Day-6

Test Design Techniques/Test Data Design Techniques/Test case Design Techniques

---

Used to prepare data for testing.

1) Reduce the Data
2) More Coverage

Types

Equivalence Class Partitioning (ECP)
Boundary Value Analysis (BVA)
Decision Table based testing.
State Transition
Error Guessing

Test Design Techniques

- Test design techniques helps to design better cases.
- Reduce the number of test cases to be executed.
- Techniques:
  - Equivalence Class Partitioning
  - Boundary Value Analysis (BVA)
  - Decision Table based testing.
  - State Transition
  - Error Guessing

Equivalence Class Partition (ECP)

- Partition data into various classes and we can select data according to class then test. It reduce the number of test-cases and saves time for testing.

Enter a Number: [ ] * Allow Digits from 1--500

Normal Test Data	Divide values into Equivalence Classes	Test Data using ECP
1		-50
2		30
3	-100 to 0 → **-50** (Invalid)	160
4	1 – 100 → **30** (Valid)	250
.	101 – 200 → **160** (Valid)	320
.	201 – 300 → **250** (Valid)	450
.	301 – 400 → **320** (Valid)	550
.	401 – 500 → **450** (Valid)	
500	501 – 600 → **550** (Invalid)	

Equivalence Class Partition (ECP)

Name: [ ] * Allow only alphabets

Divide values into Equivalence Classes

A..Z → (Valid)
a..z → (Valid)
Special Characters → (Invalid)
Spaces → **250** (Invalid)
Numbers → **320** (Invalid)

Test Data using ECP
XYZ
zyz
@#$%
Xy z
1234

Equivalence Class Partitioning (ECP)

Value check
clasify/devide/partiton the data in to multiple classes

Boundary Value Analysis (BVA)

Boundary of the values

Min
Min+1
Min-1

Max
Max+1
Max-1

Boundary Value Analysis (BVA)

- BVA technique used to check Boundaries of the input.

Enter a Age: [ ] * Allow Digits from 18--35

Min = 18 (Pass)
Min-1 =17 (Fail)
Min+1 =19 (Pass)

Max = 35 (Pass)
Max-1 =34 (Pass)
Max+1 =36 (Fail)

Input domain testing (Most Imp for Interviews)

———————————————————————————————————————

The value will be verified in the text box/input fileds.
We use ECP & BVA techniques

Decision table

———————————————————————————————————————

If we have a more number of conditions /actions, then we use decision table technique.

Decision Table

- Decision Table is also called as Cause-Effect Table.
- This technique will be used if we have more conditions and corresponding actions.
- In Decision table technique, we deal with combinations of inputs.
- To identify the test cases with decision table, we consider conditions and actions.

Decision Table Example

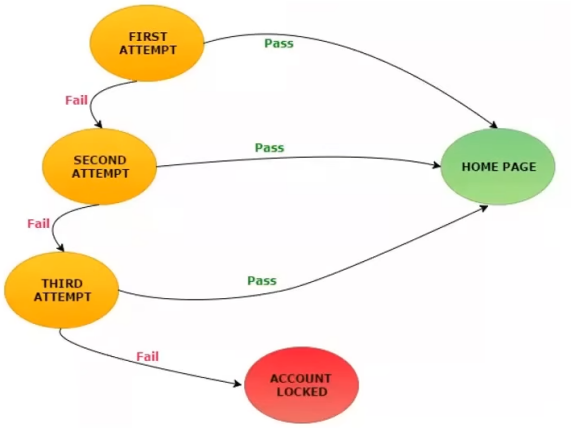
- Take an example of transferring money online to an account which is already added and approved.
- Here the conditions to transfer money are
  - Account already approved
  - OTP (one time password) matched
  - Sufficient money in the account
- And the actions performed are
  - Transfer money
  - Show a message as insufficient amount
  - Block the transaction incase of suspicious transaction

Decision Table Example...

		TC1	TC2	TC3	TC4	TC5
Condition1	Account already approved	TRUE	TRUE	TRUE	TRUE	FALSE
Condition2	OTP Matched	TRUE	TRUE	FALSE	FALSE	X
Condition3	Sufficient Money in the Account	TRUE	FALSE	TRUE	FALSE	X
Action1	Transfer Money	Execute				
Action2	Show message 'Insufficient Amount'		Execute			
Action3	Block the transaction Incase of Suspicios Transaction			Execute	Execute	X

State Transition Example

- Take an example of login page of an application which locks the user name after three wrong attempts of password.



STATE	LOGIN	CORRENT PASSWORD	INCORRECT PASSWORD
S1	First Attempt	S4	S2
S2	Second Attempt	S4	S3
S3	Third Attempt	S4	S5
S4	Home Page		
S5	Display a message as "Account Locked, please consult Administrator"		

Error Guessing

- Error guessing is one of the testing techniques used to find bugs in a software application based on tester's prior experience.
- In Error guessing we don't follow any specific rules.
- It depends on Tester Analytical skills and experience.
- Some of the examples are:
  - Submitting a form without entering values.
  - Entering invalid values such as entering alphabets in the numeric field.

Day-7

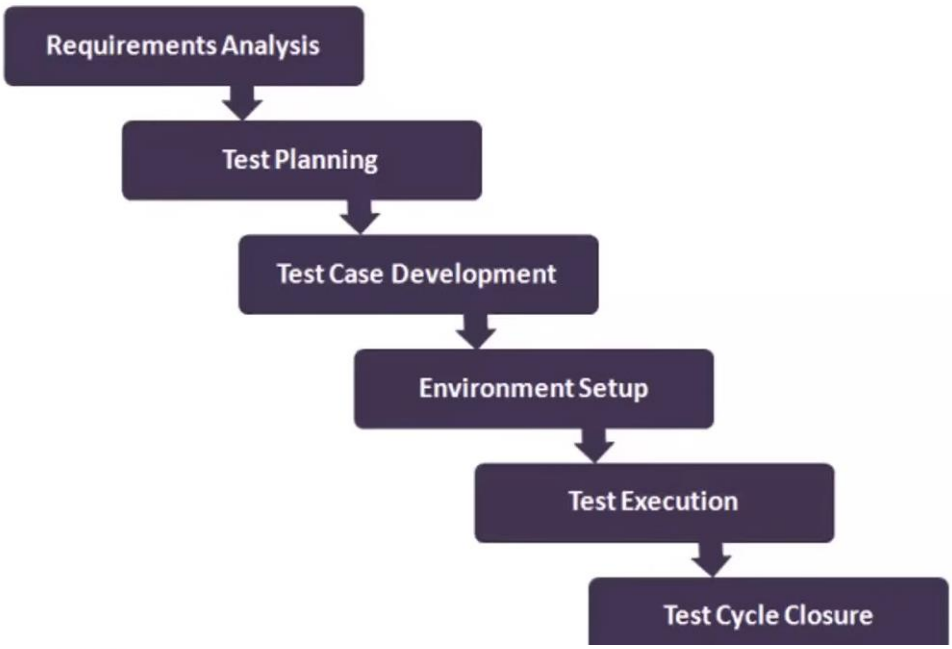
SDLC - Software Development Life Cycle

ReqAnalysis, Desing, coding, testing, deployment, maintenance.

STLC - Software Testing Life Cycle

1) Requirement Analysis
2) Test Planning
3) Test Desing
4) Test Execution
5) Defect/Bug Reporting & Tracking
6) Test Closure

Software Testing Life Cycle (STLC)



STLC

SNO	PHASE	Input	Activities	Responsibility	Out Come
1	**Test Planning**	Project Plan	➢Identify the Resources	Test Lead/Team Lead (70%)	Test Plan Document
	What to test	Functional Requirements	➢Team Formation	Test Manager (30%)	
	How to test		➢Test Estimation		
	when to test		➢Preparation of Test Plan		
			➢Reviews on Test Plan		
			➢Test Plan Sign-off		
2	**Test Designing**	Project Plan	➢Preparation of Test Scenarios	Test Lead/Team Lead(30%)	Test Cases Document
		Functional Requirements	➢Preparation of Test Cases	Test Engineers(70%)	Traceability Matrix
		Test Plan	➢Reviews on Test Cases		
		Design Docs	➢Traceability Matrix		
		Use cases	➢Test Cases Sign-off		
3	**Test Execution**	Functional Requirements	➢Executing Test cases	Test Lead/Team Lead(10%)	Status/Test Reports
		Test Plan	➢Preparation of Test Report/Test Log	Test Engineers (90%)	
		Test Cases	➢Identifying Defects		
		Build from Development Team			
4	**Defect Reporting & Tracking**	Test Cases	➢Preparation of Defect Report	Test Lead/Team Lead(10%)	Defect Report
		Test Reports/Test Log	➢Reporting Defects to Developers	Test Engineers (90%)	
5	**Test Closure/Sign-Off**	Test Reports	➢Analyzing Test Reports	Test Lead/Test Manger(70%)	**Test Summary Reports**
		Defect Reports	➢Analyzing Bug Reporting	Test Enginners(30%)	
			➢Evaluating Exit Criteria		

Day-8

Topics Covered

———

1. Test Plan 2. Use Case Vs Test Scenario Vs Test Case 3. Test Case Template 4. RTM (Requirement Traceability Matrix) 5. Test Environment Setup & Test Execution 6. Defects/Bugs 7. Contents is Defect Report 8. Defect Classification (Severity & Priority)

Test Plan Contents

- A Test Plan is a document that describes the test scope, test strategy, objectives, schedule, deliverables and resources required to perform testing for a software product.

- Test plan template contents:
  - Overview
  - Scope
    - Inclusions
    - Test Environments
    - Exclusions
  - Test Strategy
  - Defect Reporting Procedure
  - Roles/Responsibilities
  - Test Schedule
  - Test Deliverables
  - Pricing
  - Entry and Exit Criteria
  - Suspension and Resumption Criteria
  - Tools
  - Risks and Mitigations
  - Approvals

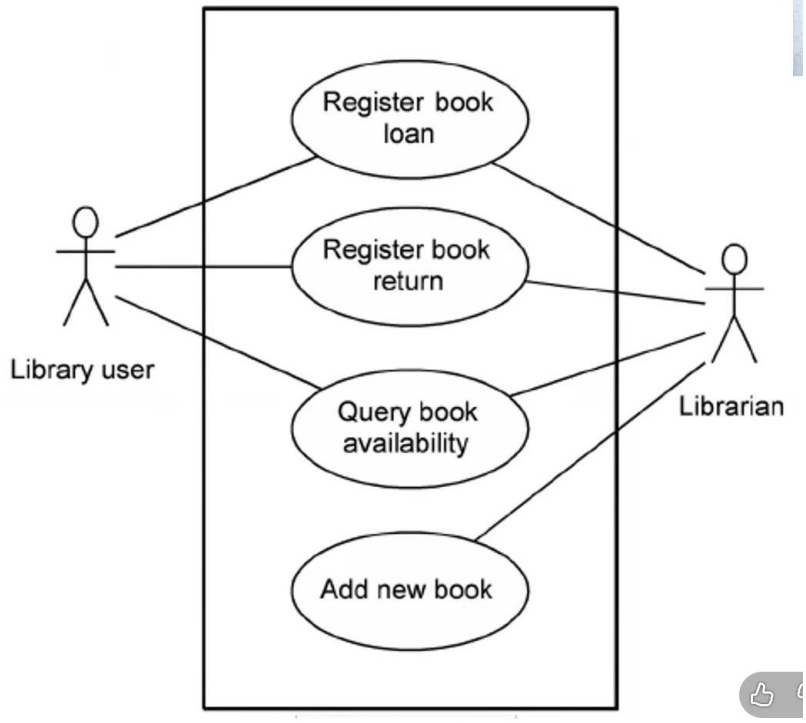
Use case, Test Scenario & Test Case

- Use Case:
  - Use case describes the requirement.
  - Use case contains THREE Items.
    - **Actor**, which is the user, which can be a single person or a group of people, interacting with a process.
    - **Action**, which is to reach the final outcome
    - **Goal/Outcome**, which is the successful user outcome.

- Test Scenario:
  - A possible area to be tested (What to test)

- Test Case:
  - Step by step actions to be performed to validate functionality of AUT (How to test).
  - Test case contains test steps, expected result & actual result.

Sample Use Case



Use Case V/s Test Case

———

Use Case - Describes functional requirement, prepared by Business Analyst(BA).
Test Case - Describes Test Steps/ Procedure, prepared by Test Engineer.

Test Scenario V/s Test Case

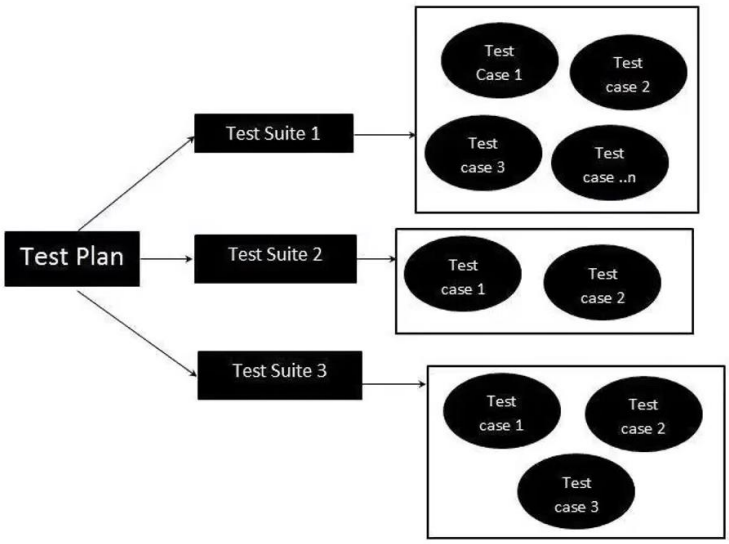
———

Test Scenario is 'What to be tested' and Test Case is 'How to be tested'.
Example:-
Test Scenario: Checking the functionality of Login button

TC1: Click the button without entering user name and password.
TC2: Click the button only entering User name.
TC3: Click the button while entering wrong user name and wrong password.

Test Suite

- **Test** Suite is group of test cases which belongs to same category.



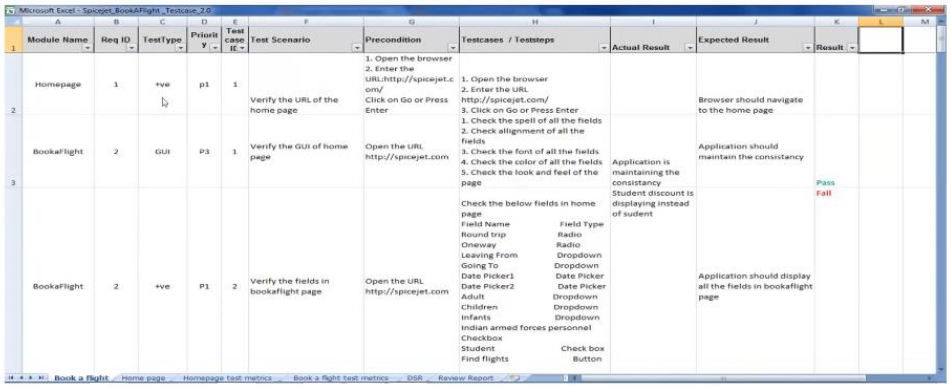
What is Test case?

A Test Case is a set of actions executed to validate particular feature or functionality of your software application.

Test Case Contents

- Test Case ID
- Test Case Title
- Description
- Pre-condition
- Priority (P0, P1,P2,P3) – order
- Requirement ID
- Steps/Actions
- Expected Result
- Actual Result
- Test data

Test Case Template



	A	B	C	D	E	F	G	H	I	J	K	L	M
1	Module Name	Req ID	TestType	Priority	Test case ID	Test Scenario	Precondition	Testcases / Teststeps	Actual Result	Expected Result	Result		
2	Homepage	1	+ve	p1	1	Verify the URL of the home page	1. Open the browser 2. Enter the URL:http://spicejet.com/ Click on Go or Press Enter	1. Open the browser 2. Enter the URL http://spicejet.com/ 3. Click on Go or Press Enter		Browser should navigate to the home page			
3	BookaFlight	2	GUI	P3	1	Verify the GUI of home page	Open the URL http://spicejet.com	1. Check the spell of all the fields 2. Check allignment of all the fields 3. Check the font of all the fields 4. Check the color of all the fields 5. Check the look and feel of the page	Application is maintaining the consistancy	Application should maintain the consistancy	Pass		
	BookaFlight	2	+ve	P1	2	Verify the fields in bookaflight page	Open the URL http://spicejet.com	Check the below fields in home page Field Name — Field Type Round trip — Radio Oneway — Radio Leaving From — Dropdown Going To — Dropdown Date Picker1 — Date Picker Date Picker2 — Date Picker Adult — Dropdown Children — Dropdown Infants — Dropdown Indian armed forces personnel Checkbox Student — Check box Find flights — Button	Student discount is displaying instead of sudent	Application should display all the fields in bookaflight page	Fail		

Book a Flight | Home page | Homepage test metrics | Book a flight test metrics | DSR | Review Report

Requirement Traceability Matrix(RTM)

- What is RTM (Requirement Traceability Matrix)?
- RTM describes the mapping of Requirement's with the Test cases.
- The main purpose of RTM is to see that all test cases are covered so that no functionality should miss while doing Software testing.
- Requirement Traceability Matrix – Parameters include
  - Requirement ID
  - Req Description
  - Test case ID's

Sample RTM

Req No	Req Desc	Testcase ID	Status
123	Login to the application	TC01,TC02,TC03	TC01-Pass TC02-Pass
345	Ticket Creation	TC04,TC05,TC06, TC07,TC08,TC09 TC010	TC04-Pass TC05-Pass TC06-Pass TC06-Fail TC07-No Run
456	Search Ticket	TC011,TC012, TC013,TC014	TC011-Pass TC012-Fail TC013-Pass TC014-No Run

Test Environment

- Test Environment is a platform specially build for test case execution on the software product.
- It is created by integrating the required software and hardware along with proper network configurations.
- Test environment simulates production/real time environment.
- Another name of test environment is Test Bed.

Test Execution

- During this phase test team will carry out the testing based on the test plans and the test cases prepared.
- **Entry Criteria:** Test cases , Test Data & Test Plan
- **Activities:**
  - Test cases are executed based on the test planning.
  - Status of test cases are marked, like Passed, Failed, Blocked, Run, and others.
  - Documentation of test results and log defects for failed cases is done.
  - All the blocked and failed test cases are assigned bug ids.
  - Retesting once the defects are fixed.
  - Defects are tracked till closure.
- **Deliverables:** Provides defect and test case execution report with completed results.

Guidelines for Test Execution

- The Build being deployed to the QA environment is the most important part of the test execution cycle.
- Test execution is done in Quality Assurance (QA) environment.
- Test execution happens in multiple cycles.
- Test execution phase consists Executing the test cases + test scripts(if automation).

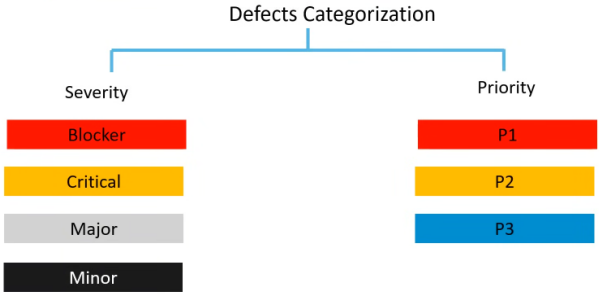
Defects/Bugs

- Any mismatched functionality found in a application is called as Defect/Bug/Issue.
- During Test Execution Test engineers are reporting mismatches as defects to developers through templates or using tools.
- Defect Reporting Tools:
  - Clear Quest
  - DevTrack
  - Jira
  - Quality Center
  - Bug Jilla etc.

Defect Report Contents

- **Defect_ID** - Unique identification number for the defect.
- **Defect Description** - Detailed description of the defect including information about the module in which defect was found.
- **Version** - Version of the application in which defect was found.
- **Steps** - Detailed steps along with screenshots with which the developer can reproduce the defects.
- **Date Raised** - Date when the defect is raised
- **Reference-** where you Provide reference to the documents like . requirements, design, architecture or may be even screenshots of the error to help understand the defect
- **Detected By** - Name/ID of the tester who raised the defect
- **Status** - Status of the defect , more on this later
- **Fixed by** - Name/ID of the developer who fixed it
- **Date Closed** - Date when the defect is closed
- **Severity** which describes the impact of the defect on the application
- **Priority** which is related to defect fixing urgency. Severity Priority could be High/Medium/Low based on the impact urgency at which the defect should be fixed respectively

Defect Classification



Defect Severity

- Severity describes the seriousness of defect and how much impact on Business workflow.
- **Defect severity can be categorized into four class**
  - Blocker(Show stopper): This defect indicates nothing can proceed further.
    - Ex: Application crashed, Login Not worked
  - Critical : The main/basic functionality is not working. Customer business workflow is broken. They cannot proceed further.
    - Ex1: Fund transfer is not working in net banking
    - Ex2: Ordering product in ecommerce application is not working.
  - Major: It cause some undesirable behavior, but the feature/application is still functional.
    - Ex1: After sending email there is no confirm message
    - Ex2: After booking cab there is no confirmation.
  - Minor: It won't cause any major break-down of the system
    - Ex: Look and feel issues, spellings, alignments.

Defect Priority

- Priority describes the importance of defect.
- Defect Priority states the order in which a defect should be fixed.
- **Defect priority can be categorized into three class**
  - P0 (High) **:** The defect must be resolved immediately as it affects the system severely and cannot be used until it is fixed.
  - P1 (Medium): It can wait until a new versions/builds is created
  - P2 (Low): Developer can fix it in later releases.

High severity, priority and low severity, priority defects

Severity \ Priority	High	Low
High	Login is taking to the blank page.	About Us link is going to blank page.
Low	After user is logged into application, he can see Home Page. But there is spelling mistake in Home Page.	User opened contact page. Email ID has spelling mistake.

More examples…

———

- Low priority-Low severity - A spelling mistake in a page not frequently navigated by users.
- Low priority-High severity - Application crashing in some very corner case.
- High priority-Low severity - Slight change in logo color or spelling mistake in company name.
- High priority-High severity - Issue with login functionality. (user is not able to login to the application)
- High Severity- Low Priority - Web page not found when user clicks on a link (user does not visit that page generally)
- Low Priority- Low Severity - Any cosmetic or spelling issues which is within a paragraph or in the page

Defect Resolution

———

After receiving the defect report from the testing team, development team conduct a review meeting to fix defects. Then they send a Resolution Type to the testing team for further communication.

Resolution Types:-

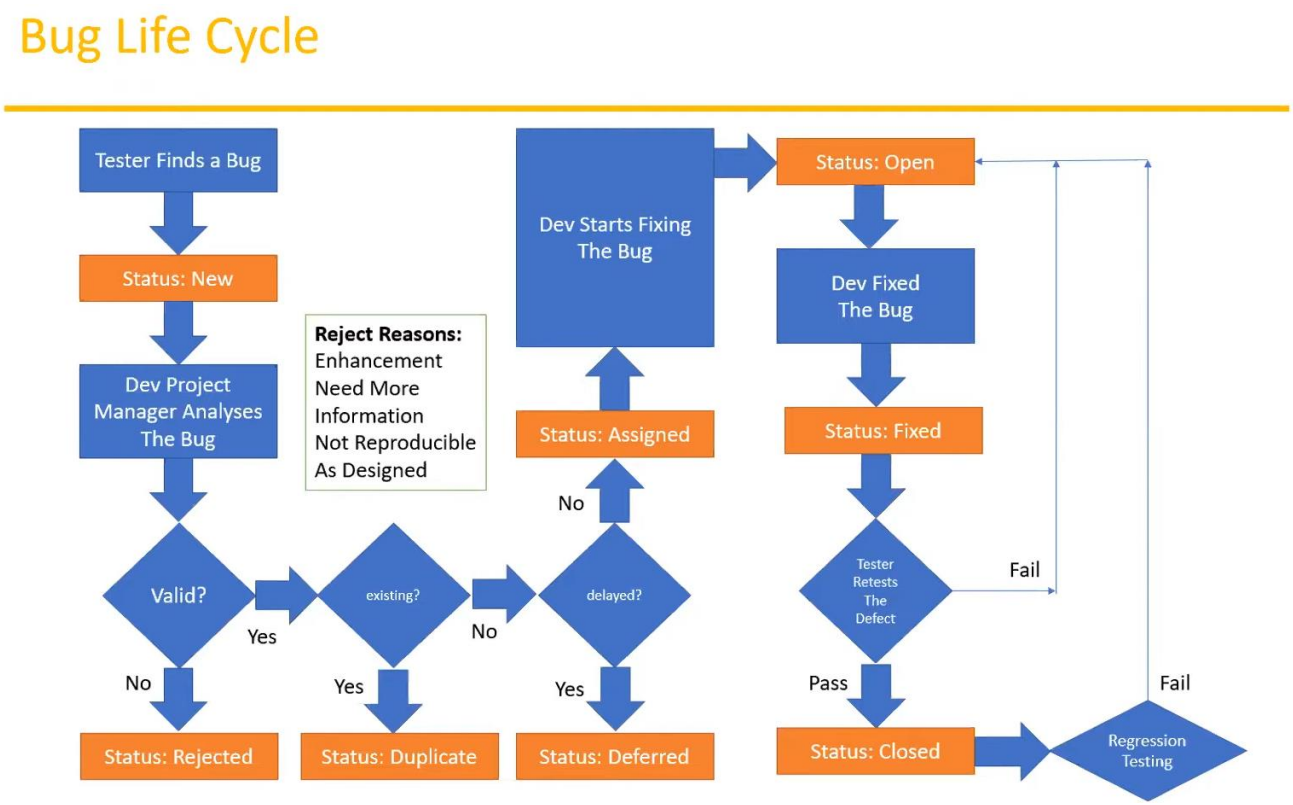
Accept
Reject
Duplicate
Enhancement
Need more information
Not Reproducible
Fixed
As Designed

Day-9

Contents Covered:

———

1. Defect/Bug Life Cycle 2. Test Closure/When To Stop Testing? 3. Software Testing Metrics 4. QA/Testing Activities 5. Principles of Software Testing



Test Cycle Closure

Activities

- Evaluate cycle completion criteria based on Time, Test coverage, Cost, Software, Critical Business Objectives, Quality
- Prepare test metrics based on the above parameters.
- Document the learning out of the project
- Prepare Test summary report
- Qualitative and quantitative reporting of quality of the work product to the customer.
- Test result analysis to find out the defect distribution by type and severity.

Deliverables

- Test Closure report
- Test metrics

Test Metrics

SNO	Required Data
1	No. Of Requirements
2	Avg. No. of Test Cases written Per Requirement
3	Total No.of Test Cases written for all Requirement
4	Total No. Of test cases Executed
5	No.of Test Cases Passed
6	No.of Test Cases Failed
7	No.of Test cases Blocked
8	No. Of Test Cases Un Executed
9	Total No. Of Defects Identified
10	Critical Defects Count
11	Higher Defects Count
12	Medium Defects Count
13	Low Defects Count
14	Customer Defects
15	No.of defects found in UAT

Test Metrics

- **% of Test cases Executed:**
  No.of Test cases executed / Total No. of Test cases written) * 100
- **% of test cases NOT executed:**
  (No.of Test cases NOT executed/Total No. of Test cases written) * 100
- **% Test cases passed**
  (No.of Test cases Passed /Total Test cases executed) * 100
- **% Test cases failed**
  (No.of Test cases failed / Total Test cases executed) * 100
- **%Test cases blocked**
  (No.of test cases blocked / Total Test cases executed) * 100

Test Metrics

- **Defect Density: Number of defects identified per requirement/s**
  No.of defects found / Size(No. of requirements)
- **Defect Removal Efficiency (DRE):**
  (A / A+B) * 100
  (Fixed Defects / (Fixed Defects + Missed defects)) * 100
  - A- Defects identified during testing/ Fixed Defects
  - B- Defects identified by the customer/Missed defects
- **Defect Leakage:**
  (No.of defects found in UAT / No. of defects found in Testing) * 100
- **Defect Rejection Ratio:**
  (No. of defect rejected /Total No. of defects raised) * 100
- **Defect Age:** Fixed date-Reported date
- **Customer satisfaction** = No.of complaints per Period of time

QA/Testing Activities

- Understanding the requirements and functional specifications of the application.
- Identifying required Test Scenario's.
- Designing Test Cases to validate application.
- Setting up Test Environment(Test Bed)
- Execute Test Cases to valid application
- Log Test results (How many test cases pass/fail).
- Defect reporting and tracking.
- Retest fixed defects of previous build
- Perform various types of testing's in application.
- Reports to Test Lead about the status of assigned tasks
- Participated in regular team meetings.
- Creating automation scripts.
- Provides recommendation on whether or not the application / system is ready for production.

---

7 Principles of Software Testing

---

1. Start software testing at early stages. Means from the beginning when you get the requirements.
2. Test the software in order to find the defects.
3. Highly impossible to give the bug free software to the customer.
4. Should not do Exhaustive testing. Means we should not use same type of data for testing every time.
5. Testing is context based. Means decide what types of testing should be conducted based on type of application.
6. We should follow the concept of Pesticide Paradox. Means, if you are executing same cases for longer run, they wont be find any defects. We have to keep update test cases in every cycle/release in order to find more defects.
7. We should follow defect clustering. Means some of the modules contains most of the defects. By experience, we can identify such risky modules. 80% of the problems are found in 20% of the modules.