Software Testing

Technologies

Java

Node JS

Python

.net

Portfolios

1. UI tester ✓

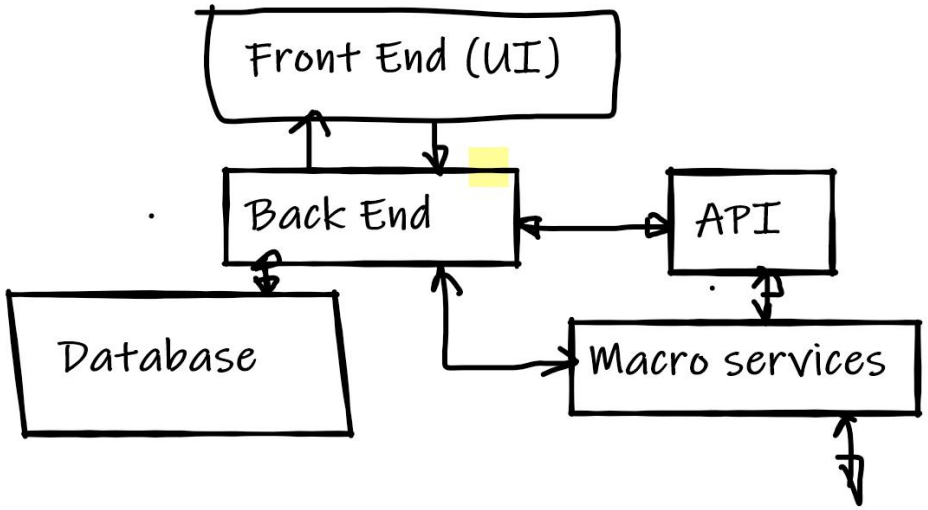
2. Functional

3. Performance

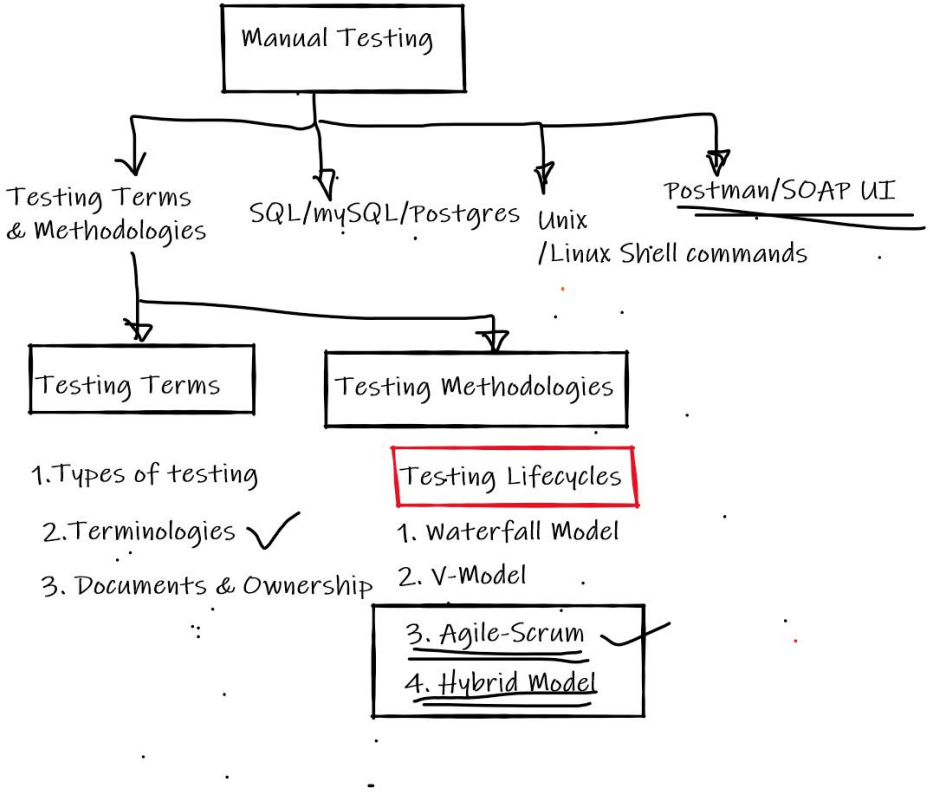
4. Database ✓

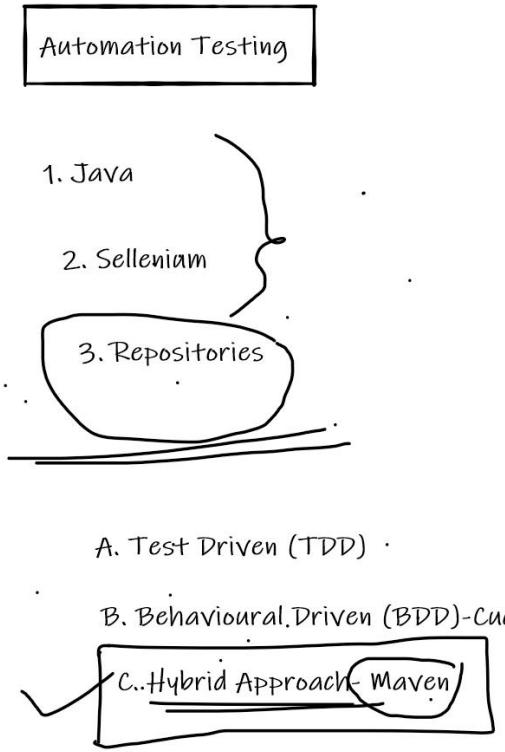
Introduction to Software Testing

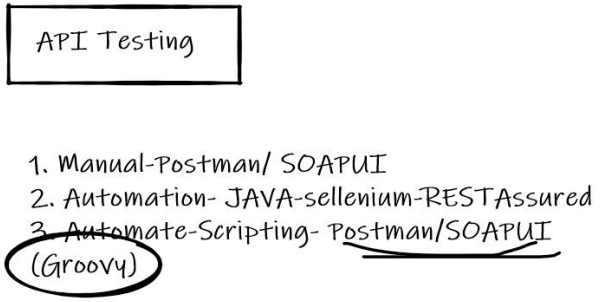
Software: instructions that tell a computer what to do that comprises the entire set of programs, procedures, and routines associated with the operation of a computer system



Software Testing







Software Testing: Checking the correctness, completeness & quality w.r.t. business requirement/functional requirement

Project

Client — Rough Requirement-Business Requirement

Business Analyst (BA) — Document-Software Requirement

Delivery Manager (DM) — Delivery Document-Reuirements chunks

Project Manager.(PM) — Managing Project Flow/Dev Flow

Facilitator

Scrum Master

Design Team

Solutioning Team

→ Software Design documnet

Developer — Develope code & UI (Integrating with DB & APIs)

1. UI-Dev
2. Backend Dev
3. Database Team

Testers (QA) — Check Functionality, performance & quality

Support/ Mainenance

Software Testing: Checking the correctness, completeness & quality w.r.t. business requirement/functional requirement

Project

Client

Rough Requirement-Business Requirement Specification (BRS)

Business Analyst (BA)

Document-Software Requirement Specification (SRS)

Delivery Manager (DM)

Delivery Document-Reuirements chunks (Product Backlog)

Project Manager.(PM)

Managing Project Flow/Dev Flow (Sprint Backlog)

Facilitator

Scrum Master

Design Team

Solutioning Team

Software Design documnet

(Software Architect)

1. High Level Design (HLD)- Hierarchial flow of functionality
2. Low Level Design (LLD)-Static logic of submodules

Developer

1. UI-Dev
2. Backend Dev
3. Database Team

Develope code & UI
(Integrating with DB & APIs)

Sprint - time to deliver part of product backlog (user stories)- Days

Product backlog-Login page
User stories- 1. User id
2. Password
3. Login button

Testers (QA)

Check Functionality, performance & quality

Support/ Mainenance

1 Sprint- 2/3weeks

Software Development Lifecycle

1. Information gathering
(Rough Requirement-->Business Requirment)

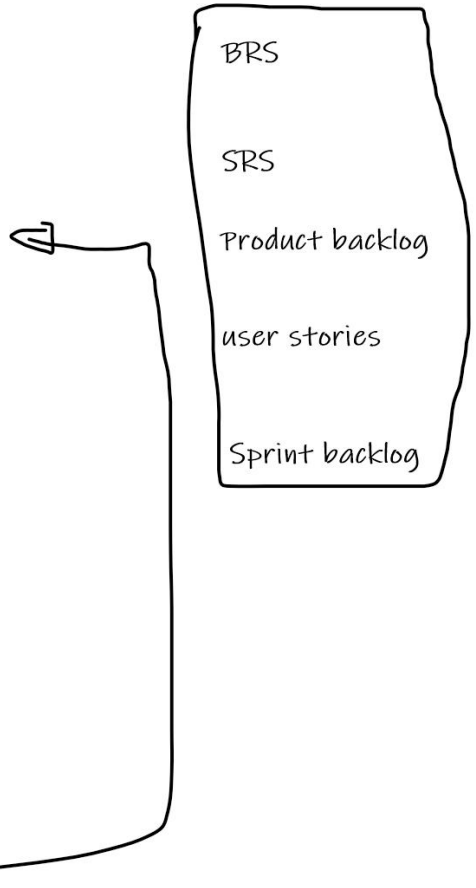
2. Analysis
(Business requirement-->Software Requirement)

3. Design
(High Level-Heirarchial functional flow
Low Level- Static Logic)

4. Coding
(Developers-Backend dev & UI dev)

5. Testing
(WBT-Unit testing, Intigration testing- Developer
BBT-Functional testing, Integration- Testers)

6. Maintenance & Support
(upgrading/reengineering/change request)



Software Quality Assurance

Continuously monitoring strength of software developed-
1. if it is meeting business requirement?
2. if it is meeting performance?
3. if it is meeting privacy?
4. if it is meeting Time to Delivery?
5. if it is cost effective?
6. Risk Management (warranty)

Software Development Lifecycle

1. Information gathering
(Rough Requirement-->Business Requirment)

2. Analysis
(Business requirement-->Software Requirement)

3. Design
(High Level-Heirarchial functional flow
Low Level- Static Logic)

4. Coding
(Developers-Backend dev & UI dev)

5. Testing
(WBT-Unit testing, Intigration testing- Developer
BBT-Functional testing, Integration-Testers)- Test Case Development & Test case execution

6. Maintenance & Support
(upgrading/reengineering/change request)

Software Quality Assurance

Continuously monitoring strength of software developed-
1. if it is meeting business requirement?
2. if it is meeting performance?
3. if it is meeting privacy?
4. if it is meeting Time to Delivery?
5. if it is cost effective?
6. Risk Management (warranty)

Software Development Lifecycle

1. Information gathering
(Rough Requirement-->Business Requirment)

2. Analysis
(Business requirement-->Software Requirement)

3. Design
(High Level-Heirarchial functional flow
Low Level- Static Logic)

4. Coding
(Developers-Backend dev & UI dev)

5. Testing
(WBT-Unit testing, Intigration testing- Developer
BBT-Functional testing, Integration-Testers)- Test Case Development & Test case execution

6. Maintenance & Support
(upgrading/reengineering/change request)

Software Quality Assurance

Continuously monitoring strength of software developed-
1. if it is meeting business requirement?
2. if it is meeting performance?
3. if it is meeting privacy?
4. if it is meeting Time to Delivery?
5. if it is cost effective?
6. Risk Management (warranty)

Software Requirement Specification:

1. Functional Requirement (description)
2. Functional Flow diagram
3. Use cases
4. User stories (description+Acceptance criteria +wireframes)
5. Snapshots (HTML pages/ Images)

Design Document: (Design team+Solutioning team+Software Arctitect)

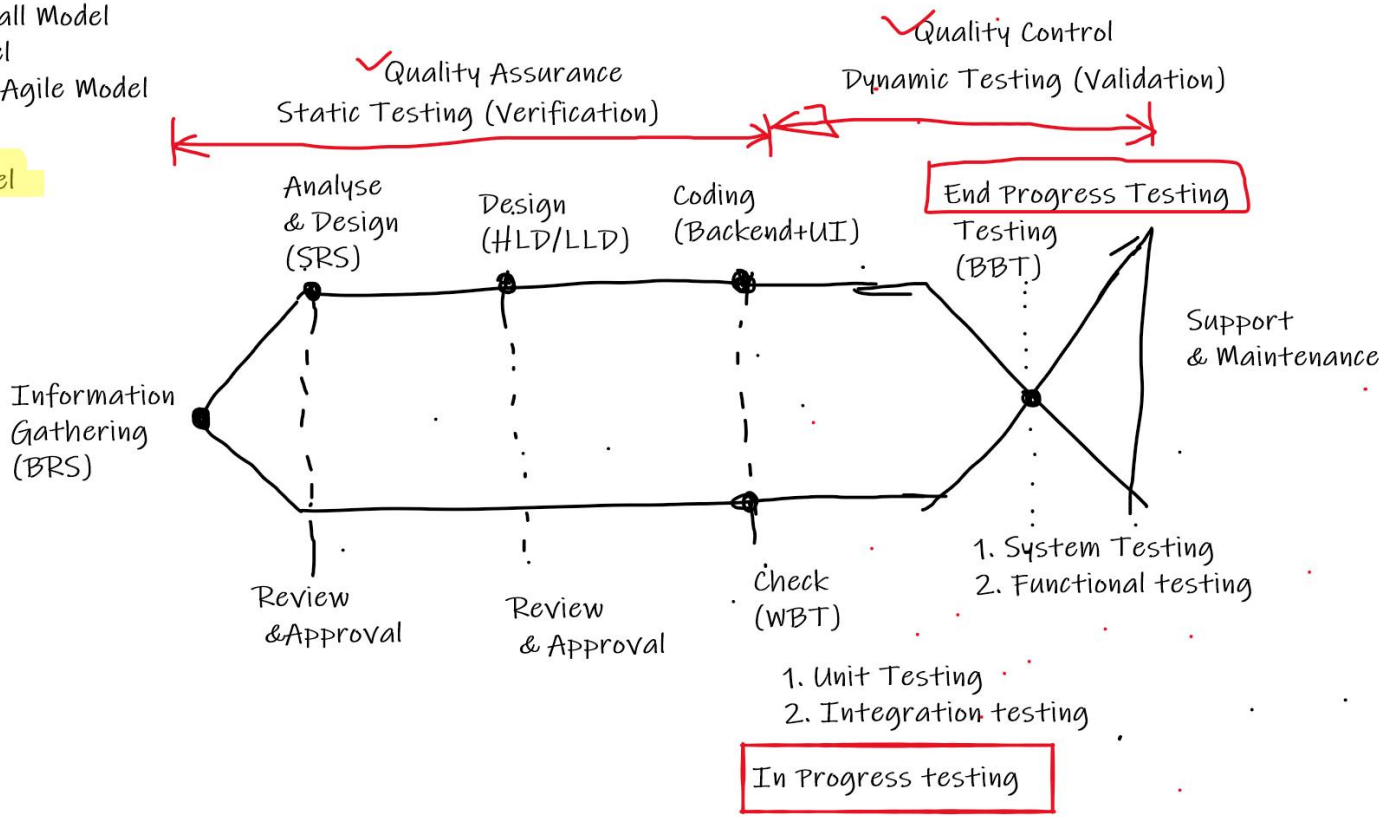
1. High Level Design Document (HLD)-
Hierarchial flow of functionality to be developed (modular design)

2. Low Level Design document (LLD)-
Static Logic to be developed (modulewise)
a. Entity-Relationship diagrams (ER- Diagram)
b. Class diagram
c. Data Flow Diagram
d. Data tables

Models of Software Developement Lifecycle

** Fish Model
1. Waterfall Model
2. V-Model
3. Scrum-Agile Model

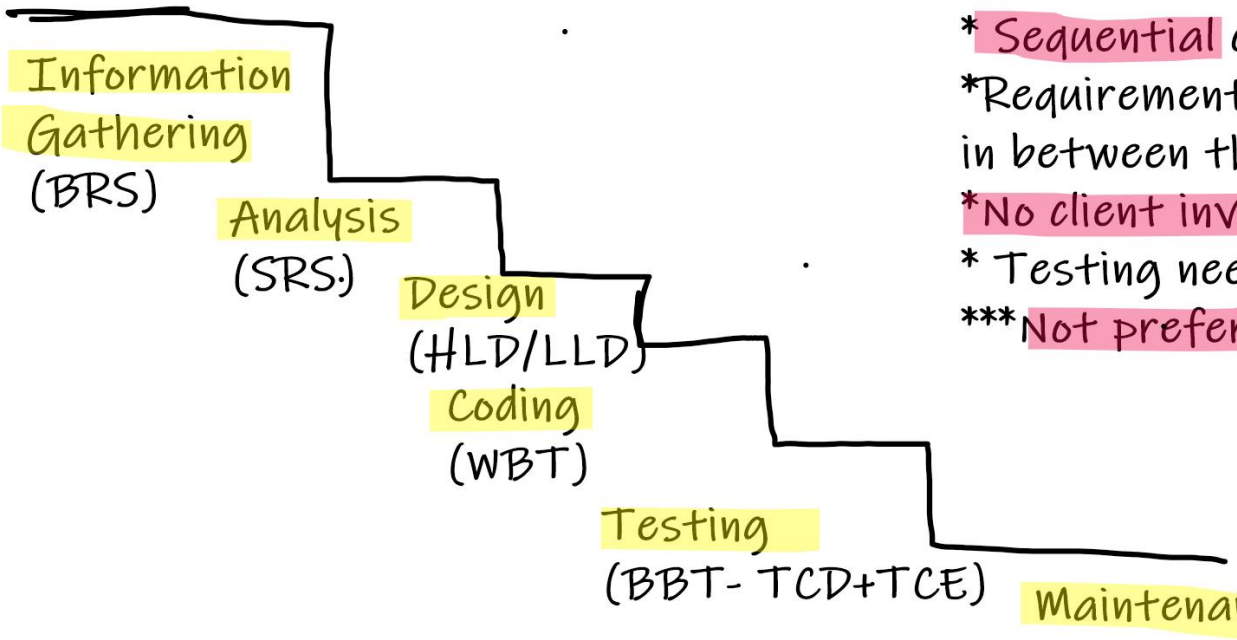
Fish Model



1. Static Testing
*Verification
*Quality assurance

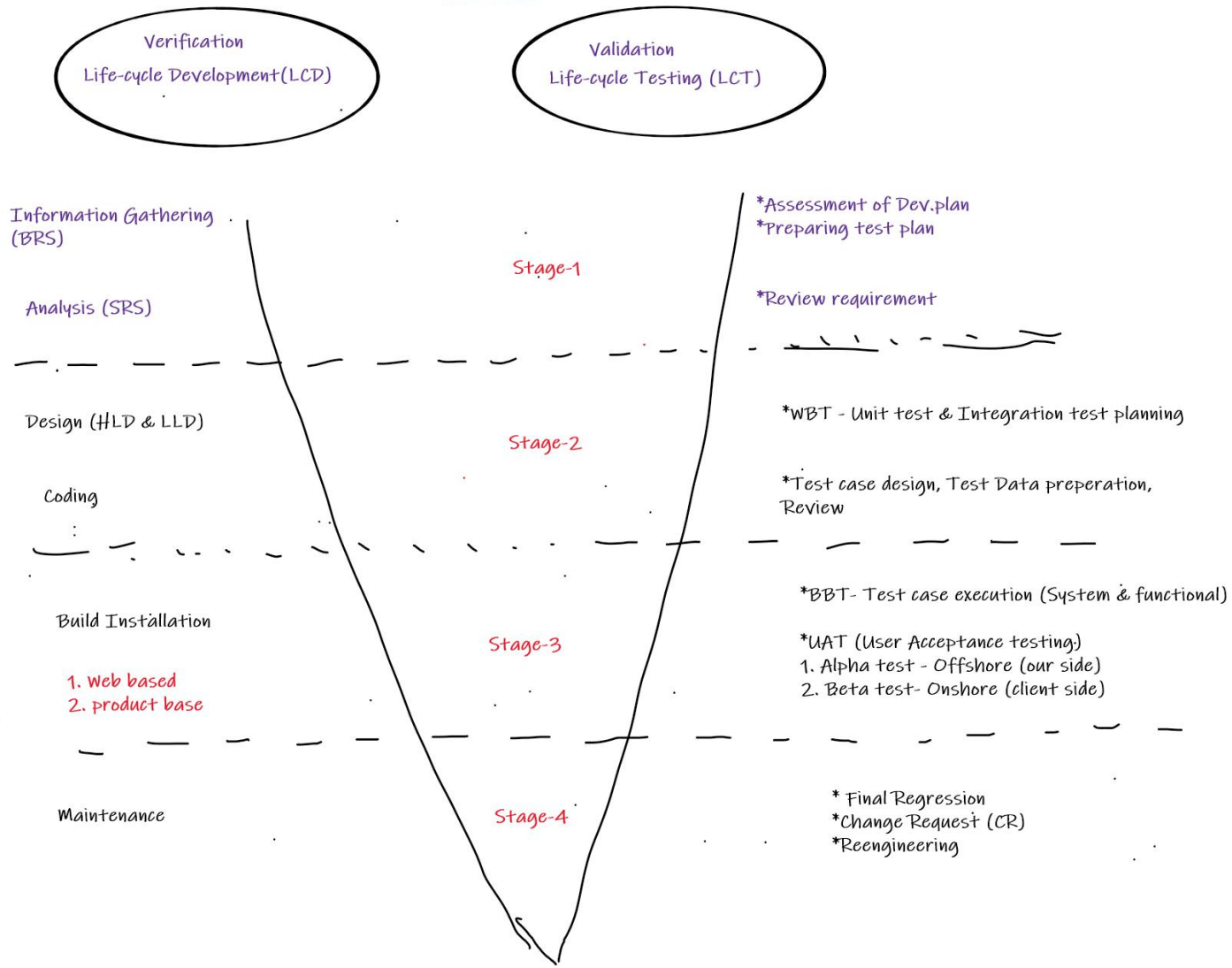
2. Dynamic Testing
*Validation
*Quality Control
a. In progress testing (WBT)
b. End Progress testing (BBT)

1. Waterfall Model



* Sequential order of development
*Requirement is fixed & cannot be change in between the process (Not Flexible)
*No client involvement in process
* Testing need to wait until coding is done
***Not prefered model

2. V-Model



Defect Removal Efficiency:

DRE= A / (A+B)

where,
A= Defect found by tester
B= Defect found by UAT tester

DRE must be --> 0.9 to 1 (& not below)

* 1-Release --> Quarterly (year)--> each 3 months
* only 4 release/year
** Less preferred model
* If defect comes in final stage of dev-cycle, we need to wait for next release (next 3 months) for fixes to come in build

3. Agile Model

AGILE methodology is a practice that promotes continuous iteration of development and testing throughout the software development lifecycle of the project. In the Agile model, both development and testing activities are concurrent, unlike the Waterfall model.

* This model is decided by Client(customer) or Organization.

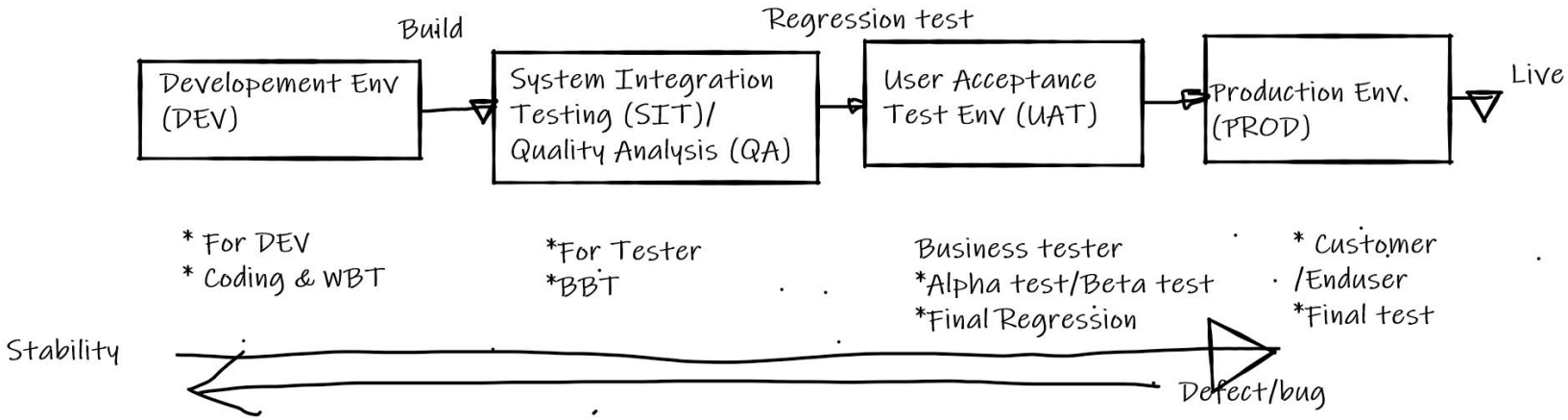
Agile Subprocess: -

1. Extream Programming (XP) --> Only programming without deep testing
2. Kanban model --> parallel coding & testing but changes are not accepted
3. Scrum model --> parallel coding & testing in iterative fashion + changes are adapted at any stage
4. Dynamic System Development Methodology (DSDM)
5. Future Driven Model (FDD)
6. Lean.

Features of Scrum agile-

* Value driven methodology (priorotizing requirment)
* 1release happenes every 2/3 weeks
* New change request has minimal impact (adaptive to CR) on dev, test & production
* Follow-up happens iteratively
* concurrent development & testing
* phasewise release

Different Environments used:



The Agile – Scrum Framework

Stakeholder

↓

Product Owner

↓

Product Backlog
*BRS
*SRS--> User stories
* Epics --> prioritiwise created modules /bunch of user stories
*Created by BA
*e.g. 100 stories

↓

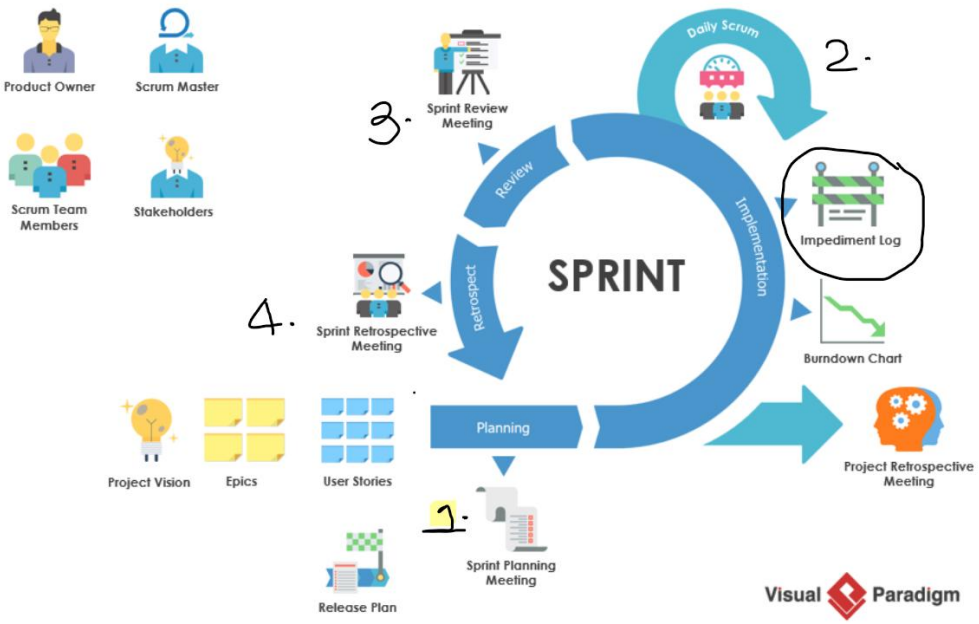
Sprint Backlog
*DM & PM decides sprints per epic
*no of stories by all team/sprint
*e.g. 10 stories/ sprint

↓

Sprint (1 sprint=1 iteration=2/3 weeks)
1. Sprint planning
*Sprint Goal
* Estimation- commitment of time to Dev & test for a story
*Story point- time to points →

100 stories--> 10 (per sprint)x10(no of sprint)x1sprint=2week

Estimation Technique-
* Planning Poaker (estimation based on fibonicci series) --> most prefered
* T-shit size
* Dot card



Reference- https://www.mountaingoatsoftware.com/agile/scrum

2. Daily Scrum/Daily Standup
* What did yesterday?
*what is today's work plan
* what are road-blocks/blocker/issues

3. Sprint Review
Demo of devoped functionality to Product Owner

4. Sprint REtrospective
* What good things done
* what things went wrong
* How things can be improved in next iteration

**Impediment Logs-
Record of issues/blocker that slows down the implementation of functionality

**Burndown chart-
Visual indication of story point consumption & thereby sprint/project progress in graphical form
Subtypes-
a. burndown
b. burnup

**Estimation:
calculated based on-
1. Complexity
2. Efforts required
3. Time required
4. Knowledge of system

*For testing estimation is given
by considering time taken for following tasks-
a. Requirement analysis--1
b. test scenario preperation (high
level scenarios based on acceptance
criteria)--2
c. test case preperation--2/3
d. test case execution--3/5

Important Terminologies of Agile:

1. Velocity --Rate at which a team progress sprint by sprint

2. Impediments--issue that slows down the speed of work
*Missing resources & seek members
*techinical, operational or organizational problem
* External issues (weather or war or pandemics)
* Lack of knowledge/skill
* Lack of management support

3. Increment -- total of all product backlogs completed during sprint

4. Build breaker -- sudden/unexpected bug/issue that may-
* stop compilation
* stop execution
* generates some warning

5. Spikes -- activity related to design / teshnical issues taken in between the sprint
* Functional spike
*Technical spike

6. Agile menifestos-
* demonstration at regular interval
* individual, cross team interactions
* client collaboration
* adapt changes at any point (CR)

7. Zero Sprint- Preperation phase of Agile
* Setting environment- VPN, Credentials (user/password), URL, Database connection, Remote connections etc. (as per Ramp-up plan document)
*Preparing backlogs
* Exploring Application

8. Definition of Done (DoD)- if application is developed, tested, Domoed & released (from one env to another) then it is DoD.

9. DONE-- Individual task is completed & reviewed

Epic
Story
subtasks

Advantages of Srum-Agile

1. Delivery/Release time is within 2/3 weeks
2. Regular follow-up of work & related issues increases productivity
3. Automation for existing functionalities is possible that-
* reduces time to test
*reduces human efforts & errors
*reduces resouces required
* reduced delivery time
4. Check-point is available for each module
5. Self organized working

Drawbacks of Scrum Agile

1. For single application if multiple clients tries to use it at a time, application becomes slow/ stucks
2. Difficult to predict efforts
3. If requirement is not understood/ misunderstood, may leads to customer disatisfaction
4. Only experience people get chance to take important decision

1-Product Owner

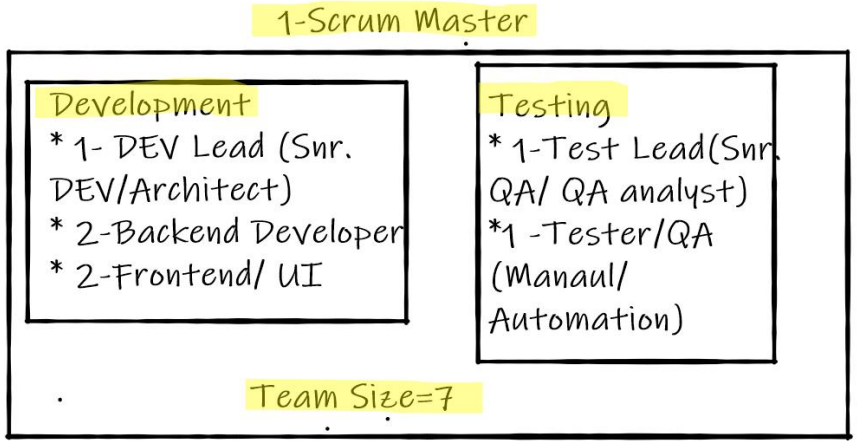
1- Delivery manager

1- BA.

1- Architect

1- Project Manager (optional)

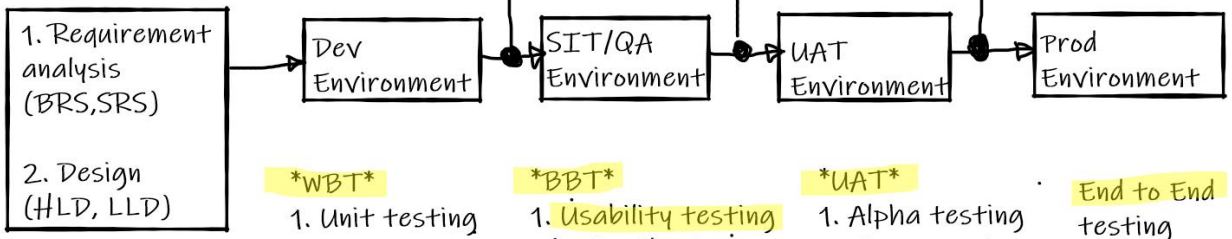
Team Structure



Team size may vary between 3 to 9

Ratio of Dev/test- 3:1 or 3:2

Testing Terms & Terminologies



1. Smoke test (for each new build)
2. Sanity test (superfacial test of each functionality)

1. Regression test (thorough check of functionality)
2. Retesting (check with repeated data)

Final Regression (Critical check before release)

WBT
1. Unit testing
2. Integration testing

BBT
1. Usability testing
--a. GUI/ UI test
--b. Manual support test (context sensitivity)
2. Functional testing
--a. Functionality testing (BIEBSC)
--b. Non-functionality testing (RCCIISPG)
3. Security testing
4. Performance testing

UAT
1. Alpha testing
2. Beta testing

End to End testing

**White Box Testing (WBT)

* It is code level testing
* Techniques of WBT-
a. Execution testing (build, debug, sequencing & looping)
b. Operation testing (integration test over client system)
c. Mutuation testing (test with different inputs)

Types of WBT-

1. Unit Testing:
* testing based on low level design (LLD)
*only checks positive scenarios
* basic checks to perform-
--Code Coverage
-- CPU use & memory use

2. Integration Testing:
* testing based in high level design (HLD)
* Verifies code based on-
--front end "calls"
--backend "database joins"

*Approaches in integration testing:

1. TopDown approach: --> other than main module, if some other module is absent, developer creates temporary module called as "Stub" for testing

2. Bottom up approach:--> if main module itself is absent then developer creates "Driver" (XML module) to collect data from remaining sub-modules

3. Sandwitch or Hybrid approach:-->if both main module & few of sub-module are absent, then some XML-module as a placeholder for testing

Smoke testing:

*to verify if the basic functionalities of that particular build are working fine as expected or not.

* not exhaustive testing but to be executed in a group

*always be the first test to be done on any 'new' build.

* Level-0 testing (Dev & tester are involved)

Sanity testing:

* to touch each implementation and its impact but not thoroughly or in-depth

* it may include functional, UI, version, etc. testing depending on the implementation and its impact

* Sanity may be performed in following cases-

a. when deployment happens in any environment, to have a general health check

b. if team is in short of time to deliver the build

* This is a wide and shallow testing that checks the stability of the build based on-

--Core & basic functionalities

-- Link validation

-- tab validation

-- page navigation

-- UI/GUI

* This is also Level-0 testing in case-A mentioned above.

* It is subset of Regression testing.

System Integration Testing (SIT)/ Black Box Testing (BBT) / QA-testing :

* the functionalities of software applications are tested without having knowledge of internal code structure, implementation details and internal paths.

*Types of Black Box Testing (BBT):

1. Usability Testing -->

* Front End Testing (UI/GUI)- look & feel, ease of use, speed interface

*Manual Support Testing- checks the context sensitivity/promting

2. Functional Testing-

a. Functionality testing-

* Behavioural testing

*Input Domain testing- (Boundary value analysis (BVA), Equivalent class partitioning (ECP)

*Error Handling

*Backend/database testing

*Service level/service sequence test

* Calculations

b. Non-Functionality testing-

*Recovery /Reliability testing

*Compatibility/portability testing (Forword compatibility, Backward compatibility, cross browsing, version compatibility)

*Configuration /Hardware compatibility

*Intersystem tesring

*Installation testing XXX

*sanitation/ garbage testing

*Parallel testing

*Globalization (Internationalization, Localization)

3.Security Testing

4. Performace testing

*Approaches in integration testing:

1. TopDown approach: --> other than main module, if some other module is absent, developer creates temporary module called as "Stub" for testing

2. Bottom up approach:--> if main module itself is absent then developer creates "Driver" (XML module) to collect data from remaining sub-modules

3. Sandwitch or Hybrid approach:-->if both main module & few of sub-module are absent, then some XML-module as a placeholder for testing

Smoke testing:

*to verify if the basic functionalities of that particular build are working fine as expected or not.

* not exhaustive testing but to be executed in a group

*always be the first test to be done on any 'new' build.

* Level-0 testing (Dev & tester are involved)

Sanity testing:

* to touch each implementation and its impact but not thoroughly or in-depth

* it may include functional, UI, version, etc. testing depending on the implementation and its impact

* Sanity may be performed in following cases-

a. when deployment happens in any environment, to have a general health check

b. if team is in short of time to deliver the build

* This is a wide and shallow testing that checks the stability of the build based on-

--Core & basic functionalities
-- Link validation
-- tab validation
-- page navigation
-- UI/GUI

* This is also Level-0 testing in case-A mentioned above.

* It is subset of Regression testing.

System Integration Testing (SIT)/ Black Box Testing (BBT) / QA-testing :

* the functionalities of software applications are tested without having knowledge of internal code structure, implementation details and internal paths.

*Types of Black Box Testing (BBT):

1. Usability Testing -->

* Front End Testing (UI/GUI)- look & feel, ease of use, speed interface

*Manual Support Testing- checks the context sensitivity/promting

2. Functional Testing-

a. Functionality testing-

* Behavioural testing

*Input Domain testing- (Boundary value analysis (BVA), Equivalent class partitioning (ECP)

*Error Handling

*Backend/database testing

*Service level/service sequence test

* Calculations

b. Non-Functionality testing-

*Recovery /Reliability testing

*Compatibility/portability testing (Forword compatibility, Backward compatibility, cross browsing, version compatibility)

*Configuration /Hardware compatibility

*Intersystem tesring

*Installation testing XXX

*sanitation/ garbage testing

*Parallel testing

*Globalization (Internationalization, Localization)

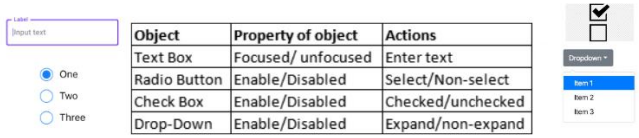
3.Security Testing

4. Performace testing

a. Functional Testing:

* performed to confirm that the functionality of an application or system is behaving as expected (as per requirement)

a. i) Behavioural Testing: cheking behaviour of object & its property



Object	Property of object	Actions
Text Box	Focused/ unfocused	Enter text
Radio Button	Enable/Disabled	Select/Non-select
Check Box	Enable/Disabled	Checked/unchecked
Drop-Down	Enable/Disabled	Expand/non-expand

a. ii) Input Domain Testing: Checking size, length & format of the object.

** Equivalent Class Partitioning Analysis--> partitioning object in to equivalent classes- like-

month values:

.... -2,-1,0, 1,2,3,4,5,6,7,8,9,10,11,12,13,14,15.....

--Invalid--|----------Valid----------|---Invalid---

min max

Characters:

-,@,#,$,%,a,b,c.....z, A,B,C.....Z,1,2,3.....

--Invalid-|----------valid----------|

**Boundary value Analysis:

testing values around boundaries like-

month values:

--at boundary invalid- 0,13

--at boundary valid- 1,12

--in valid band/ middle value- 6

Characters:

--Invalid- @,#

--Valid at boundary of lower case -a,z

--in valid band of lower case- m

--Valid at boundary of upper case -A,Z

--in valid band of upper case- P

--Valid at boundary of numbers -0,9

--in valid band of numbers-5

--invalid at boundaries- -1,-9

a.iii) Error Handling:

-- On UI when incorrect value is entered, the error message should be shown

--backend call (Method in code) to open expected page/screen when data entered in front end matches with data stored in database

a. iv) Backend Testing:

--database testing done w.r.t. input given & output stored in database

--this is done using data manipulation language (DML) or Data Definition language (DDL) in SQL

a. v) Service Level test:

--checks all services are up & running (like APIs, third party application, macro services etc. associated with main application)

--checks sequencing of pages like,

Login --> Home-->Service --> Gateway--> Acknowledgement

a. vi) Calculations:

--checks based on criterias & inputs, calculations happening correctly or not

e.g. EMI calculator application, Coupon codes (like paytm gives 50% discount on first order etc.)

b. Non-functionality testing :
* to verify the non-functional requirement of the application like Performance, Usability, etc.

b.i)Recovery Testing :
--checks the recovery from abnormal or unexpected failures
--e.g. recovery from database failure, API failures, microservice failures etc.
-- this is also called "Reliability testing"

b.ii) Compatibility Testing:
* Forword Compatibility --> Checks if application/ feature developed is OK but browser supported is not OK (less no of time bugs are present)
*Backword Compatibility--> Checks if browser supported is OK but application/ feature developed itself is not OK (more no of time bugs are present)
techniques to test-
*Cross browser testing --> check the same feature on multiple browsers (e.g. chrome, IE, firefox, Opera,Edge etc.)
*Version compatibility --> check the same feature on any browser with different versions (e.g. IE7, IE8, IE9, IE10, IE11 etc.)
* this is also called as Portability testing.

b.iii) Configuration testing:
* check developed build/application is compatible to harware or system or not?
* this is also called as Hardware compatibility testing

b.iv) Inter-system testing:
*check the working of the build/application with already linked/installed application
e.g. Flight booking for Indigo airlines from Makemytrip.com
Makemy trip-->Flight booking portal<--Indigo airline API
* Generally it is related to Web service testing

b.v) Installation testing : XXX
*testing of build/application on client PC w.r.t.
--RAM/memory space required
--running process (progress shown by progress bar & % use of system)
--disk space require to hold the application

b.vi) Sanitation testing:
* test unwanted/ unexpected extra part that is developed which was not mentioned in requirement.
*this is also called as "Garbage testing"

b.vii) Parallel testing: XXX
* Checking of same application with different versions running on same machine with same inputs
* it is a check of intactness of core functionality with every upgrade
*e.g. whatsapp old version & new version on same andrioid phone should be able to send/recieve same message

b. viii) Globalization:
*checks support of multiple language, zipcodes, extentions etc.
*Internationalization -->Support to national languages, Zip/postal codes & extentions like-
--Google support to English, Hindi, Chinese, French, Urdu etc.
--Google supports to international postal/zip codes
--Google supports to phone extentions like +40, +91, +92, +41 etc.
*Localization -->Supports to local languages, Zip codes, states etc.
--Google supports to indian languages like Hindi, Marathi, tamil, telagu, gujrathi etc.
--Google supports to all states & union territories of india
--Google supports to village/city wise postal code in a state

Performance testing:

*testing the Speed, Scalability & Stability of a software application under particular workload
--Speed - Determines whether the application responds quickly
--Scalability - Determines maximum user load the software application can handle.
--Stability - Determines if the application is stable under varying loads

**Types of Performance Testing -->

i)Load testing (under higher no. of users)
ii)Stress testing (under high data traffic)
iii)Endurance testing (if handles load for long time)
iv)Spike testing (subset of load test)
v)Volume testing (w.r.t. database)
vi) Scalability testing (check the support if system can scaling up the resources to handle load)

Security Testing:

*testing that uncovers vulnerabilities, threats, risks in a software application and prevents malicious attacks from intruders
*to identify all possible loopholes and weaknesses of the software system which might result in a loss of information, revenue & repute

*Types of Security Testing:

i)Vulnerability Scanning (scan a system against known vulnerability signatures)
ii)Security Scanning (identifying network and system weaknesses)
iii)Penetration testing (check for potential vulnerabilities to an external hacking attempt)
iv)Risk Assessment (nalysis of security risks observed in the organization)
v)Security Auditing (inspection of Applications and Operating systems for security flaws)
vi)Ethical hacking (hacking an Organization Software systems)
vii)Posture Assessment (combines Security scanning, Ethical Hacking and Risk Assessments)

Retesting:

*Running previously failed test case again on new build with multiple test data to check if defect logged earlier is fixed or not
*Retesting happens in following case-
i) if defect logged by tester is fixed
ii) if defect logged is rejected by developer
iii) if client asked to retest certain functionality to check its quiality

Regression Testing:

*Testing an application as a whole for the modification in any module or functionality is termed as Regression Testing.

*Regression testing is carried in following cases-
i) when new functionality is added to existing application
ii) when new change request (CR) is added
iii) when there is defect fix
iv) when there is performance issue is fixed (e.g. application is loading on browser in 2Sec instead of 5Sec)
v) When there is environment change (Database changed from Oracle to mySQL)

*Regression suit contains-
--failed test cases
--test cases related to newly added scenarios
--test cases covering core & basic functionality

Performance testing:

*testing the Speed, Scalability & Stability of a software application under particular workload
--Speed - Determines whether the application responds quickly
--Scalability - Determines maximum user load the software application can handle.
--Stability - Determines if the application is stable under varying loads

**Types of Performance Testing -->

i)Load testing (under higher no. of users)
ii)Stress testing (under high data traffic)
iii)Endurance testing (if handles load for long time)
iv)Spike testing (subset of load test)
v)Volume testing (w.r.t. database)
vi) Scalability testing (check the support if system can scaling up the resources to handle load)

Security Testing:

*testing that uncovers vulnerabilities, threats, risks in a software application and prevents malicious attacks from intruders
*to identify all possible loopholes and weaknesses of the software system which might result in a loss of information, revenue & repute

*Types of Security Testing:

i)Vulnerability Scanning (scan a system against known vulnerability signatures)
ii)Security Scanning (identifying network and system weaknesses)
iii)Penetration testing (check for potential vulnerabilities to an external hacking attempt)
iv)Risk Assessment (nalysis of security risks observed in the organization)
v)Security Auditing (inspection of Applications and Operating systems for security flaws)
vi)Ethical hacking (hacking an Organization Software systems)
vii)Posture Assessment (combines Security scanning, Ethical Hacking and Risk Assessments)

Retesting:

*Running previously failed test case again on new build with multiple test data to check if defect logged earlier is fixed or not
*Retesting happens in following case-
i) if defect logged by tester is fixed
ii) if defect logged is rejected by developer
iii) if client asked to retest certain functionality to check its quiality

Regression Testing:

*Testing an application as a whole for the modification in any module or functionality is termed as Regression Testing.

*Regression testing is carried in following cases-
i) when new functionality is added to existing application
ii) when new change request (CR) is added
iii) when there is defect fix
iv) when there is performance issue is fixed (e.g. application is loading on browser in 2Sec instead of 5Sec)
v) When there is environment change (Database changed from Oracle to mySQL)

*Regression suit contains-
--failed test cases
--test cases related to newly added scenarios
--test cases covering core & basic functionality

User Acceptance Testing (UAT):

* testing performed by the end user or the client
*to verify/accept the software system before moving the software application to the production environment
*to validate end to end business flow

* UAT is carried out in a separate testing environment (Pre-PROD) with production-like data setup
*It is also called as Customer Acceptance Testing/ User feedback testing
* types of UAT:
i) Alpha Testing -->
--Inhouse testing
--both DEV & QA team is involved
-- defect/problems are been fixed immediately
--Applicable for Web based (Service based) application (e.g. amazon.com)

ii) Beta Testing-->
--Onsite testing/end user testing
-- Client &/ end users are involved (No Developer & no testers)
-- problem is been fixed in next version of application (not immediately)
-- Applicable for Product based applications (e.g. Whatsapp)

Final Regression Testing:
* Before build goes live (to the customer/user) it is critically checked that the build that hasn't changed for a period of time w.r.t.
i) defect fixes done over the time
ii) newly added scenarios/requirements
iii) core & basic functionalities
iv) dependencies if any
*Test cases to prepare final regression suit are taken from UAT test & QA/SIT test

* if some defect leaked is found at this stage, the immediate fix of the defect on same day (within stipulated time) is carried, called as "Hot Fixes"
* defect found at this stage goes through following rigerous stages before Hot Fix,
--Analysis (Reason?)
--Impact (area affected directly/indirectly)
--Coding (quick & simple wayout)
-- testing

Additional Testing terms

Monkey Testing:

*the user tests the application by providing random inputs and checking the behavior (or try to crash the application).
* does not follow any predefined test cases or strategy
*this technique is well used in Stress testing/ Load testing
*Can identify some out of the box errors.
*Easy to set up and execute with "not so skilled" resources.
*Can identify bugs which may have a higher impact
*Here tester has some knowlege of application

Gorilla Testing:

*one module or the functionality in the module is tested thoroughly and heavily.
*to check the robustness of the application
*performed by any of testers or developers

Ad-Hoc testing:

*testing is performed on an Ad-hoc basis i.e. with no reference to the test case and also without any plan or documentation in place
*tester has adequate knowledge of application

Exploratory Testing:

*to explore the application and looking for defects that exist in the application
*keep a track of what flow you have tested and what activity you did before the start
*tester has minimal knowledge of application
* Exploratory testing has following advantages-
--Handson approach
--minimum planning & maximum execution
--no formal documentation required
--just informal notes are maintained

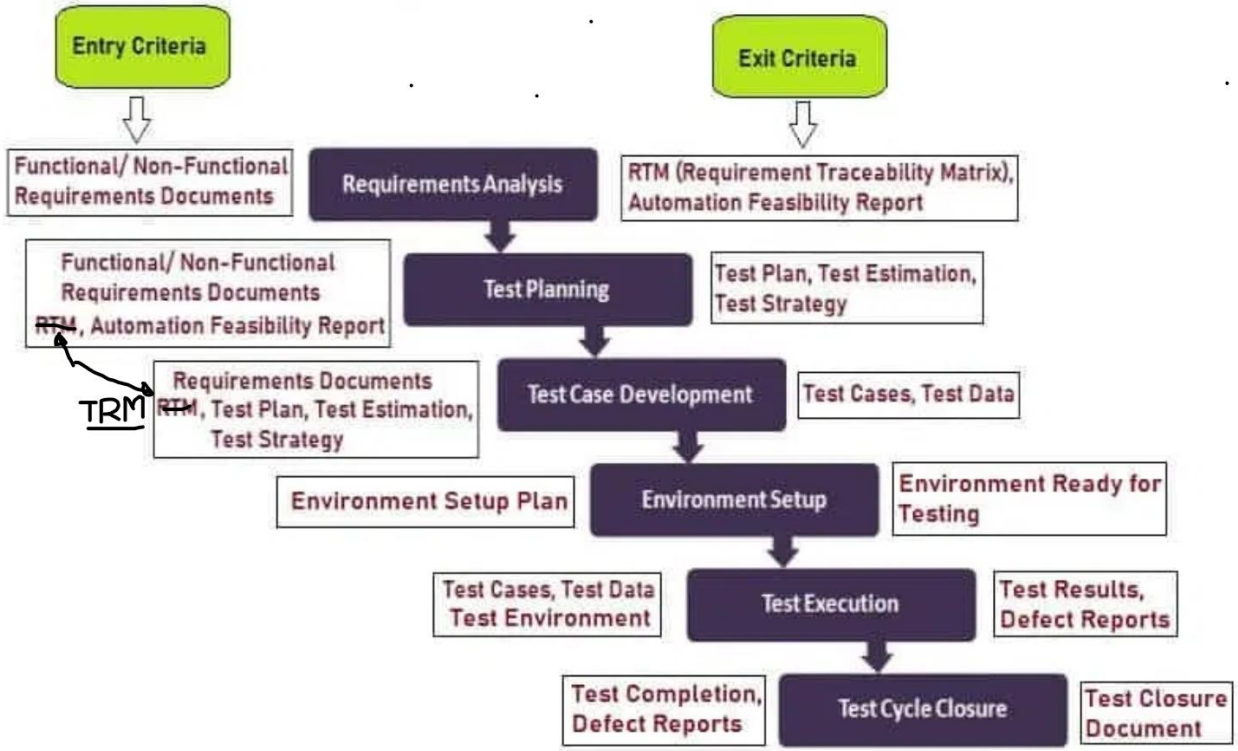
Happy Path Testing:

*to test an application successfully on a positive flow & not look for negative or error conditions.

Software Testing Life-Cycle (STLC)

Software Testing Life Cycle (STLC) is a sequence of specific activities (Verification and validation) conducted during the testing process to ensure software quality goals are met

*Entry Criteria :
--specific conditions or all those documents which are required to start a particular phase of STLC (Software Testing Life Cycle) is called "Entry Criteria"
--e.g. to start the Test Cases development phase following criteria should be met-
#The requirement document should be available.
#Complete understanding of the application flow is required.
#The Test Plan Document should be ready.
*Exit Criteria:
-- items/documents/actions/tasks that must be completed before concluding the current phase and moving on to the next phase
--e.g. to conclude the Test Cases development phase following things should be compleated-
#Test Cases should be written and reviewed.
#Test Data should be identified and ready.
#Test automation script should be ready if applicable.



1. Requirement Analysis:

*team studies the requirements from a testing point of view to identify testable requirements
*interact with various stakeholders to understand requirements in detail

Activities in Requirement Phase Testing

--types of tests to be performed.
--testing priorities and focus.
--Identify test environment
--Analyse Automation feasibility

Deliverables:

a. Test Responsibility Matrix (TRM)--> Matrix of DEV phases v/s Testing phases
b. Requirement Traceability Matrix (RTM)--> maps and traces user requirement with test cases
c. Automation feasibility report. (if applicable)

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

2. Test Planning:

*determines the test plan strategy along with efforts and cost estimates for the project

Activities in Test Planning phase

-Test tool selection (Zephyr, TestLodge, TestLink, Sellenium, Jmeter, HPALM etc.)
--Test estimation (resouces complexity, efforts)
--Resource planning (Testers, Env, DBs etc.)
-- determining roles and responsibilities. (Testing, reporting, client connect etc.)
--Training requirement

Deliverables:

a. Test Plan (prepared by test lead)
--Sprint (Iteration) ID
--Tasks (items)
--Features to be tested & not to be tested
--PASS & FAIL criterias
--Blocker defect, Suspension criteria
--Test deliverables (no. of Test cases)
--Roles & Responsibilities
--Approvals & Signatures
b. Test Estimation Document
total efforts required are calculated based on available resources, complexity & weightage

	Weightage	# of Function Points	Total
Complex	5	3	15
Medium	3	5	15
Simple	1	4	4
Function Total Points			**34**
Estimate define per point			**5**
Total Estimated Effort (Person Hours)			**170**

3. Test Case Development:

*creation, verification and rework of test cases & test scripts

Activities in Test Case Development Phase:

--Identify Test Scenarios (from description, acceptance criteria & wireframes in story)
--Review & Approval of test scenarios (Self review, Peer Review, Internal Review, External review & approval)
--Create test cases/ Automation script
--Review Test cases/Script (Peer Review)
--Create Test data (prerequisite for tests)

Deliverables:

1. Test Scenario document
Story ID | Scenario ID | Test Scenario details
2. Test Scenario Review emails/records
3. Test case document
--Scenario ID (e.g. TS_001)
--Test case ID (e.g. TC_01)
--Test Case detail (case statement)
--Prerequisits (Reference, URLs etc.)
--Test data (Credentials, input data etc.)
--Test steps (steps to execute)
-- Expected Result (expected outcome of each step)
--Priority (High, Medium, Low)
-- Test Method (Manual, Automation)
--Test case type (Unit test, Usability,Functional, Performance, Regression, UAT, Intregration etc.)
4. Test case review document
5. Test data mapping document

4. Environmental Setup

*decides the software and hardware conditions under which application is to be tested
*tester need to check readiness of the environment by smoke testing

Activities in Environmental setup phase:

-- Understand the required architecture, environment set-up
--prepare hardware and software requirement list for the Test Environment.
--Setup test Environment and test data
Perform smoke test on the build

Deliverables:

a.Environment ready with test data set up
b.Smoke Test Results.

5. Test Case Execution

*testing of the software build is done based on test plans and test cases prepared
*consists of test script execution, test script maintenance, defect reporting & retesting

Activities in tets case execution phase:

--Execute tests as per plan
--log defects for failed cases
--Map defects to test cases in RTM
--Retest the Defect fixes
--Track the defects to closure

Deliverables:

a. Completed RTM with the execution status
b. Test cases updated with results .
c. Test case execution report
d, Defect reports

Testcase:
TODO--> InProgress--> PASS,FAIL, BLOCKED, UNEXECUTED --> DONE

6. Test Cycle Closure/ Reporting

* preparing completion reporting, collection of test completion matrices and test results.
*taking collaborative decisions about problems faced & future strategies of testing

Activities in test cycle closure:

--Prepare test metrics
--Prepare Test closure report
--Test result analysis to find out the defect distribution by type and severity

Deliverables:

a. Test Summary report
b. Test Closure report
c. Test metrics

Templates & Reports

Test Scenario template:

Project Abbreviated Name:	<Mention Project Abbreviated Name here >	Author:	<Mention name of author here>
Project Name:	<Mention project name>	Reviewed by:	<Name of reviewer>
Release Name	2021_Release-X		

Req ID	Test Scenario ID	Test Scenario Description	Test Cases Mapped to	Test Scenarios Created On	Test Scenarios Updated On
ABC1234	e.g. TS_01	<Also add input data, if any>	< Test Case id >	<date of creation>	<date of upgradation if any>

Test Case template:

Project Abbreviated Name:		<Mention Project Name >	Author:							
Project Name:			Reviewed by:							
Type of Testing:			Tested by:							
Prerequisites / Assumptions:			References (If any):							
Test Case Id	Test Description	Steps For Execution	Expected Result	Test Data	Test Case Created On	Test Case Updated On	Result	Defect ID	Execution Date	Comments

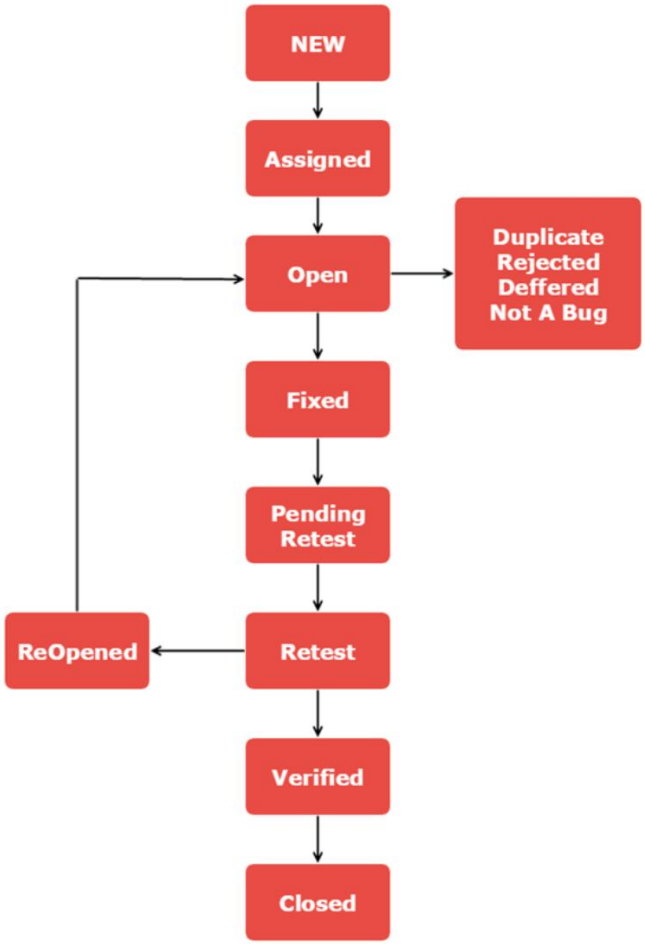
Test Execution Summary Report

Project Abbreviated Name:	< Mention Project Abbreviated Name here >
Project Name:	
Project Manager:	
Report prepared by:	
Build no.:	
Features tested:	
Hardware specifications:	
Software specifications:	
Test start date:	<dd-Mon-yy>
Test finish date:	<dd-Mon-yy>
Total No. of Test cases:	
No. of Test cases executed:	
No. of Test cases passed:	
No. of Test cases failed:	
No. of Test cases blocked:	
No. of Test cases Yet to be executed:	

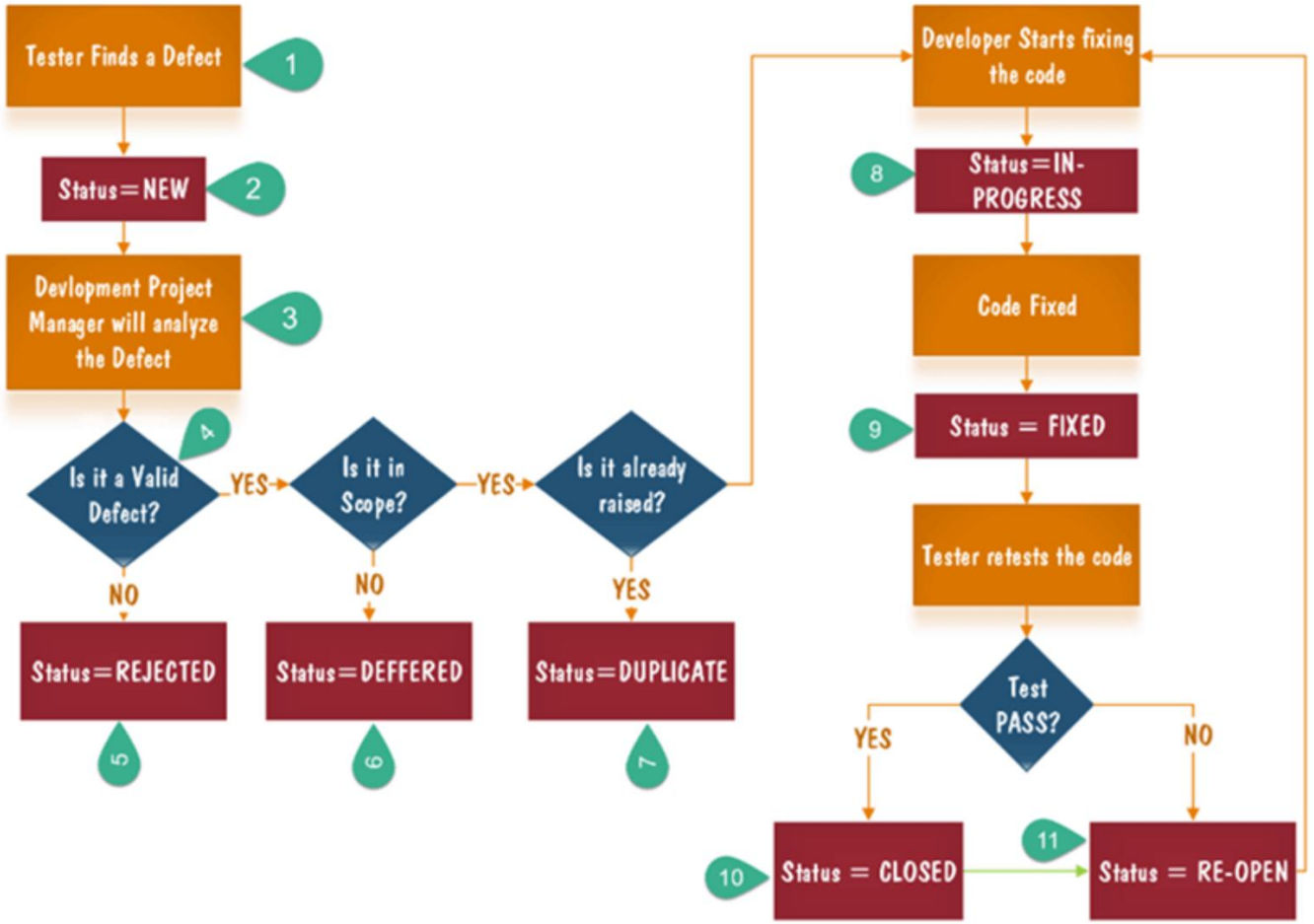
Defect Summary Report

Total No. of defects:	
Total No. of defects verified:	
No. of defects verified (Passed):	
No. of defects verified (Failed):	
No. of open defects:	

Defect/Bug Lifecycle



Actual Testing Workflow



Defect Priority:

* it is related to priority of the requirement to be built
* Defect priorities are categorized as,

3. Low: The Defect is an irritant but repair can be done once the more serious Defect has been fixed
2. Medium: During the normal course of the development activities defect should be resolved. It can wait until a new version is created
1. High: The defect must be resolved as soon as possible as it affects the system severely and cannot be used until it is fixed

Defect Severity:

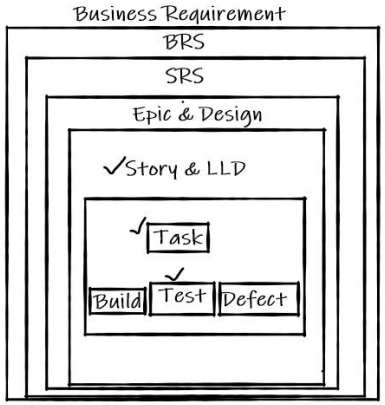
* it is related to functionality of the application/module developed
* Defect severity can be categorized as-

--Blocker --> cant proceed as dependacies are failed
--Critical --> cant test as major part failed
--High --> test is failed due to incorrect implementation
--Medium --> test is functional with undesirable behaviour
--Low --> not causing any loss of functionality (e.g. cosmatic changes required)

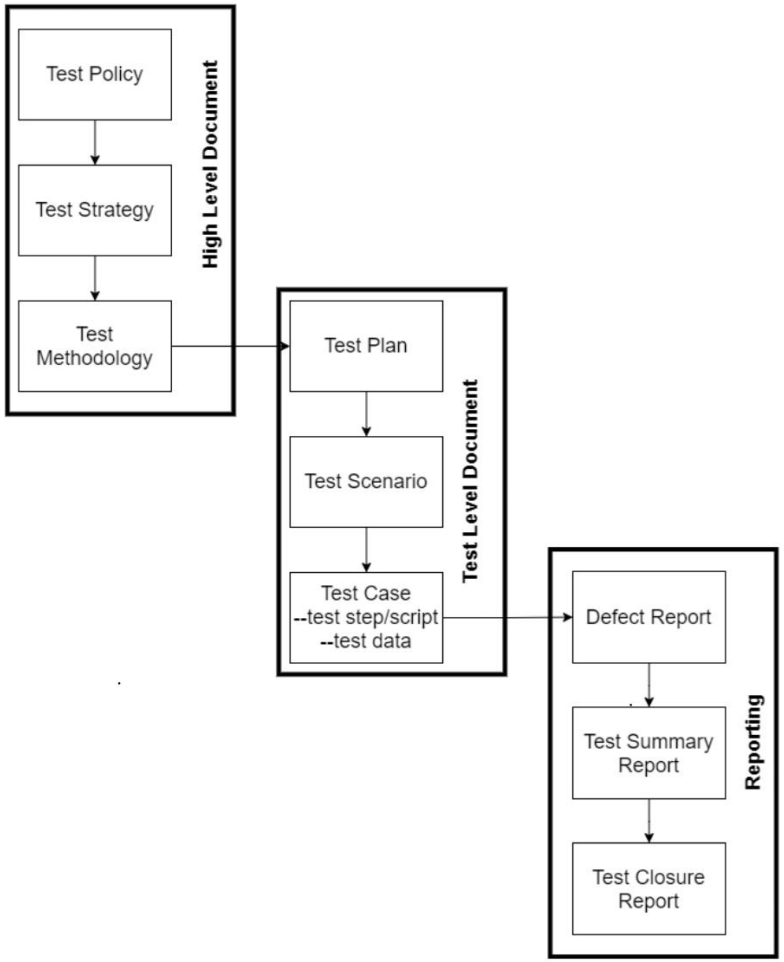
Priority V/s Severity Matrix

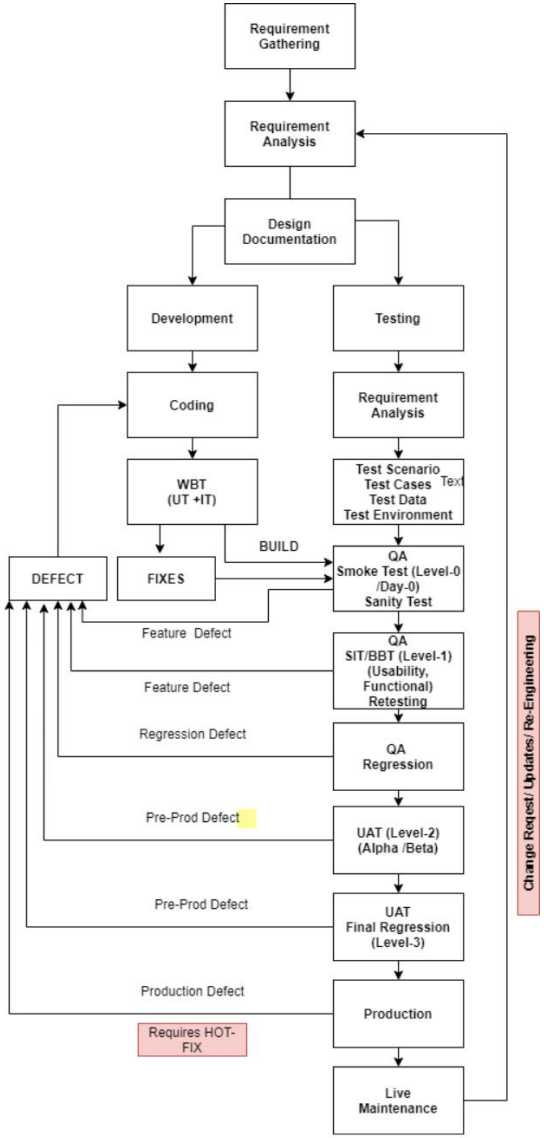
Severity → Priority ↓	H	L
H	**Critical Functionality is not working (Show Stopper/Blocker Defect)** e.g. Login page on submit not redirecting to Home page	**Required part is missing but not impacting the functionality** e.g. Company Logo is missiong on website
L	**Rarerly used functionality is not working** e.g. using promocode failure	**Mistake in development that does not affect functionality** e.g. typo in Label name of field, instead of 'Submit' button is named 'OK'

Project Level Workflow



Test Level Documentation





Reviews in Test documentation:

* Reviews are applicable to Test plan, Test Scenario & Test Cases written
* Types of Review are as follow-

1. Self Review (by own)
2. Peer Review (by Couligue)
3. Internal Review (Cross team/ by BA)
4. External Review (By client side)