no that maximum errors occur at this module. It will not follow any test documents. Usually done after structured testing is over. To verify things beyond the requirements from users perspective.

Here defect recording is most important.

Eg : testing the back button of the amazon payment page where the cart page should reopen. This is not present in the requirement but done as an user’s perspective

1. **Usabilty Testing:**

Type of software testing done from user’s point of view . defines how friendly is the your application. Eg: the submit button should not be small and should be visible clearly. Any performance flaws in between.

1. **Boundary value Analysis:**

Type of black box testing where extreme end or partition of the input values are given and a functionality is tested.

Here inputs applied are : min, just above min, nominal, just below max and max.

Equivalence class partition plays a huge role here

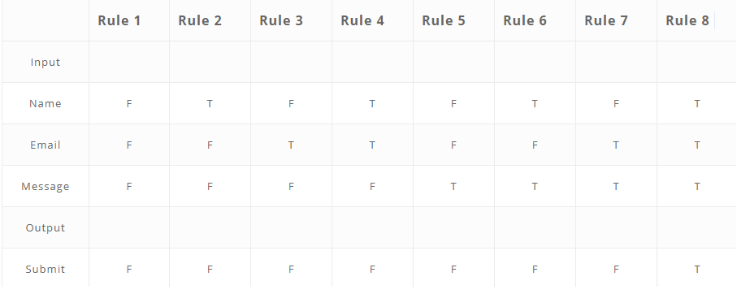
It is believed that most of the errors occur at the boundaries.

1. **Equivalence class Partioning:**

Done when we have to divide the range of imputs into a sets of similar input class. Here the test condition can be divided to equivalence input class that can be considered the same.

1. **Decision table testing:**

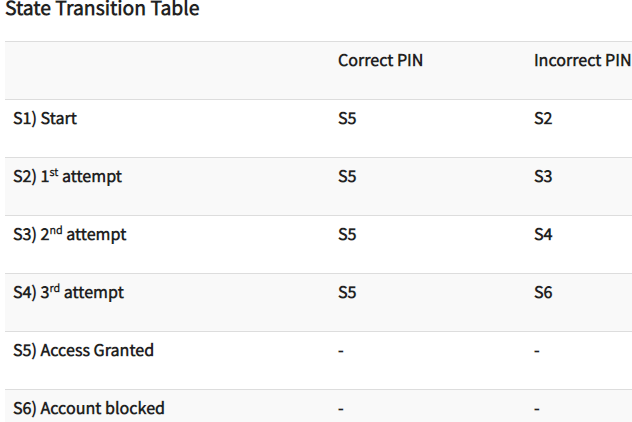
Also known as cause-effect graph .This type of technique is used when an output is achieved by giving set of input value. Eg: submission of a form



1. **State Transition Technique:**

Is a technique used when changes in the input conditions changes the state of the application output.

Eg : three times entering invalid username/password locks ur account.



1. **Types of white box testing**

Statement coverage

Decision coverage

Loop coverage

1. **Back box testing technique**

Boundary value analysis, equivalence class partitioning, cause effect graph, state transition table.

1. **Test Plan**

Test case identifier, scope, features to be tested, features not to be tested, risk, training, responsibility, timeline, test strategy.

1. **UAT:**

To give appropriate input to check whether the application meets the users needs or no. Done in the separate environment which is like production. Here users and clients do it . Business requirements should be present , defects should be regression tested, RTM, test closure sign off of system testing must be done.

1. **System Testing**

Done after integrating all the modules , entire business flow is tested.

1. **Valuable steps to resolve issues while testing**

Record, report, Control.

1. **Latent defect**

Present in the application since long which has not caused any failures because such set of inputs are never come across.

1. **Measure the quality of test execution**

Defect rejection ratio and defect leakage ratio(no of defects missed/total no of defects)

1. **Mutation testing**

Is a fault based testing technique. Its done to know the efficiency of the test cases. Its changing the source code deliberately and checking if there are errors. If no errors then test cases are not up to date

Eg if(x>y)

1. **Fault based testing**
2. **Fault masking**

Presence of one defect hides the presence of another defect.

If negative values introduces some unhandled exceptions then negative values are avoided and exceptions don’t occur but hiding the actual error that could occur by giving neg values.

1. **Priority and severity**

Severity is the effect that application will have on the response of the application

Critical, high, medium, low

Priority: is the order in which the developer fixes the defects.

Low , medium, high

1. **How to eliminate product risk of the project**
2. **What is the common risk that leads to project failures**
3. **Test report contents**

It contains: project information, test summary, test objective and defect

1. **Data flow diagram**

Flow of data is represented in the form of diagram to understand the flow of data.

1. **N+1 Testing**

Is the form of regression testing where testing is done and defects are found in N cycle and N+1 cycle of testing is done to verify if the errors are rectified or not. This kind of testing is done until errors are resolved.

1. **Fuzz testing**

Used to break the system , used to identify the security loop holes or errors in code.

This is done by giving wrong data inputs. Eg: memory leaks , wrong input values can be found out using fuzzers(tool).

1. **Code coverage tool /test coverage tool**

Runs parallel with the test execution monitors the execution of source code line by line. Finally gives the percentage of code coverage at the end of the test execution

1. **Different types of test plans used during span of the project**

Project test plan , acceptance test plan, system test plan and integration and unit test plan

1. **Test coverage techniques**

Decision coverage, statement coverage, path coverage

1. **Risk based testing**

Technique used to minimize the probability of the risk. It involves assessing the risk based on complexity, business criticality, visibility, defect prone area, usage frequency. It involves prioritizing the areas , functions or modules of risk prone areas.

Positive risks : new business idea or project

Negative risk: threats or recommendations to minimize them should be implemented in the project.

1. **Preventive and reactive approach on testing**
2. **Severity and priority**

|  |  |
| --- | --- |
| Priority | Severity |
| Related to order in which bug will be fixed | Related impact to application |
| Associated with scheduling | Associated with functionality and standards |
| How soon the bug should be fixed | Seriousness of the defect on the functionality |
| Manager or clients decide priority | QA engineer decides |
| Status is decided based on customer requirements | Based on technical aspect of the product |
| *High-*  *Medium-*  *Low-* | *Critical*-blocker to application(Login to application, showing server error or showing blank page)  *Major*- Major functionality is not working yet application working(Amount not getting credited even showing success)  *Medium*-causes undesirable behaviour but still functional(sent a mail nothing is displayed but sent in mail box)  *Low*-not an effect to application(some font size is wrong, alignment issues) |

Severity

Priority

1. **SMOKE TESTING:** Type of testing done by tester once the build is given by the developers. This involves testing of critical and main functions of the build whether they are working fine for the testers to decide whether the application build can be further testable or no.
2. **SANITY TESTING:** It is a subset of regression testing. If there are any new minor functionalities included or any defect fixes done in the current build sanity testing is done to verify whether the above two situation have not caused any effect on the other functionalities of the application.
3. **DIFFERENCES:**

|  |  |
| --- | --- |
| SMOKE TESTING | SANITY TESTING |
| * Major objective is to verify if the build is stable and testable or no. | Objective is to verify whether new functionalities and defect fixes do not affect other parts of application. |
| * Done by developers and testers | Done only by testers |
| * Major test cases are referred to and tested | High level functionalities are tested without any test case doc. |
| * Application starting or no, main page appearing , we can login or no |  |
| * Entire testing end to end. | Only particular component of the entire system |

1. **DIFFERENCES:**

|  |  |
| --- | --- |
| REGRESSION | RE-TESTING |
| Repeated testing of an already tested program, after modification to discover any defects introduced or uncovered as a result of changes in software being tested or in a related or unrelated component. | Re-Testing involves testing of defects that are fixed against the build . It involves in depth testing of that big fixed area. |
| Done when defect is being fixed, CR, performance issue fixed, env change. | Only includes re-executing test case that are failed earlier not include related passed test cases |
| Checks unexpected side effects | Carried out before regression testing |
| Purpose- new code change does not affect existing functionality. | Purpose – that particular defect is working fine. |

1. **Verification and Validation differences:**

Verification: is the process of checking whether the software meets the specifications. Are we building the product right?

Validation: Is the process of checking is the specification captures customer needs.

|  |  |
| --- | --- |
| VERIFICATION | VALIDATION |
| Static testing | Dynamic testing |
| Docs, design, code, program | Actual testing |
| Does not involve executing of code. | Involves executing of code. |
| Walkthrough, review, Inspection | White, black and grey box testing. |

1. **Boundary value Analysis:**

It’s a test case design strategy for black box testing. This technique is used to identify errors at the extreme boundaries at the input domain.

Eg: Suppose our testing module accepts 1 to 1000as input range, we can have lower boundary inputs such as 0 and 99 and higher like 2 and 1001 as part of negative testing .

1. **Equivalence Partitioning:**

Its also a testing strategy for black box testing. Here huge range of input is divided into different equivalence data classes.

Eg : Divide input domain into 3 classes if we have to test a specific module which can accept inputs from 1 to 100.

Class 1 : input from 1 to 100 any one test data.

Class2: < 1 neg case

Class 3: > 100 neg case.

**10.Test Coverage:**

It is analysis which helps in knowing if all the requirements are tested at least once.It can have various things and different in different teams.

1. There should be at least on test case around one requirement.
2. All the testers should be assigned to test cases equally.
3. All the requirements must be covered with all the test cases designed.

Eg for test coverage: testing a notepad application. Check for font, colors format, tabs etc is fine but testing how it works when opened with different application, when it’s closed opened many times.

Also we need to test Risk coverage: eg - when testing web –based application.

How to adopt a proper test coverage method:

* Aware of how much work is involved and should they add any more test cases by knowing till where the implementation has progressed.
* Take help of the RTM
* Should be aware of each and every requirement.
* Focus on critical requirement. Get reminders from previous releases also.
* Have automation
* Have a test management tool.
* Keep track of bulid cycles and defect fixes.

1. **Testing without solution doc being give? Use** exploratory testing.
2. **Usability Testing:**  the ease with which an end user can easily access the application.
3. **Big Bang Approach:** Combining all the modules at once and testing the flow once to check if independent modules work fine as a whole application.
4. **Top- Down Approach;** Here all the high level
5. **How to initialize page factory:** by using *PageFactory.initElements(d, THIS);*

**Manual Testing Concepts:**

1. SOFTWARE TESTING: basically validating the software developed from actual to expected. Its a phase in SDLC and happens after development. It is checking if all the requirements are met according to client’s need. It is identifying gaps and errors.
2. WHITE BOX AND BLACK TESTING
3. GREY BOX TESTING: internal code is partially available to the testers. Here algorithms and data structures are exposed.

Eg: Regression testing

1. TEST STRATERGY: is a high level document prepared to understand how to go about testing. It defines the test coverage and testing scope. When there is a proper test strategy doc possibility of missing any testing is very low. It usually refers to the BRS doc and helps in finding any missing requirements.
2. Scope and overview
3. Test Approach: test levels, test types, environment requirements, roles and responsibility
4. Testing tools
5. Industry standards to follow
6. Test deliverables
7. Testing metrics
8. Requirement Traceability Matrix
9. Risk and mitigation
10. Reporting tool
11. Test summary
12. TEST DELIVERABLES:

* Test strategy
* Test plan
* Effort estimation report
* Automation feasibility report
* Test Scenario
* Test cases
* Test data
* Test metrics
* Bug report
* RTM
* Test summary report
* Test incident report
* Test execution report
* Test closure report.
* User guide
* WSR
* Installation/Configuration report
* Release notes: sent to client on what is being fixed and what is being developed and even open bugs need to be mentioned

1. Requirement traceability matrix: Mapping requirements to test cases.

Forward traceability: Are we building the product right, whether all the requirements are being covered

Reverse traceability: Whether right product is being built is the coding going beyond the requirements anything unwanted being built and tested.

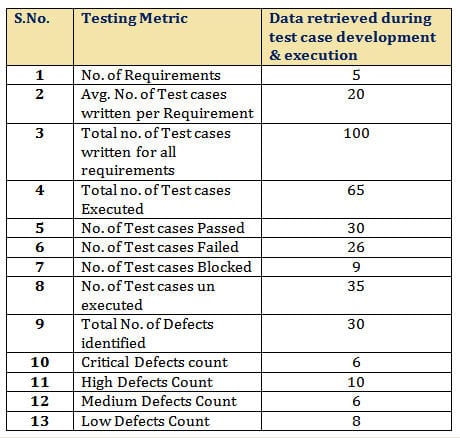
Bi-Directional traceability:

1. Test Metrics:

Base metrics : calculated by the tester like total no of test cases , total failed , passed , executed etc.

Calculated metrics: Can be derived from the base metrics can be used by the managers or lead to check the testing progress.

Eg: % test cases failed,passed etc.



1. **%ge Test cases Executed** =**(**No. of Test cases executed / Total no. of Test cases written) \* 100.
2. **%ge Test cases not executed** =**(**No. of Test cases not executed / Total no. of Test cases written) \* 100.
3. **%ge Test cases Passed** =**(**No. of Test cases Passed / Total no. of Test cases Executed) \* 100.
4. **%ge Test cases Failed** =**(**No. of Test cases Failed / Total no. of Test cases Executed) \* 100.
5. **%age Test cases Blocked** =**(**No. of Test cases Blocked / Total no. of Test cases Executed) \* 100.
6. **Defect Density =**No. of Defects identified / size

**Here size is no of requirements**

1. **Defect Leakage = (**No. of Defects found in UAT / No. of Defects found in QA testing.) \* 100
2. **Defect Removal Efficiency (DRE) = (**No. of Defects found during QA testing / (No. of Defects found during QA testing +No. of Defects found by End-user)) \* 100
3. **Defects by Priority**: **%ge Critical Defects** = No. of Critical Defects identified / Total no. of Defects identified \* 100

**%ge High Defects** = No. of High Defects identified / Total no. of Defects identified \* 100

1. SYSTEM TESTING: Testing fully integrated application end to edn is called system testing.
2. STUBS: dummy programs/modules called by the module under test
3. DRIVERS: dummy programs or modules that call the module to be tested
4. Non-Functional testing: testing how fast the system responds , the performance of the application mainly load, stress , scalability, security and compatibility
5. ACCEPTANCE TESTING:

Alpha testing: done by in house developers and testers at the work location side. Sometimes done by the clients

Beta Testing: Done by the limited number of end users at the client’s place

Gamma Testing: testers are not involved . Gamma version is released to limited end user just before the go live. Performance and security is considered while testing and there will be mostly no scope for enhancements at this point in time.

1. PRINCIPLES OF TESTING:

* Testing shows the presence of defect.
* Exhaustive testing is not possible
* Absence of error fallacy
* Pesticide paradox
* Defect clustering
* Early testing
* Testing is context dependant

1. DEFECT CLUSTERING: triggering of one defect by another defect usually by not noticing any defect which would lead to new defects in later stages.

Eg: if an application accepts non related customers to login which goes un noticed and later if the count of legal customers are sent request of feedback and review then unrelated customer also gets the request which would result in a large error in the application.

1. Defect, Bug, Error(compilation of code execution error), Failure(occurs after deployment)
2. Priority and Severity:

* High Severity High Priority: login button not working
* High Severity low Priority: defect in reservation functionality which could be fixed in next release
* low Severity High Priority: logo improper
* low Severity low Priority: FAQ page takes long time to load

1. BUG LIFE CYCLE: new 🡺open🡺assigned🡺fixed🡺testing🡺verified🡺closed

Rejected, deferred, duplicate, reopened, not reproducible, needs more information

1. TEST PLAN: Test plan is a dynamic document which is a blueprint of the testing activity that has to be carried out. It gives the strategy, schedule, deliverables estimations of the application to be tested.

Components of test plan:

* Scope
* Out of Scope
* Assumptions
* Schedules: how many days for design how many days for execution
* Roles and responsibility
* Test environment
* Test deliverables
* Risks and mitigation plan
* Bug tracking tools
* Automation and manual tools
* Exit criteria

Also includes:

* Test Strategy
* Test coverage
* Test cycles
* Pass/fail criteria
* Business requirements
* Features to be tested
* Features not to be tested
* Tools
* Types of testing

AGILE METHODLODY:

Its combination iterative and incremental(Here requirements keep on changing and new requirements also will come in)

Principles: 1. Requirements are allowed to change at any stage of the development

2. releases happen in a short gap

3.Main goal is to stratisfy the customers

4. There is good communication with clients, BA developers testers

SCRUM:

* *Product backlog:* Prepared by product manager
* *SPRINT*: every one sprint something is being delivered to the customer
* *Sprint planning*: meeting conducted with dev testing and architecture to take up the work
* *Scrum master*: uses JIIRA and allocates work by creating stories , tasks
* *SCRUM meeting :* work updates
* *Sprint retrospective meeting:* meeting at the end of sprint to know what went well and what did not go well
* *Bug triage meeting:* test manager will conduct and dev testers will be there. List of bugs in open state are shown and bugs are re prioritize and discuss whether fixed in this release or next
* *Stories:*  features or module
* *Epic:* Collection of stories
* *Story points:* approx of deadline of task

1. How to will you identify the quality of testing?

Ans: By knowing the defect rejection ratio and defect leakage ratio.

1. MUTATION TESTING:

Changing the code by giving valid inputs to check the correctness of the testing being performed.

1. LATENT DEFECT: A defect which is existing from long time in the application but still has not caused an issue as such input conditions are never been met
2. MASKED DEFECT: A defect that has been present in the system but never been tested because another defect as been effected by that part of the application.
3. RISK BASED TESTING:

Here testing is done identifying risks in the application and choosing such test cases and prioritizing risk involving test cases.

|  |  |
| --- | --- |
| EXPLORATORY | MONKEY |
| Testers are involved in minimum planning and maximum test execution | Random test data is prepared by some tool and passed to the application to see the outcome and the adverse effect |
| Here they will try to find out and test the features which are not present in the test case documents | Basically to check if system fails if we type anything randomly |
| Here test design techniques could also be used for eg: tester may use boundary value analysis without formally writing the boundary test cases |  |

HP-ALM

1. Workflow of hpalm :

1. Release specification
2. Requirement specifications
3. Test planning
4. Test execution
5. Defect tracking

Management tab: Release -is a group of changes

Release:

Cycle: on round of development and testing

In management tab and will be displayed under releases: eg: smoke regression, integration, end to end

Requirement:

New: type of requirement, which release, which cycle, Direct cover status: Not covered, passed, failed, blocked, No run, NA, Not completed.

Requirements export to ALM:

Login🡺 map🡺 validate🡺upload

Risk assessment: Highly risky elements should be tested first.

1. **Various Tabs in HP-ALM:**

DASHBOARD: analysis view, dashboard view.

MANAGEMENT: Releases.

REQUIREMENT:

TESTING: Test resources, Test plan, Test lab, Test runs.

DEFECTS: