MANUAL TESTING

Prepared By: Karimunnisa

Software Testing

Software:-

A set of executable programs or collection of programs in a computer is called Software.

Testing:-

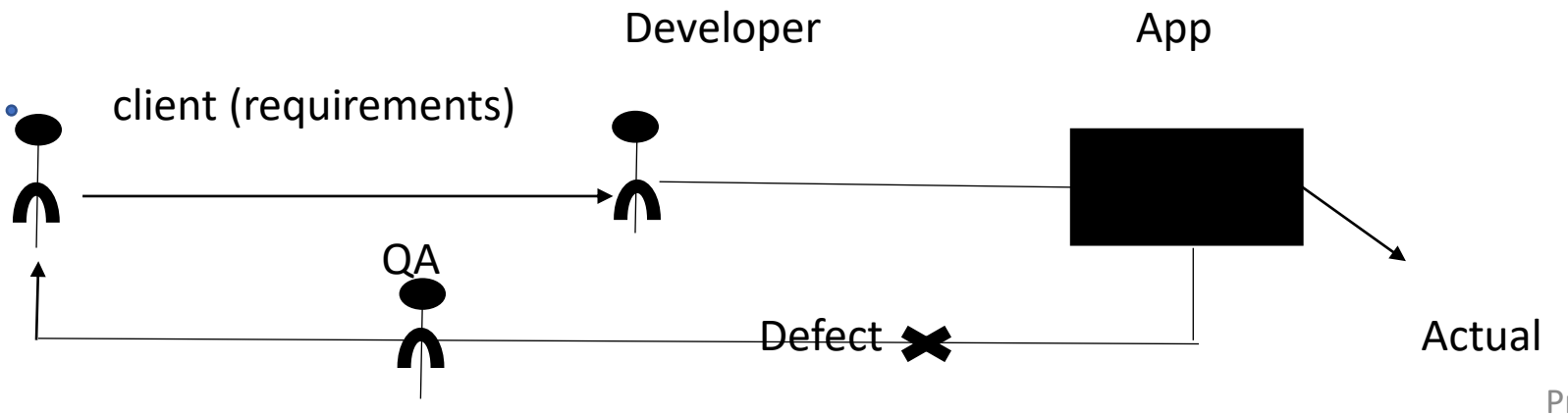
Comparison between expected values and actual values.

- Before developing the application, the client requirements are called Expected values.
- After development, the final output of the application is called Actual value.

Software Testing

Definition:

It’s a process of executing an application with an intention of finding defects.



Quality:-

Customer/Client/End user satisfaction is called Quality.

Difference between Error, Defect, Bug and Failure:-

Error:- While developing the application if programmer find out any mistake in coding part is called Error.

Defect:- During testing, if any expected value not equal with actual value is called Defect.

Bug:- Defect accepted by development team then it is called a Bug/Anomaly.

Failure:- After development and testing, during utilization if customer facing any problem is called Failure.

Difference between Project and Product:-

Project:- An application developed for specific client requirement is called Project.

Eg:- TCS, TechMahendra companies are examples for project/service based companies.

Product:- An application developed for multiple client requirements or entire market requirement is called Product.

Eg:- Gmail, Yahoo mail, Rediff mail etc..

Google, Microsoft companies are examples for product based companies.

Software Testing 2 ways:

1)Manual Testing

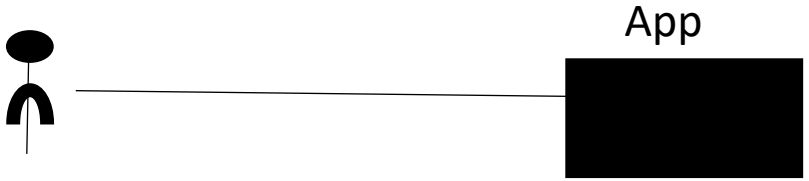
2)Automation Testing

Manual Testing:-

It’s a process to conduct testing on application by human interactions is called Manual Testing.

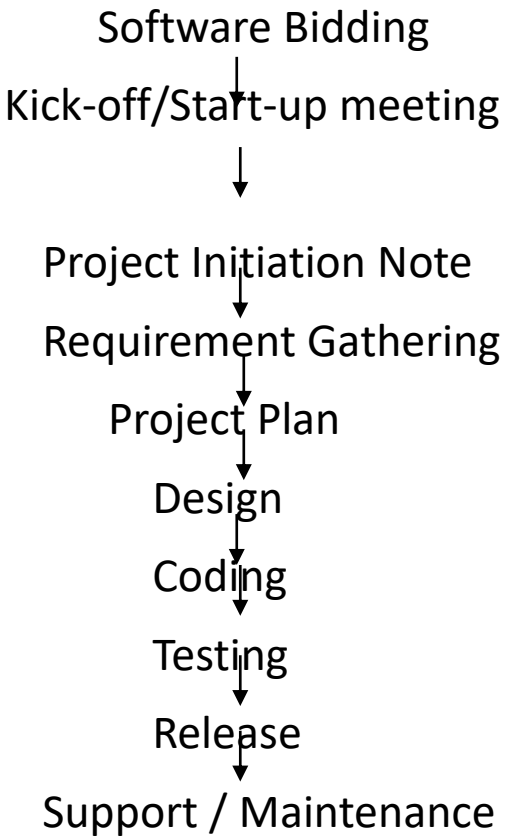
Automation Testing:-

It’s a process to conduct testing on application by the help of software tools is called Automation Testing.



SDLC

SDLC stands for Software Development Life Cycle. It will explain all the implementation activities of project development.



1. Software Bidding:-

At this phase, C.E.O (Chief Executive Office) go to client side to confirm the project and make agreement between software company and corresponding client.

2. Kick-off Meeting:-

Once project is confirmed by C.E.O, he will conduct meeting with Project Manager's to select Project Manager for current project.

Note:-

Any kind of start-up meeting can be called as Kick-off meeting.

3. Project Initiation Note:-

Once Project Manager confirmed by C.E.O, Project Manager will prepare initiation/information document called P.I.N. It contains information of client, project, domain etc.,.

4. Requirement Gathering:-

Task:- Collecting requirements of customer.

Role:- Business Analyst, System Analyst (Domain Experts)

Documents: Business Requirement Specification document (BRS)

System Requirement Specification document (SRS)/ FRS

Process:- By the help of Project Initiation Note, Business Analyst will collect requirements from client prepare as a BRS document. Based on BRS, with the help of System Analyst (SA), prepares number of SRS documents for every team. At this phase, to understand the project requirement, every software engineer can analyse SRS document only.

5. Project Plan:-

Task: Prepare plan for project implementation.

Role: Project Manager with the help of development manager.

Documents: Project plan contains development plan, Testing plan, Design plan documents.

Process: As per client requirement, in order to complete the project, Project plan will be prepared by Project Manager, with the help of Development Manager, Test Manager. This plan document contains scope, resources, roles and responsibilities, what to develop and how to develop etc.

6. Design:-

Task: Requirements designing for application development.

Role: Architect/ Technical team

Document: HLD [High Level Design]

LLD [Low Level Design]

Process: Based on requirements and plan, before start coding by programmer, Architect / Technical team design the requirements as a HLD / LLD documents.

HLD defines the requirements as a module wise.

LLD defines the requirements as a functionalities.

7. Coding:-

By following requirements, plan, design, programmer start writing code for responsible modules development using different technologies such as Java, C#, Python, Ruby etc.

8. Testing:-

Whenever all implementation activities completed, to confirm the correctness of code which is carried out by programmer called “White Box Testing”. This testing collectively of 2 levels – i) Unit Testing and

ii) Integration Testing.

Whenever 100% coding and White Box Testing completed, to confirm the correctness of customers requirements, testing an front end application called “Black Box Testing”. It is a collectively of 2 levels – i) System Testing and

ii) User Acceptance Testing.

9. Release Process:-

Once all activities are successfully completed, if client also accepted in user acceptance testing, then the delivery team will release the project to the customer.

10. Support/Maintenance:-

After delivery to the customer, during utilization, if customer is facing any problem called “Failure”. If any failure occurred, to handle those requirements, Maintenance team will provide support to the project.

Difference between Verification and Validation

Verification :- It’s a process of verifying developing the project is right or not called “Verification”, also called as “Static Testing”.

Verification is to check, whether software confirms to the specifications and it is done by Development team, it is a in-house activity of the development, it is an Quality Assurance (QA) activity.

Validation :- It’s a process of validating the developed product is wrong or not, called “Validation”, also called as “Dynamic Testing”.

Software Development Models

There are various Software Development Approaches defined and designed which are used during development process of software. Those approaches referred as “Software Development Models”.

Software Development Models classified into –

1) Sequential Models

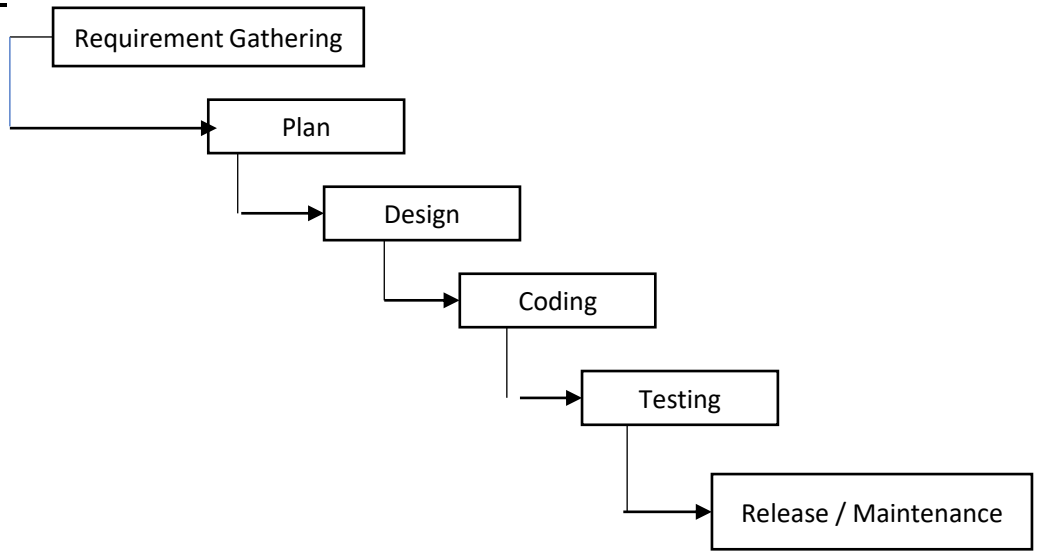
2) Incremental Models

1) Sequential Models :-

These models are best suitable for small size of projects, where all SDLC activities will be carried out one after another, for all entire project.

Waterfall model and V model are best examples for sequential model.

I) Waterfall Model :-



- It is a sequential model, suitable for small size of projects.
- Waterfall model same as SDLC process.
- Whenever client requirements are clear in Waterfall model all implementation activities will be carried out step by step process.
- In Waterfall model, only Validation required, verification not required.
- In this model, flow of activity looks like a waterfall, so this model is titled as Waterfall model.

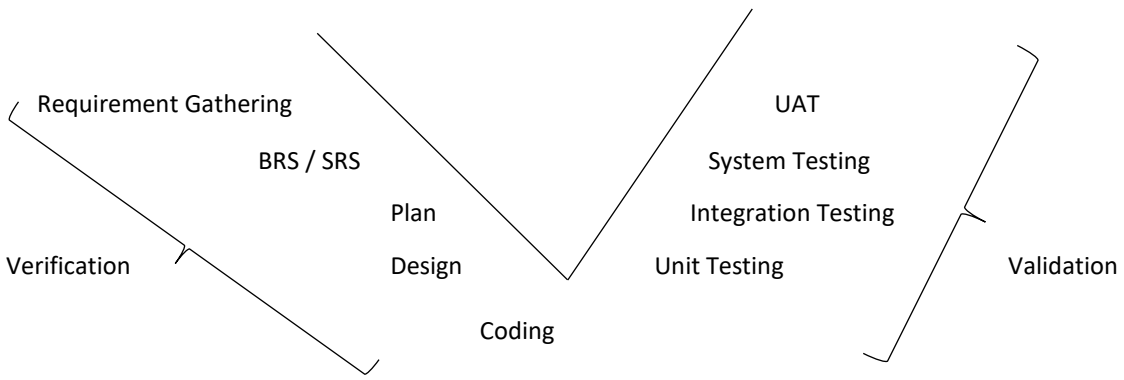
Advantages:-

- Easy to understand and simple.
- Works well for small projects.
- Easy to manage step by step activities.

Disadvantages:-

- Not suitable for big size projects.
- In middle of the project, if any change requirements needed then going back and changing requirements are tedious job.

II) **V Model** :-



‘V’ stands for Verification and Validation. This model is suitable for small size of application, where the requirements are clear and chances of making mistakes are more, while implementing the application to reduce this at every step of implementing software testing apply both verification and validation.

In ‘V’ model, once requirement documents are ready, based on those requirements, along with developers, testers will start testing activities such as Test plan, Test Scenario, Test cases documentations. So testing activities will start before coding.

Advantages:

- Simple and easy to use.
- Save time.
- Testing activities will be start before coding.
- Ensure 100% quality.
- Implementing both verification and validation .

Disadvantages:

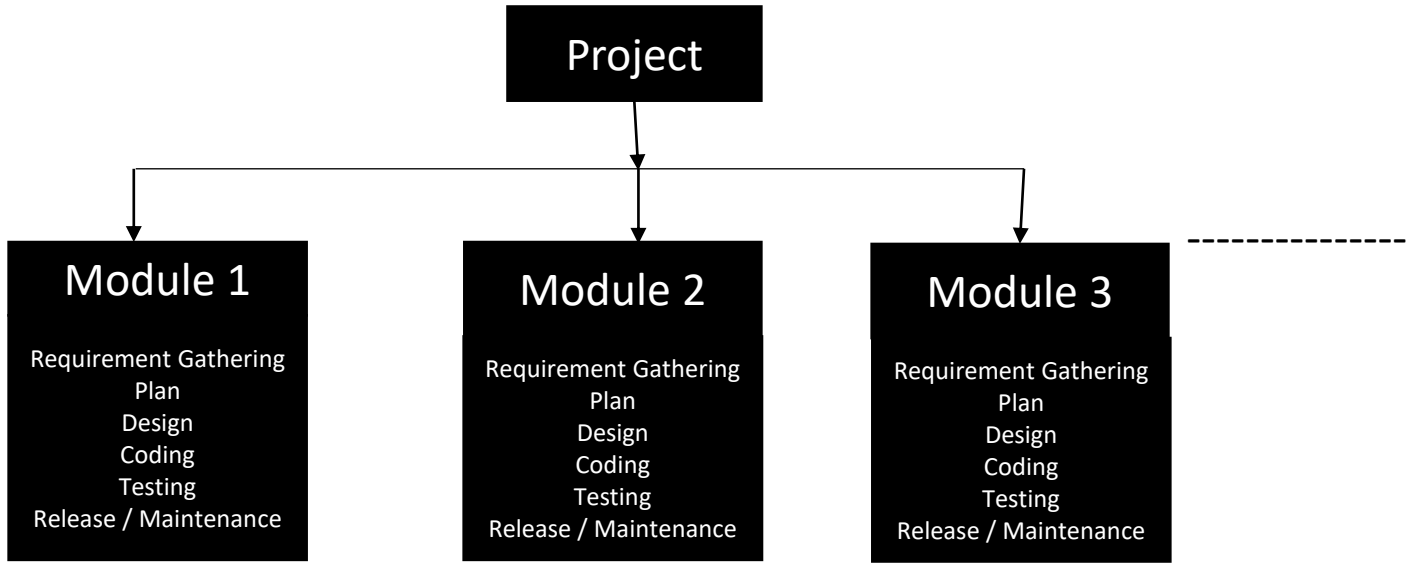
- Not suitable for big size of the projects.
- If any changes happen in middle of the project, along with requirement, development and testing documents has to be updated.

2) **Incremental Models**:

These models are best suitable for big size of project. In incremental model, a big project will be dividing into modules, then all SDLC activities will be carried out module by module.

RAD model, Prototype model, Spiral model and Agile model are the best examples for Incremental models.

I. RAD Model :-



RAD stands for Rapid Application Development. Due to lack of time or within short term to complete big size of projects in RAD model, a big project will be dividing into modules and every module will be considered as a mini project, a separate team will be scheduled to implement all SDLC activities, for those modules parallelly. Once all modules are implemented, these modules will be combined and delivered to customer.

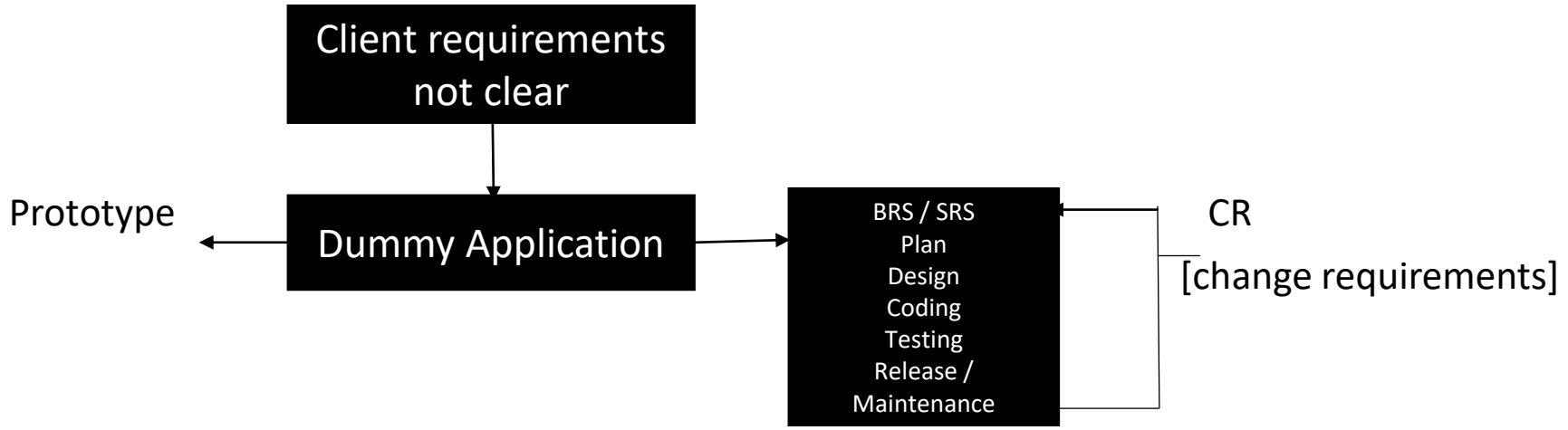
Advantages:

- Reduce development time.
- Suitable for big size of projects.

Disadvantages:

- Not suitable for small size of projects.
- Expensive model.
- Integration problem will be occurred.

II. **Prototype Model**:-



Due to lack of requirements or if customer confusing in the requirements, in this model instead of developing actual application, develop Dummy application called “Prototype “. It will be accepted by client.

Once the client approved the Prototype, based on the sample BRS/SRS will be designed, and based on those requirements all implementation activities carried out such as Plan, Design, Coding, Testing activities. If any changes requested by the customer after deliver, the same will be implemented as Change Requirement (CR). Those cycles will be continued with same process.

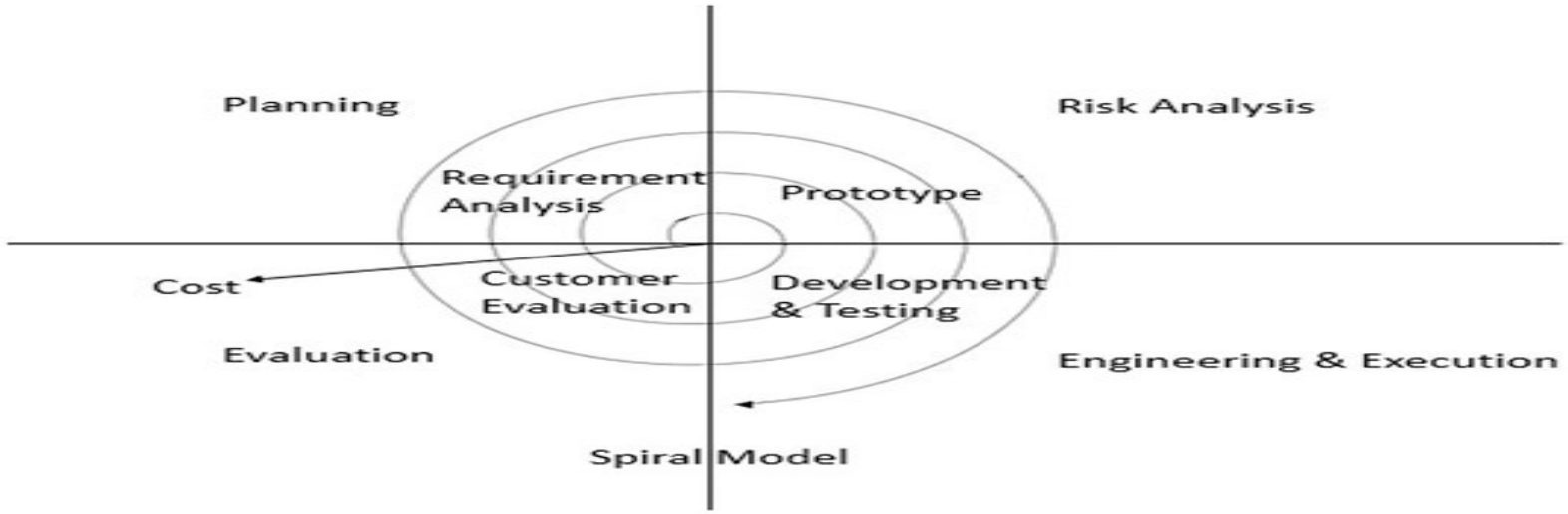
Advantages:

- Change Requirements will be implemented after deliver to the customer.
- If customer requirements are not clear also we can implement project based on prototype.
- Missing functionality can be identified easily.

Disadvantages:

- Not suitable for small size of projects.
- Expensive model.
- More time and resources required.

Spiral



This model is suitable for Maintenance or Long term projects. Whenever customer requirements are frequently changing or dynamic requirements of the customer, in this model application will be implemented requirement by requirement.

In this model, the flow of activity looks like a spiral net, so this model titled as a “Spiral Model”.

Advantages:

- Additional functionality can be added in next cycles.
- Project will be deliver to customer early in the software cycle.
- Best suitable for Maintenance and Long term projects.

Disadvantages:

- Expensive model.
- Require more time.
- Not suitable for short term or small size of projects.

IV. **Agile Model/ Agile Methodology:-**

Agile model is a Iterative (same process will be repeating again and again multiple times, means getting requirements, analysis, design, coding, testing will repeats) and Incremental (adding new features/modules on existing software) process or approach.

It is a day-by-day Waterfall model.

Agile Principles:

- Customer no need to wait for long time.
- We develop, test and release piece of software to customer with few number of features.
- We can accommodate / accept requirement changes from customer easily.
- In this model, work along with client representative team from day 1 onwards.
- Release is very faster within 2/3 weeks can release product/project.

It has more frameworks, such as XP, Scrum, Iterative, where we will discuss on Scrum process.

Scrum

Scrum is a framework through which software product is build by following Agile principles.

Agile is a defined process with some principles.

Scrum includes a group of people called Scrum Team (normally 5 to 9 members). They are Product Owner, Scrum Master, Development (Dev) team, Testing (QA) team and everybody work together to deliver quality product within the short time.

Roles and Responsibilities of Scrum Team:

1. Product Owner:

- Product Owner is the main person who writes or defines functionality of product by directly contacting with client and get the requirements.
- He will prioritize those features and assigns to developers and testers.
- Accept or reject work result.

2. Scrum Master:

- Scrum Master takes care of entire Agile process (from beginning of software to till delivery).
- Project Manager or Client will work as a Scrum Master.
- He is responsible for sprint plan, sprint meeting and scrum meeting.
- He will handle any issues in team.
- He will give awareness to people on Agile process.

3. Dev Team:

Developers will develop the software.

4. QA Team:

Testers will test the software.

Scrum Terminology/Agile/Scrum Ceremonies

1) User Story:

It is a feature / module in a software. It gives general explanation of software feature.

Note:

Product Owner will define these User Stories and Epic.

3) Product Backlog:

It is a list of all user stories prepared by Product Owner at the beginning of Agile process.

4) Sprint:

Duration of time to complete User Stories, decided by Product Owner and Team, usually 2-4 weeks of time.

5) Sprint Planning Meeting:

Meeting conducting by team (Product Owner, Scrum Master, Dev team, Testing team), to define what can be delivered in the sprint / particular duration of time and to review previous user stories completed as per sprint plan or not and if any user stories are unable to complete within sprint plan then it will consider as a back logs and to explain new sprint user stories.

6) Sprint Backlog:

List of committed stories by Dev/QA for specific sprint.

7) Scrum Meeting: scrum call/standup meeting

Meeting conducted by Scrum Master every day 15 mins, called as Standup meeting / scrum call, to discuss the status of the project.

Everyday it focus on 3 things in this meeting:

a. What did you do yesterday?
b. What will you do today?
c. Are there any blockers in your way?

8) Sprint Retrospective Meeting:

It is conducted only once at the end of every sprint.

It focus on 3 things :

a)What went well.

b) What went wrong and

c) What steps need to be taken.

9) Story Point:

Rough estimation of user stories given by Dev and QA. During sprint planning meeting itself, every story will be estimated by Product Owner, Dev and QA.

1 story point = 1 hr/ 1 day (6-8 hrs) depends on company

Suppose, for

Login -> Dev given 5 (means 5 hrs)

QA given 3 (means 3 hrs),

so totally 5+3 =8 hrs(means 1 day)

10) Burndown/Up Chart:

It shows how much work remaining in sprint. This will be designed by ScrumMaster daily. It is a graph.

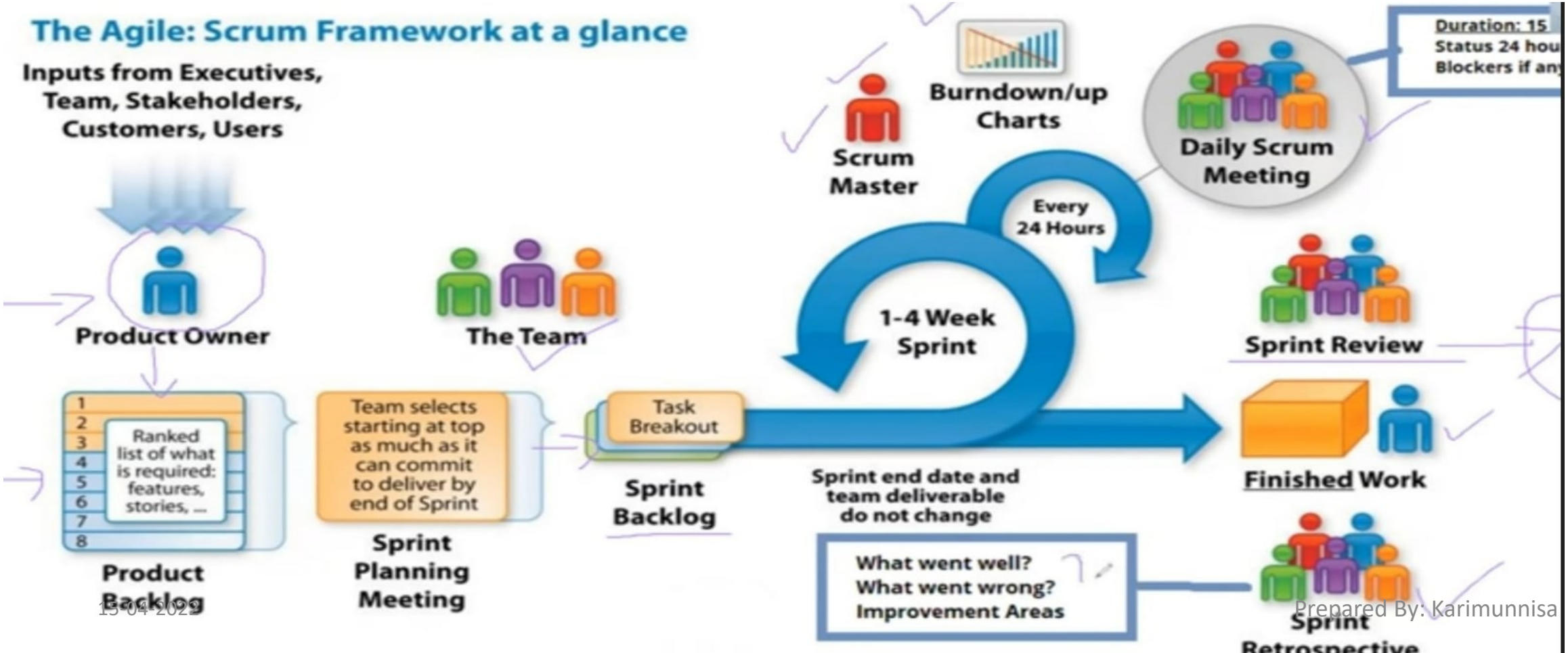
Advantages:

- Requirement changes are allowed in any stage of development.
- Release is very fast.
- Customer no need to wait for long time.

- Good communication between team.
- It is easy model to adopt. Flexible model.

<u>Disadvantages</u>:

- There is less documentation, since we deliver software very faster.



Software Testing Methods and Levels

Software Testing = Verification + Validation

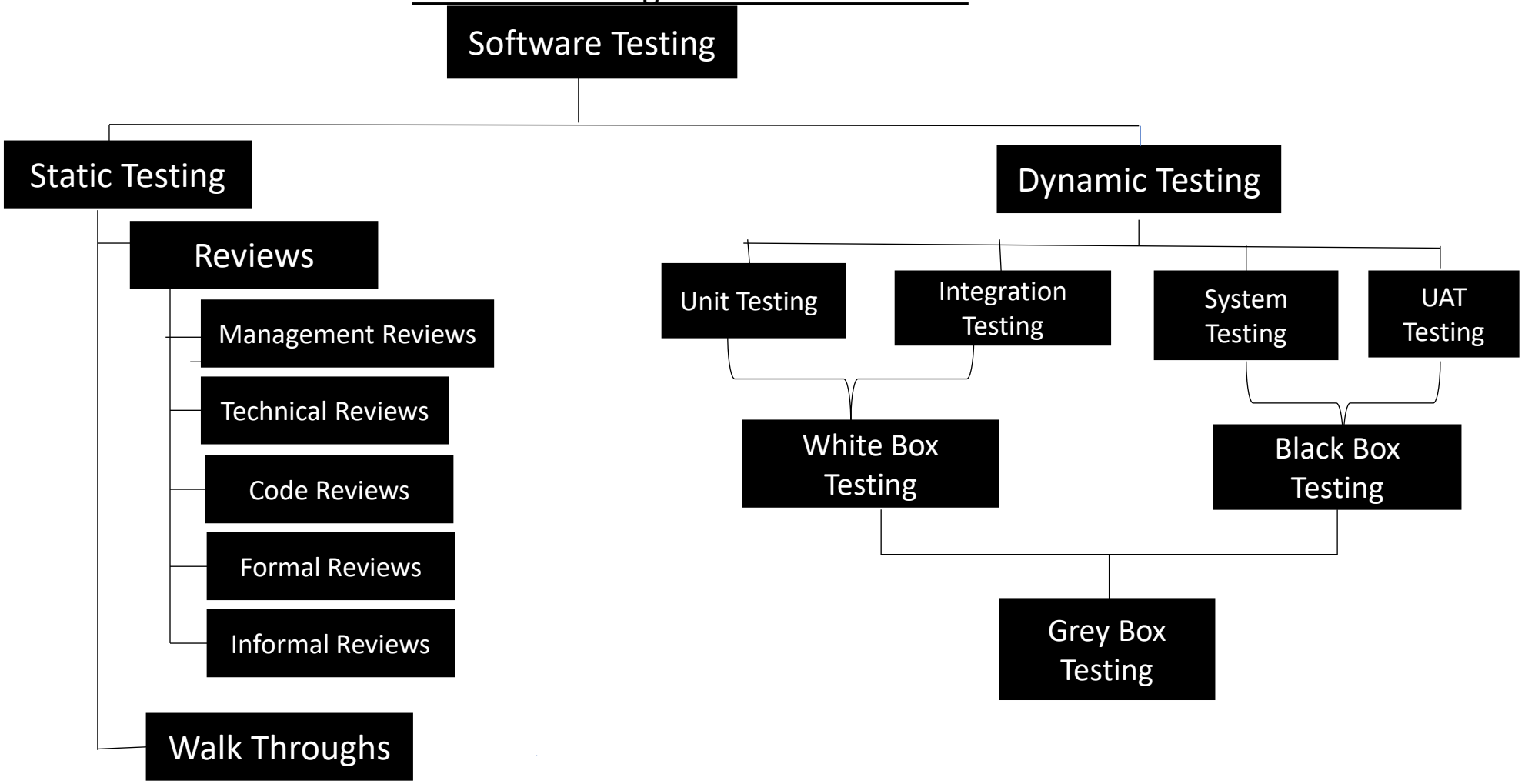
Verification:

It’s a process of developing a project is right or not is called Verification, also called as Static Testing.

Validation:

It’s a process of validating the developed application is right or not called Validation, also called as Dynamic Testing.

Software Testing Methods and Levels



Static Testing

Static Testing is carried out by Reviews and Walk Throughs.

1) Review:

Examining project related work called Review.

Example: Examining requirements, design, code etc..

Types of Reviews:

i. Management Review :

This review will be conducted by high level management team (Business Analyst, Project Manager) to monitor the project status.

ii. Technical Review :

This review will be conducted by technical team to decide the best approach of design implementation.

iii. Code Review:

This review will be conducted by programmer to decide the best approach of coding part.

iv. Formal Review :

If a review is carried out by following systematic procedures and proper documentations, then those reviews are called Formal review.

Inspections and Audits are best examples for formal review.

If a formal review conducted while executing task, then it is called Inspection.

If a formal review conducted after completion of task, then it called Audit.

v. Informal Review :

If a review is conducted without following any procedures and documentations, then those reviews are called Informal review.

Objectives of Static Testing: (Reviews)

i. To find defects in requirements.

ii. To find defects in design

iii. To identify the deviations in any process.

iv. To provide valuable suggestions to improve the process.

Note: To detect defects from requirements, plan, design, coding. we do static testing.

2) Walk Throughs :

A step by step code reviewed by experts called Walk Through.

- Review can be done by anytime and anyone, it is basically a informal review

Dynamic Testing

Dynamic Testing is carried out by 3 methods such as,

(1) WhiteBox Testing

(2) BlackBox Testing

(3) GreyBox Testing

WhiteBox Testing

After coding, before start QA activities, testing conducted on source code by programmer to ensure the 100% code coverage or whether code is working as per client requirement or not is called WhiteBox Testing.

It is a collectively of 2 levels, such as (i) Unit Testing

(ii) Integration Testing

(i) Unit Testing :-

A smallest testable part in the source code of the application (or) each and every part of the code working as per requirement or not called Unit Testing, also called as Module Testing (or) Component Testing.

- A unit is a single component or module of a software. So conducting testing on single component or module of a software is called Unit Testing.

While conducting Unit Testing, to ensure code coverages, programmer will follow whitebox testing techniques such as – (a) Condition coverage

(b) Loops coverage

(c) Path coverage

(d) Mutation coverage // they will chk both valid inputs and invalid inputs to chk the code coverage.

(ii) Integration Testing :-

Once the unit testing is completed, developers will integrate all source code units and checks interactions in between all modules, which is called Integration Testing.

Types of Integration Testing:-

(I) Incremental Integration Testing.

(II) Non Incremental Integration Testing.

Incremental Integration Testing :

Incrementally adding the modules and testing the data flow between the modules one after the other, is called Incremental Integration Testing.

There are 4 approaches of Incremental Integration Testing.

(1) Top-Down approach

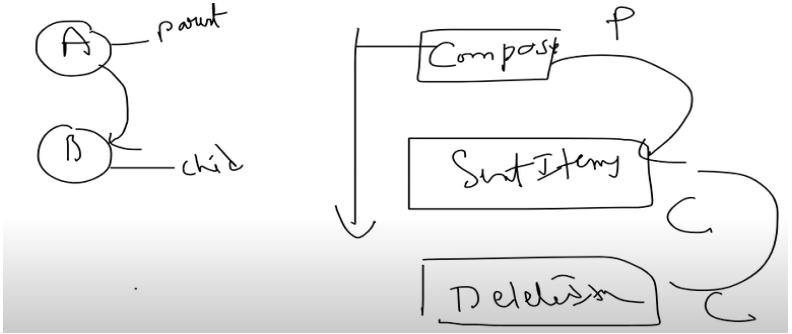
(2) Bottom-Up approach

(3) Hybrid approach (4) Big bang approach

(1) <u>Top-Down Approach</u>:

Incrementally adding the modules and testing the data flow between the modules. And ensures module added is the child of previous module.

Eg: A module is there, and when we integrate with B module then that B module should be the child module of A.



- This approach is recommended if there are any incomplete programs at top level.

- In this approach, integration testing will be carried out from top to bottom.

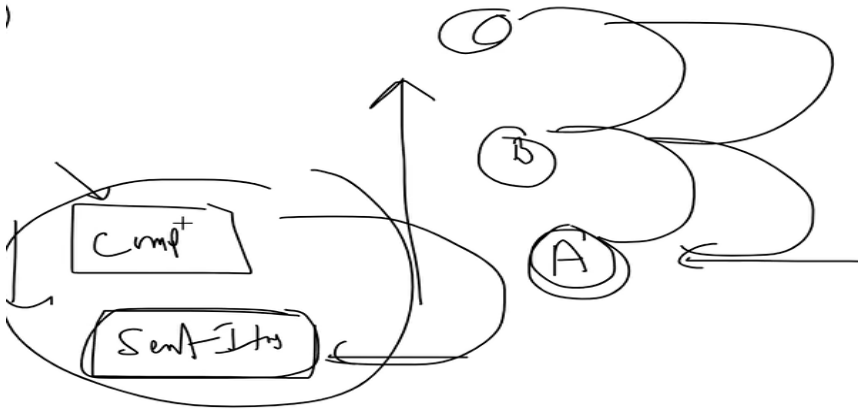
- The incomplete programme at bottom level will be replaced with **Stubs**.

Note:

Stub is a temporary code used to stop user actions when top module is incomplete.

2)**Bottom up Approach**:

Incrementally adding the modules and testing the data flow between the modules. And ensure the module added is the parent of previous module.



- This approach is recommended when there are incomplete programmes at the bottom level.

- In this, testing will be carried out from bottom to top.

- The incomplete programme at bottom level will be replaced with **Drivers.**.

Note:

Driver is a temporary code used to continue user actions when bottom module is incomplete.

3) **Sandwich/Hybrid approach**:

Combination of top-down and bottom-up approach is Sandwich/hybrid approach.

- In this the middle level modules are integrated with both Stubs and Drivers.

4) **Big bang approach** :

This approach is recommended whenever 100% coding completed and all source code units are available, then conduct integration testing called as Big bang approach.

Non incremental Integration Testing :

Here we integrate all the modules at one shot and test the data flow between the modules. (we don't prefer most of the time this type).

Drawbacks:

-> We might miss the data flow between some of the modules.

-> If we find any defect we can't understand the root cause of defect.

Black Box Testing

After 100% coding and WhiteBox testing, testing conducted on application by testing engineer (or) domain experts to ensure the requirement coverage (or) to check whether the application developed as per client requirement (or) not , called BlackBox Testing, also called as Closed Box Testing or Specification Based Testing.

It is collectively of 2 levels :

(i) System Testing

(ii) User Acceptance Testing

System Testing

System Testing is a process of validating both Functional requirements and Non Functional requirements.

- Validating client / business requirements are called Functional Testing.
- Validating client / business expectations are called Non Functional Testing.

Once the project is scheduled for system testing, every testing engineer validate Functional requirements.

Whenever 100% functional requirements validate, then tester can go through non functional requirements.

Types of Functional Testing

1) Smoke Testing / Build Verification Testing
2) Sanity Testing.
3) Positive / Negative Testing
4) Re Testing
5) Regression Testing
6) Database Testing
    - Data Validation
    - Data Integrity
    - Data Volume Testing

6) Exhaustive Testing
7) End-to-End Testing
8) Adhoc Testing
    - Buddy Testing
    - Pair Testing
    - Exploratory Testing
    - Monkey Testing

(1) Smoke Testing / Build Verification Testing :

Functional testing starts with Smoke Testing.

In this type of testing, once build release by developer, to check whether application is stable or not, by checking basic functionality called Smoke Testing, also called as Build Verification Testing.

we will chk whether the build is stable or not.

(2) **Positive / Negative Testing** :

Positive Testing :

In this type of testing, application check with valid inputs called Positive Testing.

Negative Testing :

In this type of testing, application check with invalid inputs called Negative Testing.

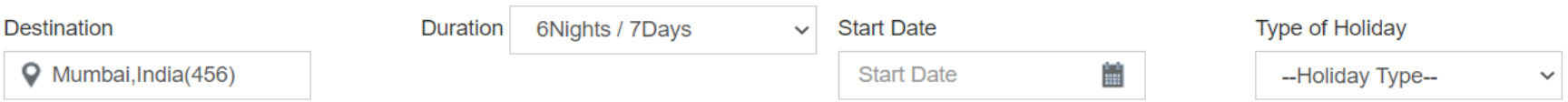
Note:

- The objective of positive testing is to confirm customer requirements.
- The objective of negative testing is to find defects.

(3) **Re Testing** :

Once defect fixed by developer, to check its working as per customer requirements or not, checking or testing again and again that functionality called Re Testing.

Eg:



Here, if “Type of Holiday” is not working, then the tester sent this defect to developer, where developer fixes the defect and sent back to tester, then the tester will test the defect again and again for quality. This is called Re-Testing.

(4) **Regression Testing** :

Once the defect fixed by developer, to check side effects of functionality along with defect, testing defect related all functionalities called Regression Testing.

We can ensure 100% customer satisfaction only by Regression testing, but manually its required huge resources to perform regression testing, but it can handle through automation tools like Selenium.

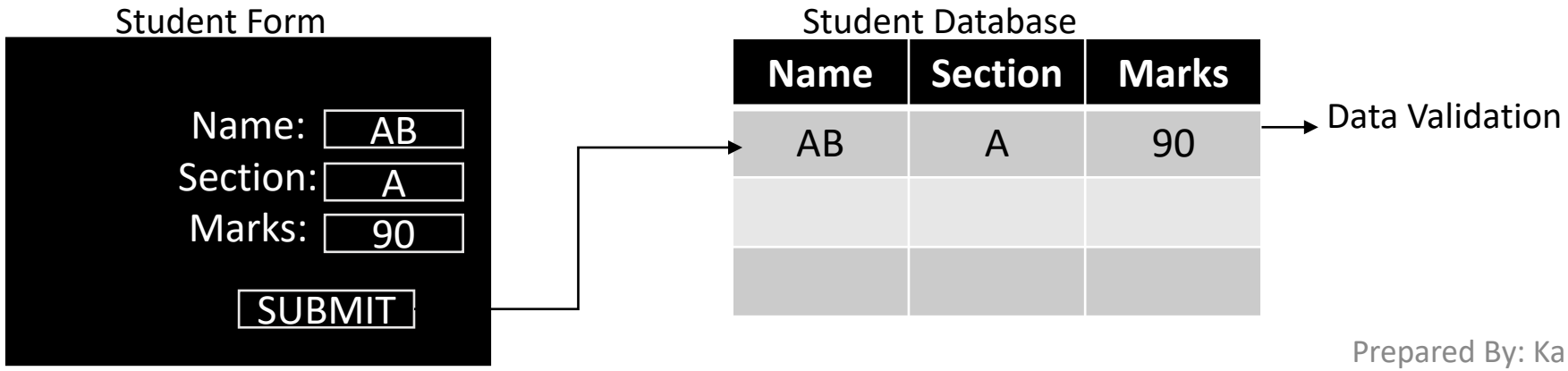
(5) **Database Testing** :

It is a collection of data backend application of software, every data will be insert into database as a table format.

During database testing, we can perform below types of testing's.

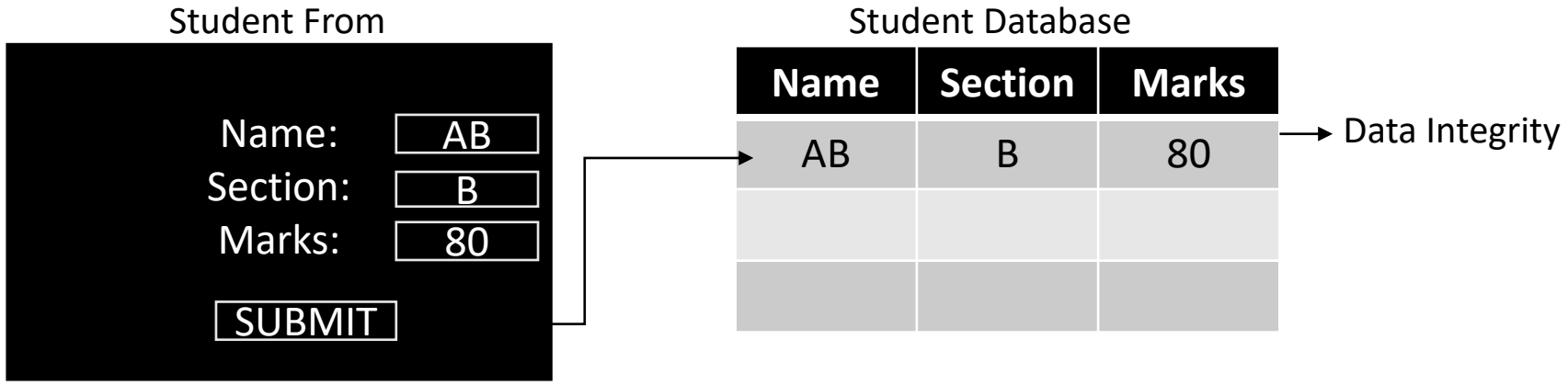
(i) Data Validation :

In this type of testing, insert new data into the database and checking data will be correctly inserted or not, called Data Validation.



(i) Data Integrity :

In this type of testing, update existing data and check correctly updated or not, called Data Integrity.



(ii) Data Volume Testing :

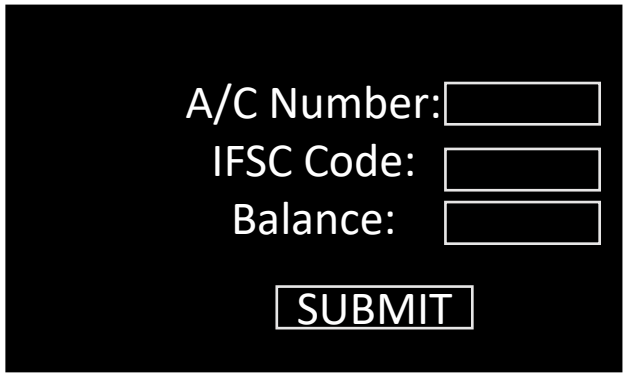
In this type of testing, checking capacity of database by entering sample data into the database, called Data Volume Testing or Data Memory Testing.

(6) **Exhaustive Testing** :

If we test a functionality with all possible inputs called Exhaustive Testing.

We can perform Exhaustive testing on Account Number, IFSC code functionalities, but to perform exhaustive testing on maximum balance its required huge resources. So it requires Automation help.

Example: Bank Application



(7) **End-To-End Testing** :

If we conduct testing on application from starting to ending by all requirements of application called End-To-End Testing.

We can perform this testing whenever we have sufficient time.

(8) **Adhoc Testing** :

Due to lack of time, instead of testing all modules, testing main modules of application called as Adhoc Testing.

Due to lack of time, under Adhoc testing, tester's will perform below types of testing's.

(i) Buddy Testing

(ii) Pair Testing

(iii) Exploratory Testing

(iv) Monkey Testing

(i) **Buddy Testing** :

Due to lack of time, testing team can join with development team to continue development and testing parallelly to complete as soon as possible called Buddy Testing.

(ii) **Pair Testing** :

Due to lack of time, testing team will combine as a pair or group and perform testing on the project. While working they can share knowledge and ideas of project called Pair Testing.

(iii) **Exploratory Testing** :

Due to lack of skills, explore knowledge on project by using search engine, interactive with seniors or verify project related documents, then work on project is called Exploratory Testing.

(iv) **Monkey Testing** :

Due to lack of time, without documents, process, standards work on the project randomly to break the system called Monkey Testing.

Sanity Testing :

The Sanity testing is used to determine that the changes or the functionality are working as expected. If the sanity testing fails, software product is rejected by the testing team to save time and money.

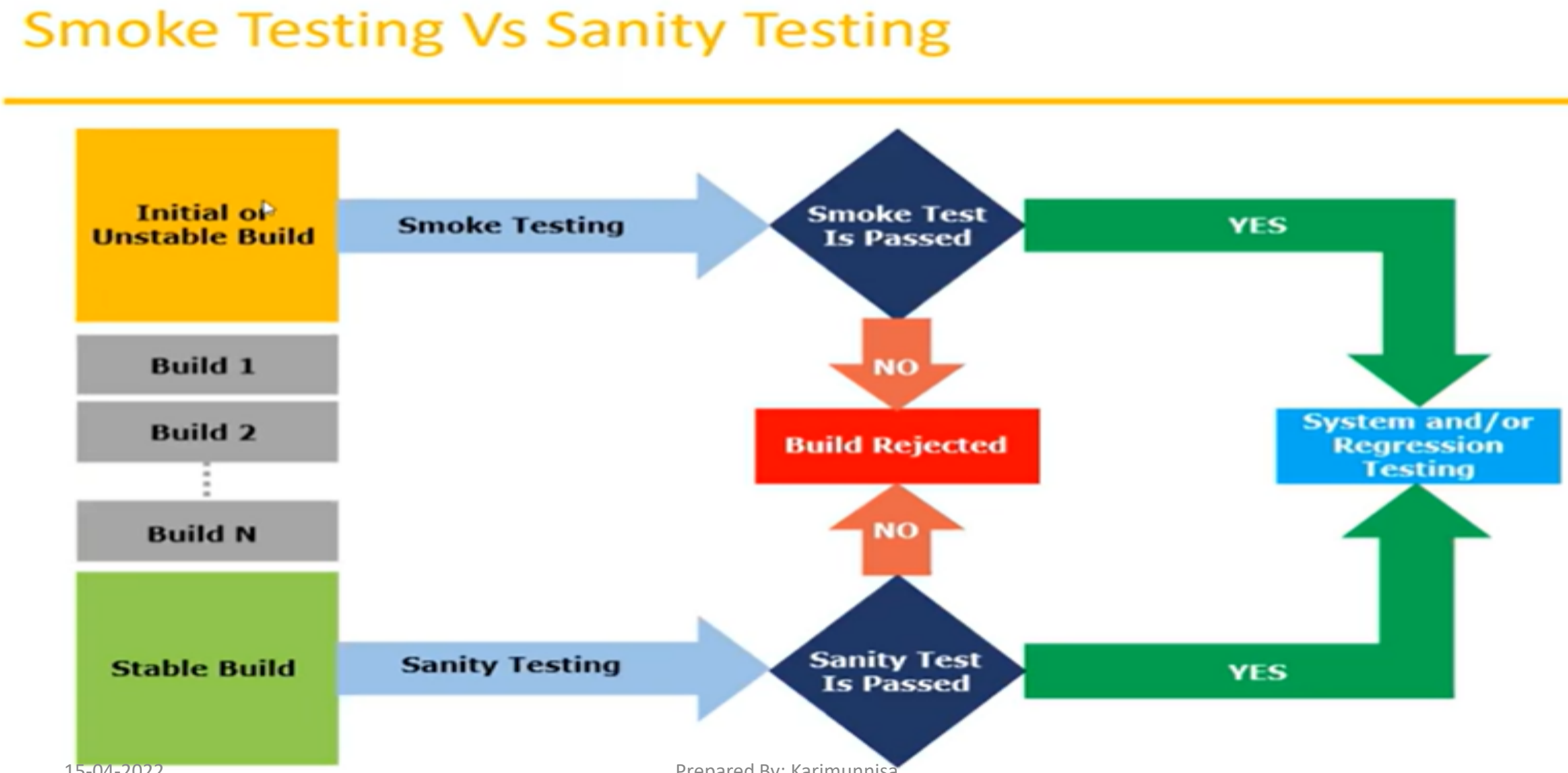
It is performed only after the software product has passed the smoke test and QA team has accepted for further testing.

Smoke Vs Sanity Testing

- Smoke and Sanity Testing come into the picture after build release.

Smoke Testing	Sanity Testing
Smoke Test is done to make sure the build we received from the development team is testable/stable or not	Sanity Test is done during the release phase to check for the main functionalities of the application without going deeper.
Smoke Testing is performed by both Developers and Testers	Sanity Testing is performed by Testers alone
Smoke Testing, build may be either stable or unstable	Sanity Testing, build is relatively stable
It is done on initial builds.	It is done on stable builds.
It is a part of basic testing.	It is a part of regression testing.
Usually it is done every time there is a new build release.	It is planned when there is no enough time to do in-depth testing.

Smoke Testing Vs Sanity Testing



Non-Functional Testing

Once the project is validated by all functional requirements to ensure customer expectations, testing engineer can perform below types of non-functional testing's.

(1) GUI Testing

(2) Usability Testing

(3) Compatibility Testing

(4) Security Testing

(5) Installation Testing

(6) Recovery Testing

(7) Standard Testing

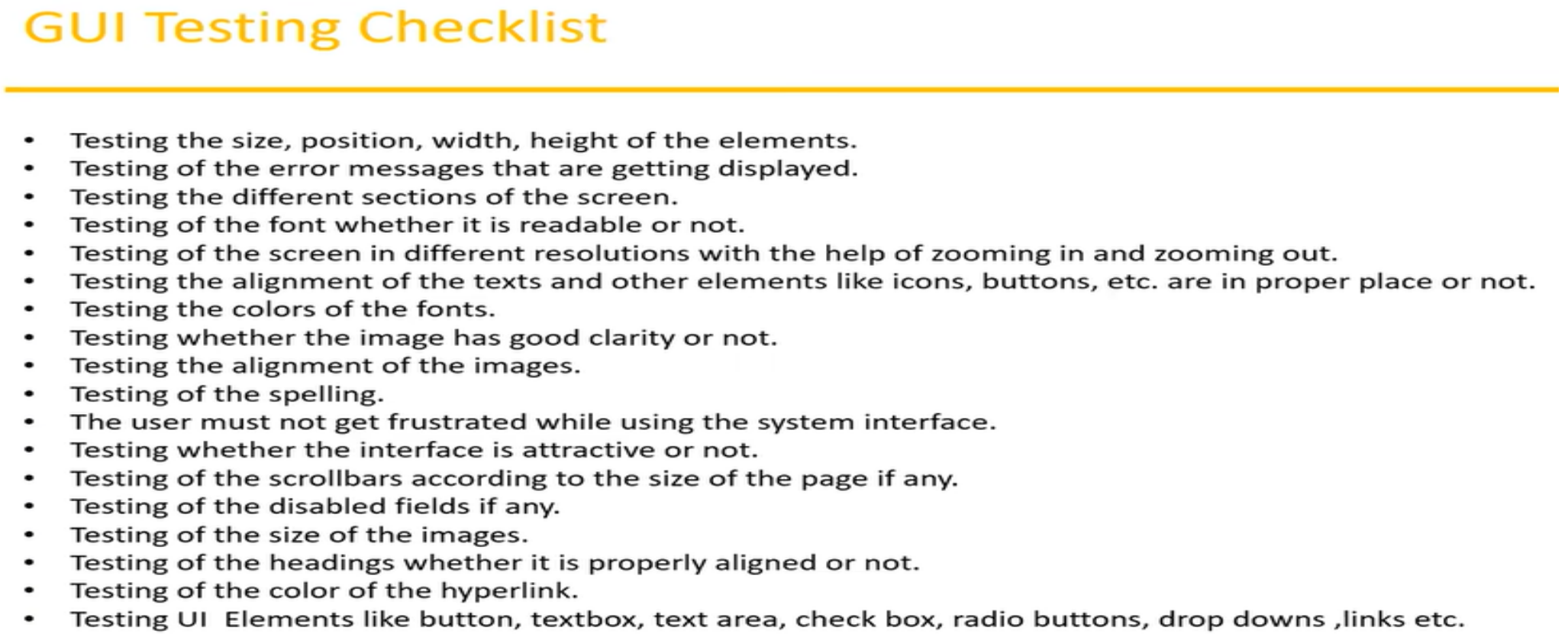
(8) Performance Testing

(1) **GUI Testing** :

Graphical User Interface Testing or GUI testing is a process of testing the user interface(UI) of an application.

A GUI includes all the elements such as menus, checkbox, buttons, colours, fonts, sizes, icons, content and images.

GUI testing checklist:



(2) **Usability Testing :**

In this type of testing in customer point of view, application is user friendly or not means user is able to understand and operate the application easily or not.

During this testing, validates application provided context sensitive help or not to the user.

(3) Compatibility Testing : Configuration Testing:

In this type of testing, to check the application compatibilities, validate both software compatibilities and hardware compatibilities.

Software Compatibility :

In this testing application validate by various operating systems, various browsers supporting or not called Software Compatibility.

Hardware Compatibility :

In this testing application validate by various hardware devices configuration supporting or not called Hardware Compatibility.

(4) **Security Testing :**

To check whether application is securable or not, security tester will perform below types of testing.

Authentication Testing:

In this type of testing, we need to check whether the valid user is accepted or not and invalid user is rejected or not, called Authentication Testing.

Authorization Testing (Access Control) :

In this testing, whenever the login as a level of user into the application, check whether he crossing any beyond user limits or not, called Authorization Testing or User Permission Testing.

Encryption Testing :

In this testing, to check whether the securable data default converted into encrypted format or not, called Encryption Testing.

(5) **Installation Testing :**

In this testing, we need to follow installation guidelines which are defined under installation document.

During this test, we have to verify below steps:

- While installation, application is user friendly or not.

- After installation, check memory.

- Verify uninstallation process.

(6) **Recovery Testing** :

In this type of testing, checks how the system handles unexpected events such as power failure, application crash etc.

(7) **Standard Testing :**

In this type of testing, we need to check whether application developed by following company standards or not such as ISO [International Standard Organization] etc.

(8) **Performance Testing :**

Speed in processing to check performance of the application, testers will follow below types of testing. Jmeter tool

(i) Load Testing :

The execution of software under customer expected configuration and customer expected load , to estimate the speed in processing and is there any chances of breaking the application, called Load Testing.

Note: gradually (slowly) increase the load on the application.

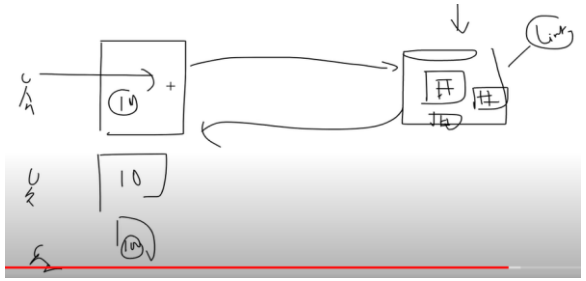
(ii) <u>Stress Testing</u> :

The execution of software customer expected configuration and more than customer expected load, called Stress Testing.

Note: suddenly increase/decrease the load and checks the speed and stability of the application.

(iii) **<u>Volume Testing</u> :**

Volume means size, testing how much data the application can handle.



Functional Testing Vs Non-Functional Testing

Functional Testing	Non-functional Testing
- Validates functionality of Software. - Functionality describes what software does. - Concentrates on user requirement. - Functional testing takes place before Non-functional testing.	- Verify the performance, security, reliability of the software. - Non-Functionality describes how software works. - Concentrates on user expectation. - Non-Functional testing performed after finishing Functional testing.

User Acceptance Testing

User Acceptance Testing is performed after the product is thoroughly tested. In this testing, client itself use the application to make sure if everything is working as per the requirements and perfectly in the real world scenario.

User Acceptance Testing is also known as End User Testing.

User Acceptance Testing conducts at 2 levels:

(1) Alpha Testing

(2) Beta Testing

(1) Alpha Testing :

Alpha Testing is done by users or customers in development or testing environment means they will come back to company where ever software is developed and do the testing and this testing is done before the software is released for Beta test.

(2) **Beta Testing** :

Beta Testing is done by users or customers in a real/customers environment.

Note:

After Alpha and Beta testing's, the product will release to production environment and then actual users will start using the software.

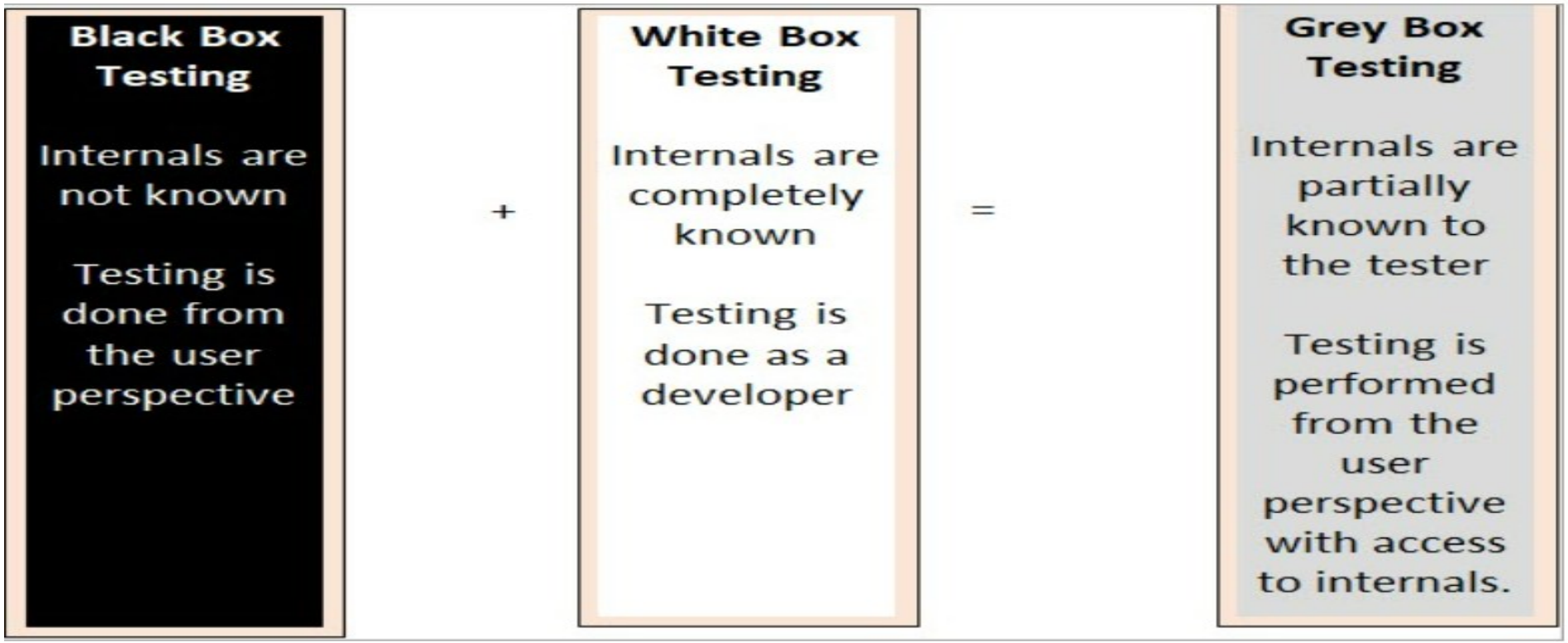
Alpha Testing Vs Beta Testing

Alpha Testing	Beta Testing
First phase of testing in customer validation.	Second phase of testing in customer validation.
Performed at developer's site – testing environment. Hence the activities can be controlled.	Performed in real environment, and hence activities cannot be controlled.
Whitebox and/or Blackbox testing techniques are involved.	Only Blackbox testing techniques are involved.
Build released for Alpha Testing is called Alpha Release.	Build released for Beta Testing is called Beta Release.
System Testing is performed before Alpha Testing.	Alpha Testing is performed before Beta Testing.
Issues/Bugs are logged into the identified tool directly and are fixed by developer at high priority.	Issues/Bugs are collected from real users in the form of suggestions/feedbacks and are considered as improvements for future releases.
Alpha Testing is used to evaluate the quality of the product.	Beta Testing is used to evaluate customer satisfaction.
It focus on finding bugs.	It focus on collecting suggestions/feedback and evaluate them effectively.

GreyBox Testing

It also called as GreyBox Analysis. In this testing, tester has limited knowledge of internal details of the design than test application called GreyBox Testing.

- This testing technique is a combination of Black box testing and White box testing.
- In Black box testing, the tester does not have any knowledge about the code. They have information for what will be the output for the given input. In White box testing, the developer has complete knowledge about the code.
- Grey box tester has knowledge of the code, but not completely.



Purpose

- The purpose of Grey box testing is to improve the quality of the product for which it covers both functional and non-functional testing altogether, which saves time and the lengthy process to test the application.
- Another purpose is to have the application tested from the user's perspective rather than the designer's opinion, and to get the developers enough free time to fix the bugs.

Examples

- **#1)** While testing a website, if the tester clicks on any link and it comes up with an error, the Grey box tester can make changes in the HTML code to check the same.
- In this particular scenario white box testing is being done by changing the code in HTML directly and can be verified in real-time. and as the tester is testing the changes at the front end–black box testing is also being done. Combination of the White box and Black box results in grey box testing.

Advantages:

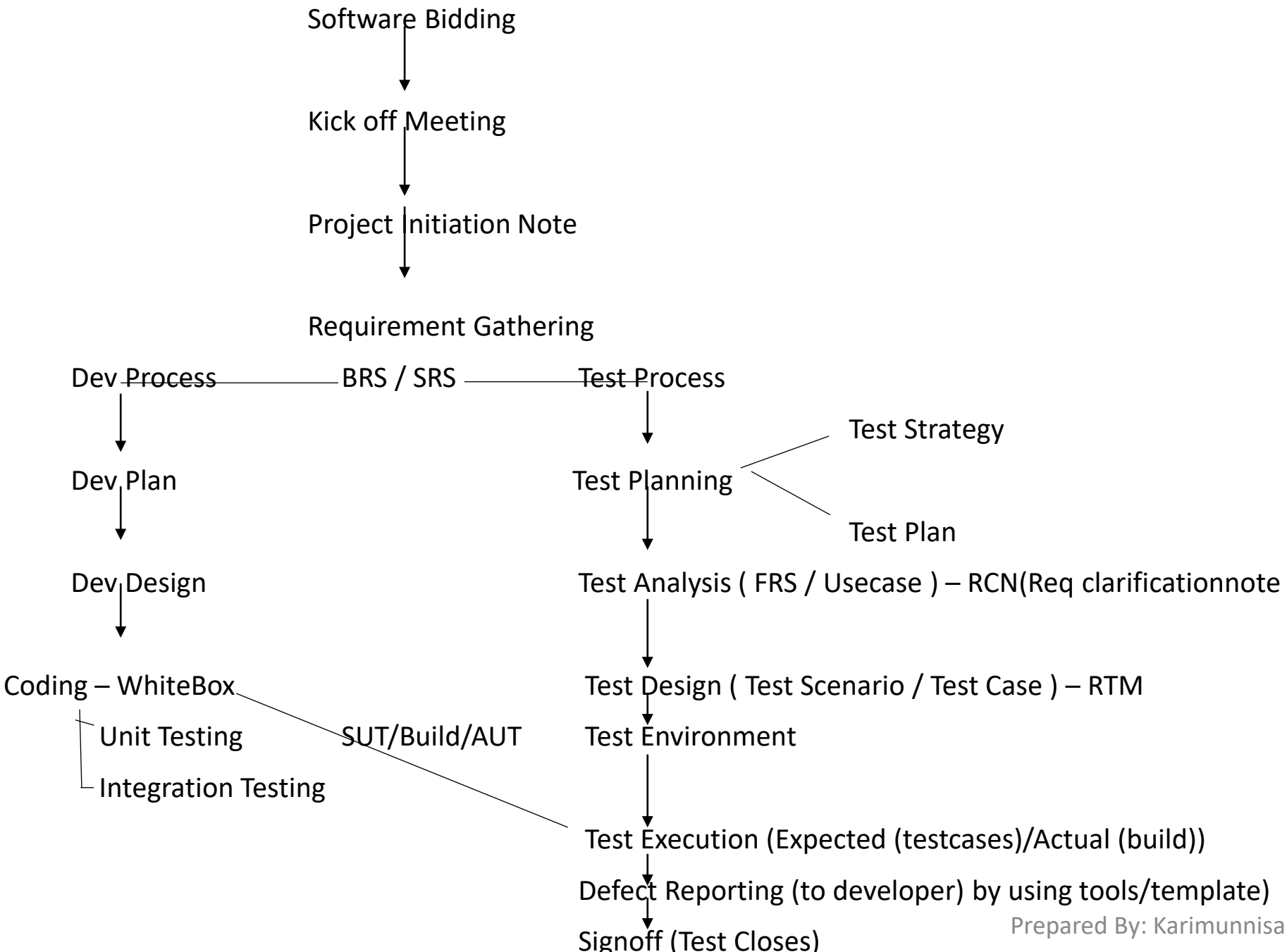
Grey box testing has several advantages. **Few of them are mentioned below:**

- Improves the quality of the product.
- Complex applications can be tested with ease.
- Do not require the tester to have complete knowledge of coding.
- Obtain benefits of both black box and white box testing.

Difference between Black Box, White Box and Grey Box Testing

Black Box Testing	White Box Testing	Grey Box Testing
Testers does not have any knowledge and information for the internal structure.	The internal structure is completely known.	The internal structure is partially known.
Also known as Closed box, Functional testing and Data-driven testing.	Also known as Glass, Clear box, Structural testing, or Code-based testing.	Also known as Gray box testing or translucent testing.
Done by testers, developers and the end-users.	Done by testers and developers only.	Done by testers, developers and the end-users.
Testers has nothing to do with the internal working of the system.	Testers has no knowledge of the internal working of the system.	Testers has partial knowledge of the internal working of the system.
Hidden errors cannot be found easily.	Hidden errors can be found easily as internal working is well to the developer.	Not easily found but can be found at user level.

STLC


Software Bidding
Kick off Meeting
Project Initiation Note
Requirement Gathering
Dev Process
BRS / SRS
Test Process
Dev Plan
Test Planning
Test Strategy
Test Plan
Dev Design
Test Analysis (FRS / Usecase) – RCN(Req clarificationnote
Coding – WhiteBox
Test Design (Test Scenario / Test Case) – RTM
Unit Testing
SUT/Build/AUT
Test Environment
Integration Testing
Test Execution (Expected (testcases)/Actual (build))
Defect Reporting (to developer) by using tools/template)
Signoff (Test Closes)

STLC

STLC stands for “Software Testing Life Cycle”, it will explain all the implementation activities of testing process.

STLC PHASES

1) Test Planning
2) Test Analysis
3) Test Design
4) Test Environment
5) Test Execution
6) Defect Reporting
7) Sign Off

(1) Test Planning :

STLC starts with Test Planning. At this phase prepare 2 documents such as

(i) Test Strategy and

(ii) Test Plan documents.

- Test Strategy is a high level plan or project level plan which is prepared by Test Managers.
- Based on strategy, number of Test Plan documents will be prepared by Test Lead for responsible module.

(2) **Test Analysis** :

In this phase, to understand project requirements testing engineer start analysing various requirement documents such as – (i) FRS and

(ii) UseCase documents

While analysing requirements, if we need any clarifications, update into RCN (Requirement Clarification Note) and send to BA (Business Analyst) (Domain Experts).

(3) **Test Design** :

In this phase, design the requirements as a 2 documents such as (i) Test Scenarios and

(ii) Test Cases

- What to test (or) one unique test condition is called Test Scenario.
- How to test and step by step process is called Test Cases.

Number of test scenarios and test cases update into RTM (Requirement Traceability Matrix).

(4) **Test Environment** :

It is a combination of software and hardware configurations such as browser, operating systems, system configurations etc.

(5) **Test Execution** :

Whenever test cases prepared by testing engineers and as per release date, build released by developer to compare expected values and actual values, every test case execute on build.

(6) **Defect Reporting** :

During execution, if any mismatches in between expected and actual called Defect.

By the help of defect tracking tool, defect will report to developer.

(7) **Sign Off / Test Closer** :

Whenever all the testcases are executed successfully and all major defects are fixed, then tester can signoff from current project called Test Closer.

Test Planning

Once project is scheduled for system testing, prepare 2 types of plan documents, such as-

(1) Test Strategy

(2) Test Plan document.

(1) Test Strategy Document :

- It is a high level plan document developed by Test Manager with the help of Project manager, based on BRS and SRS/FRS documents.
- This documents defines software testing approach to achieve testing objectives.
- It is a static document, that is not updated.

Some companies include Test Strategy with plan and prepare as a only one Test Plan document.

Components of Test Strategy Document :

I. Scope and objectives
II. Business issues
III. Roles and Responsibilities
IV. Communication and Status reporting
V. Test deliverability

VI. Industry standards to follow

VII. Test automation and tools

VIII. Testing measurements and metrics

IX. Risks and mitigation

X. Defect reporting and tracking

XI. Change and configuration management

XII. Training plan

(2) <u>Test Plan Document</u> :

Test Plan document is a Module level plan prepared by Test Lead, based on Test Strategy and requirement documents, by following IEEE 829 format.

- IEEE stands for Institute of Electrical and Electronic Engineer's.
- This plan document contains Scope, Resources, Roles & Responsibilities etc.
- Test Plan will explain what to test (or) when to test requirements.

<u>IEEE 829 Test Plan Document</u>:

<u>Test Plan Document Components</u> :

1. Project Overview
2. References
3. Scope

4. Testing types
5. Entry Criteria
6. Suspension and Resumption criteria
7. Exit Criteria
8. Communication plan
9. Status reporting
10. Test Execution flow
11. Defect tracking
12. Test Environment
13. Test Deliverables
14. Staffing plan
15. Risk analysis
16. Mile stones
17. Approvals

1. Project Overview :

    Introduction of current project

2. References :

Names of requirement document's either FRS or Usecase document.

3. Scope :

List of modules from the project.

(i) In Scope : List of modules to be tested

(ii) Out Scope : List of modules not be tested.

4. Testing Types :

List of testing types which is need to perform on project. Eg. Functional testing type/performance test etc.

5. Entry Criteria : [when to start testing]

(i) 100% unit testing and integration testing should be successful and build release.

(ii) All test cases should be prepared and reviewed.

6. Suspension and Resumption Criteria :

Suspension Criteria	Resumption Criteria
(i)When show stopper defects found.	(i)If patch is releases against build rejection.
(ii) When there is a change request from the client.	(ii) Once CR is implemented.

7. Exit Criteria : [when to stop testing]

(i) When all test cases executed successfully and passed

(ii) When all defects are resolved and closed.

8. Communication Plan :

List of channels to interact with team members (test lead, test manager, project manager, client).

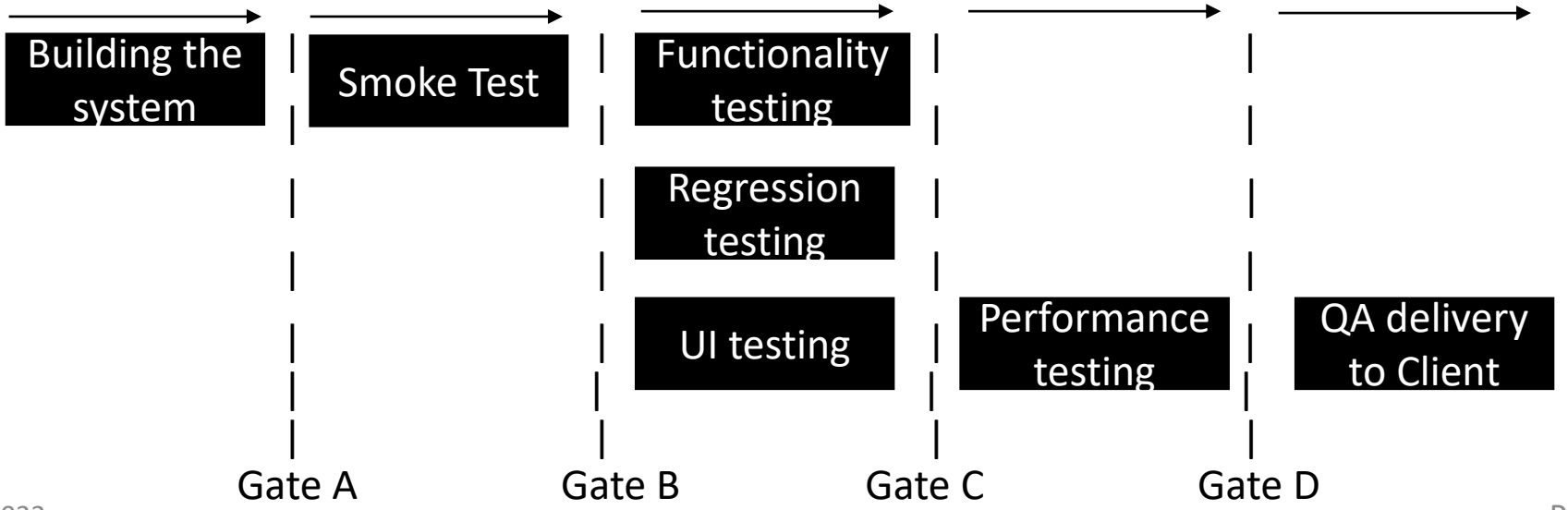
Eg: Email, go to meeting, zoom meetings etc.,.

9. Status Reporting :

To intimate work status every QA need to prepare status reporting (daily report or weekly report etc).

10. Test Execution Flow :

While executing different types of testing's on the project, we have to follow Test Execution flow which is designed by Test Lead.



11. Defect Tracking :

If expected value not equal with actual value called Defect.

To define importance of defects need to confirm Priority and Severity levels.

Severity: It indicate impact of defect.

Priority: it indicates urgency to fix.

12. Test Environment :

It is a combination of both software and hardware configurations. Eg. os, browsers, system configurations, etc.,

13. Test Deliverables :

List of testing documents which is need to prepare by testing engineers.

Eg: Test plan, Test scenario, Test cases, Defect reports, Test summary reports etc.,.

14. Staffing Plan :

Confirm team size and roles and responsibility.

Eg:

Name	Number of person's	Responsibility
Test Lead	2	Planning & controlling QA activities.

15. Risk Analysis :

List of risks and how to overcome (mitigations/solutions)

16. Milestones :

To complete work on time, confirm work schedules as a task, effort, start date, end date.

Mile Task	Effort	Start Date	End Date	Deliverables
Test Plan	3 Hrs			Test plan approval
Test Scenarios Design	16 Hrs			Test Scenario document
Test Cases Design	15 Hrs			Test Case document

17. Approvals :

Finally documents before send to team members it should be approved by Test Manager or Project manager.

Review Test Plan Document

Once Test Plan prepared by Test Lead before start analysing by Team members, in order to check whether plan prepared as per client requirement or not perform below reviews –

(i) Self Review : Reviewed by same person.

(ii) Peer Review : Reviewed by colleagues.

(iii) Lead Review : Reviewed by Team Lead.

(iv) Manager Review : Reviewed by Test Manager or Project Manager.

(v) Client Review : Reviewed by client.

Test Analysis

Once test plan prepared and reviewed, in this phase, testing engineer will analyse various requirements such as SRS/FRS, usecase documents to understand what to be test and how to test all requirements.

While analysing requirement's, if there are any questions or doubts, we record out in a process template called RCN [Requirement Clarification Note] and we send this document to BA to get clarifications.

Business Requirement Specification [BRS] Document Template :

1.0	Introduction
	1.1 Client Introduction
	1.2 Project Introduction
2.0	Existing System
3.0	Drawbacks in existing system
4.0	Proposed system
5.0	System Architecture
6.0	Business Requirements

System Requirement Specification [SRS] Template :

SRS also called as FRS [Functional Requirements Specification] document (or) FD [Functional Document] (or) BRD [Business Requirement Document] (or) BDD [Business Design Document]

1.0	Overview
2.0	Prototype
3.0	Form/Page Elements
4.0	Business Validation or input validation and error states
5.0	Use case diagram/DFD's/Task flow diagram
6.0	Use case

RCN Template :

Project Name	Gmail				
Module Name	Login				
Prepared By					
Prepared Date					
#	Requirement Specification Reference	Clarification Required	Clarification Provided	Clarification Provided By	Clarification Provided Date
01	FRS/Usecase	Forget Password			

Test Design

After understanding various requirement documents, those requirements will design as a 2 documents, such as (a) Test Scenario and (b) Test cases

- What to test (or) one unique test condition (or) one line order of requirement is called Test Scenario, which define High Level Test Design of the requirement.
- How to test (or) step-by-step process is called Test case, which define Low Level Test Design of the requirement.

Example 1: Test Scenario for waterbottle

(1) Verify cap either sealed or not

(2) Verify capacity of water bottle

(3) Verify purity of water

(4) Verify design and color

(5) Verify company label

(6) Verify reusable or use and throw

(7) Verify quality of water bottle

(8) Verify manufacturing date and price.

Example 2: Test Scenario for pen:

(1) Verify the type of pen, whether it is ball point pen, ink pen or gel pen.

(2) Verify if pen is with cap or without cap

(3) Verify click operation

(4) Verify grip whether is it comfort or not

(5) Verify smoothness of writing

(6) Verify ink color

(7) Verify refel type or ink type

(8) Verify reusable or use and throw

(9) Verify company name and logo

(10) Verify pen design and color

Example 3: Test Scenario for Lift:

(1) Verify type of door's, either manual or automatic

(2) Verify capacity

(3) Verify speed in between every flow

(4) Verify buttons outside and inside working or not properly

(5) Verify intimations

(6) Verify power supply

(7) Verify display floor number

(8) Verify emergency alarm

(9) Verify light and fan functionality

Example 4: Test Scenario for washing machine :

(1) Verify capacity

(2) Verify model (top load or front load)

(3) Verify functionality

(4) Verify power supply

(5) Verify speed

(6) Verify design and color

(7) Verify it is automatic or not

(8) Verify driyer

IEEE829 Test Case Format

A	B	C	D	E	F	G	H	I	J	K
TestCase Template										
	Project Name	Gmail	Reviewed By							
	Module	Login	Reviewed Date							
	Document References	FRS	Approved By							
	Author		Approved Date							
	Created Date									
TestCaseID/Name	Testcase_Description	Pre_Condition	StepNo	Step Description	Input Fields	Expected Result	Actual result	Testcase Result	Testcase Priority	Comments

Project Name : Name of the current project

Module Name: Name of responsible module

References: Name of requirement documents, FRS/Usecase

Author: Name of Testing Engineer

Created Date: Date of testcase designing

TestcaseID/Name: One unique id or name to identify document in future.

TestcaseDescription: Name of Test Scenario

Pre_Condition: List of steps which need to complete before requirement

Stepno: Serial number

StepDescription: How to test, step by step clear requirement

Input Fields: Prep
are input data by using test case design techniques

Expected Result: Client expectation from requirement document

Actual Result: After execution final result update as a actual result

Testcase Result: If expected value=actual value, then result should be pass, else fail

Testcase Priority: It is used to define importance of testcase

Comment : Comments if any.

Reviewed By, Reviewed Date: After testcase design reviewed by Test Lead and date also updated by Test Lead.

Approved By, Approved Date: Testcases should be approved by Test Manager or Project Manager.

Black Box Design Techniques/Test Data (or) Input Techniques

Test Data/Design technique used to prepare data for testing which can cover each and everything of the functionality.

There are 5 types of Test Data or Test Design or Testcase Design techniques:

1)Boundary Value Analysis (BVA)

2)Equivalence Class Partition (ECP)

3)Decision Table (DT)

4) State Transition (ST)

5) Error Guessing (EG)

1. Boundary Value Analysis (BVA) :

- It defines the size and range of the inputs.
- While using BVA technique, starting value consider as a Lower Boundary and ending value consider as upper Boundary.
- Instead of checking all possibilities, lower boundary check with +1 and -1, and ending value check with -1 and +1.

Example :

BVA:

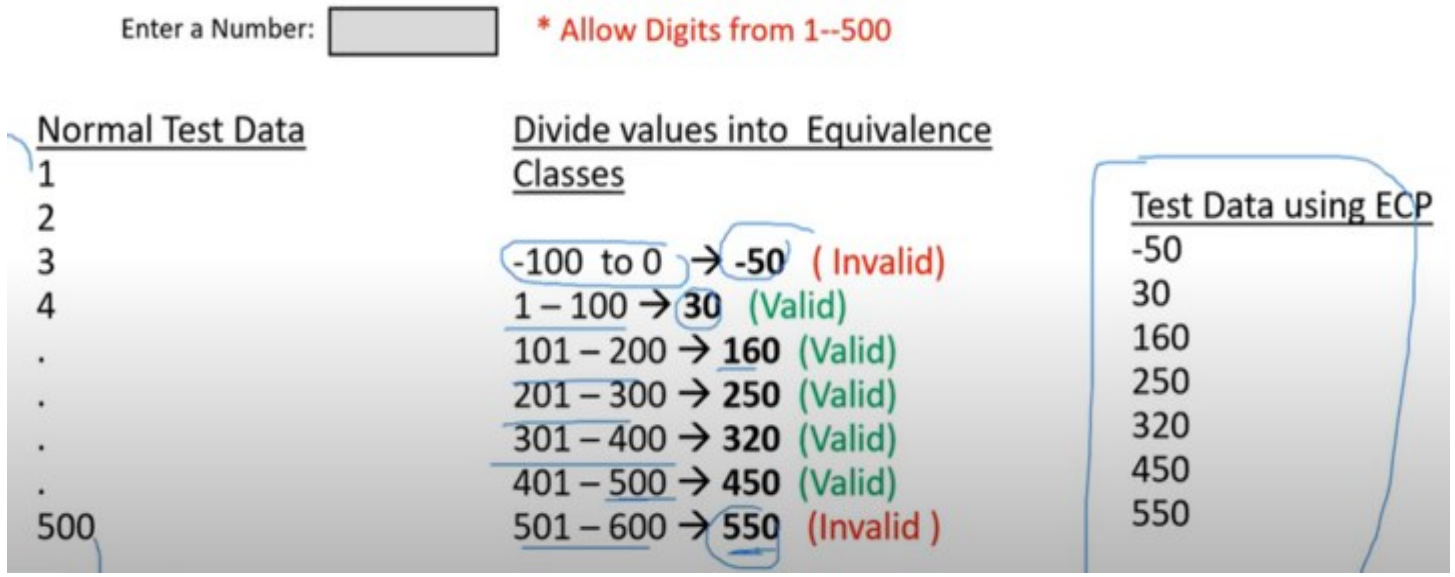
Min = 18 (Pass)	Max = 35 (Pass)
Min-1 =17 (Fail)	Max-1 =34 (Pass)
Min+1 =19 (Pass)	Max+1 =36 (Fail)

2. Equivalence Class Partition (ECP) :

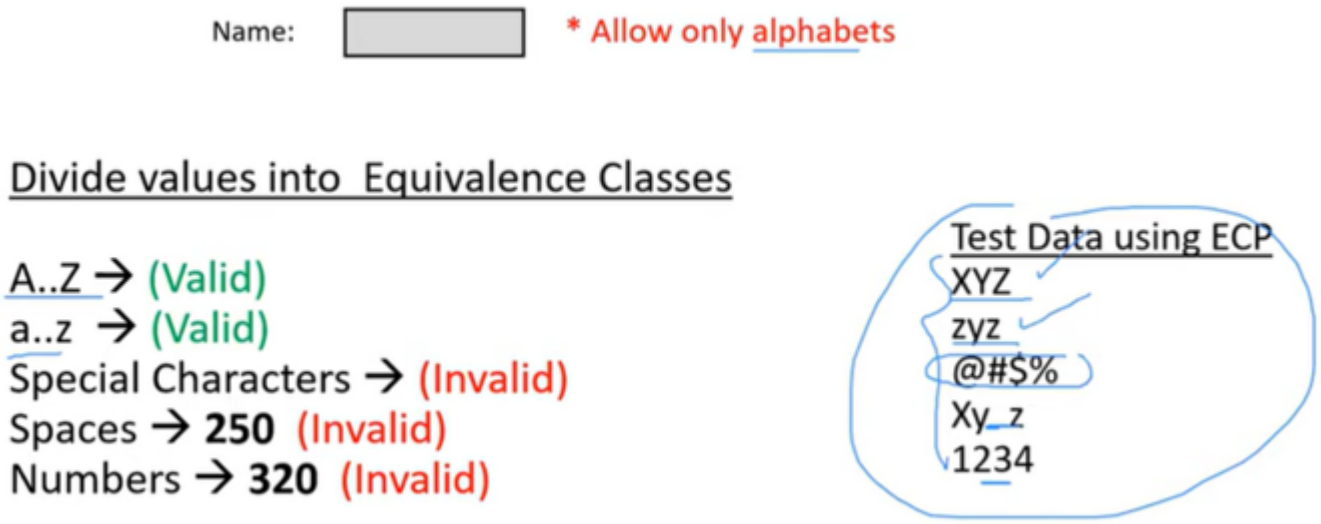
- It defines the type of inputs/values such as alphabets, numeric, blank spaces, special characters etc.
- In ECP, first we have to divide what is valid and what is invalid, then divide into group.
- But while making groups, instead of checking all possibilities, starting, middle, ending values will be consider.

Example:

Example:



Example 2:



Example 2:

Verify Radio button:

Value	Expected
Select one way	one way should be active, Round trip and Multi city should be inactive.
Select Round trip	Round trip should be active, One way and Multi city should be inactive.
Select Multicity	Multicity should be active, One way and Round trip should be inactive

Example 3:

Decision Table for Upload Screen

You can upload only '.jpg' format image
file size less than 32kb
resolution 137*177.

upload photo | upload | *upload .jpg file with size not more than 32kb and resolution 137*177

Conditions	Case 1	Case 2	Case 3	Case 4	Case 5	Case 6	Case 7	Case 8
Format	.jpg	.jpg	.jpg	.jpg	Not .jpg	Not .jpg	Not .jpg	Not .jpg
Size	Less than 32kb	Less than 32kb	>= 32kb	>= 32kb	Less than 32kb	Less than 32kb	>= 32kb	>= 32kb
resolution	137*177	Not 137*177	137*177	Not 137*177	137*177	Not 137*177	137*177	Not 137*177
Output	Photo uploaded	Error message resolution mismatch	Error message size mismatch	Error message size and resolution mismatch	Error message for format mismatch	Error message format and resolution mismatch	Error message for format and size mismatch	Error message for format, size, and resolution mismatch

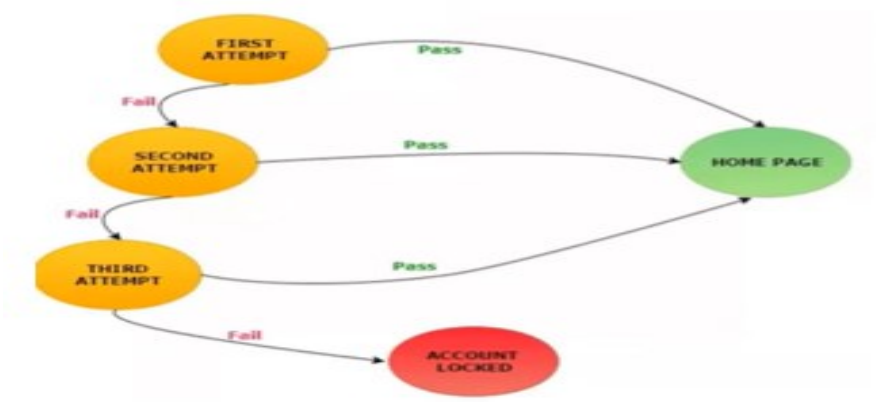
4. State Transition :

Every application has different states(user interface), the state will navigate from one state to another state based on user operations.

- In this technique, we provide positive as well as negative input test values to evaluate the system behaviour.

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"		

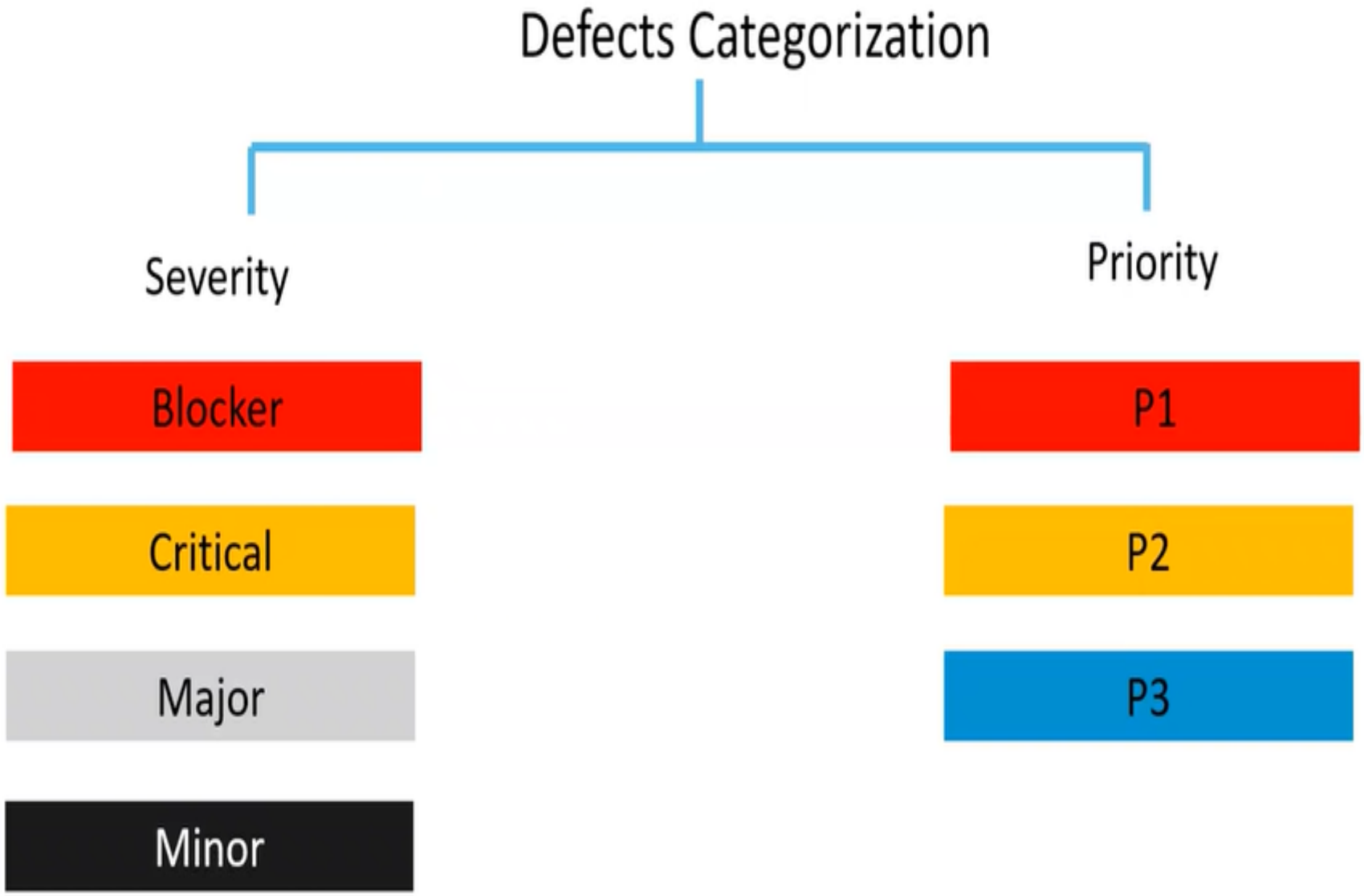
5. Error Guessing :

- 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 testers analytical skills and experience.

Examples are:

- Submitting forms without entering values.
- Entering invalid values in the fields etc.

Priority and Severity



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.

Priority: How soon we have to fix the defect. Urgency to fix the defect.

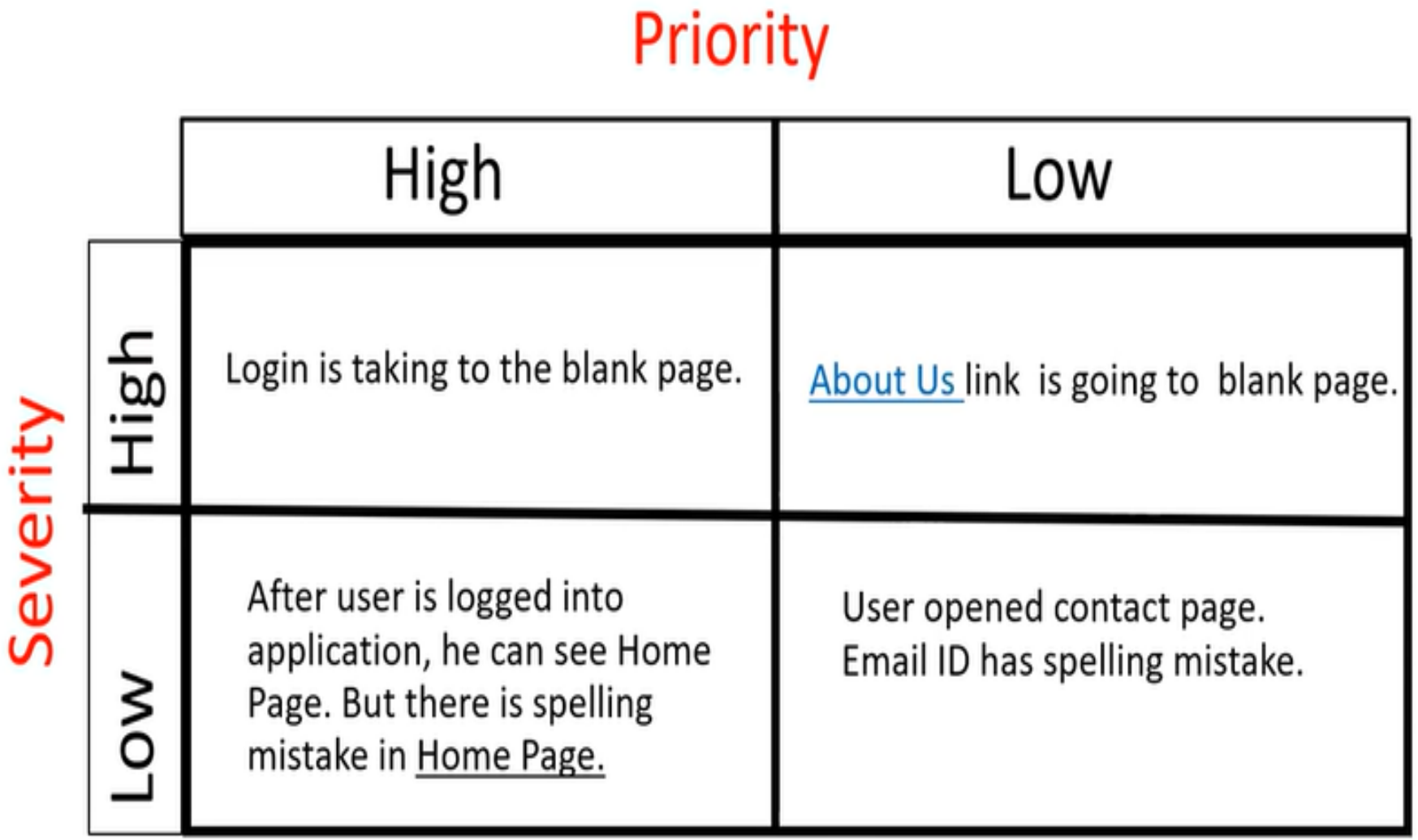
Defect Priority

- 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.

Note:

- Severity and Priority will be provided by Testers, but Priority can be changed by Developer, BA also, but Severity cannot be changed, it should be provided by Testers only.

High severity, priority and low severity, priority defects



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

TestCase Design Based on UseCase Document

Usecase also one kind of requirement document which is prepared by BA, it has more elaborated information than FRS. It describe functional requirement.

Usecase contains 3 items.

- Actor, which is the user, which can be a single person or group of people, interacting with a process.
- Action, which is to reach the final outcome.
- Goal/Outcome, which is the successful user outcome.

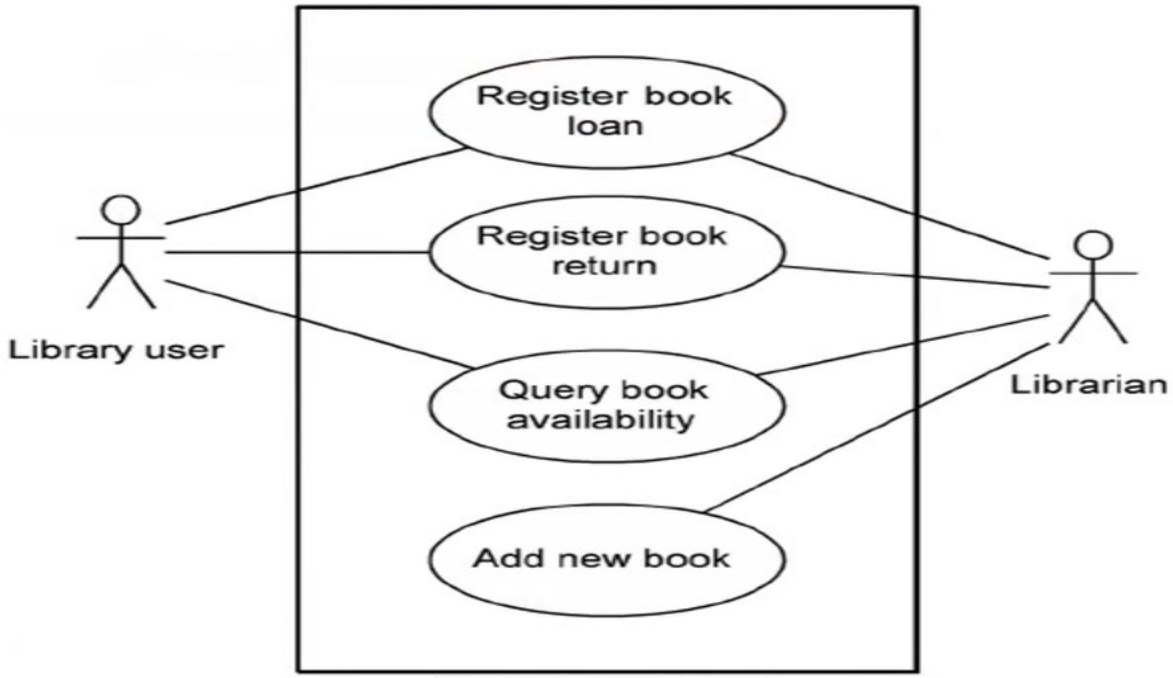
Library Application Book Issue Module Usecase Document:

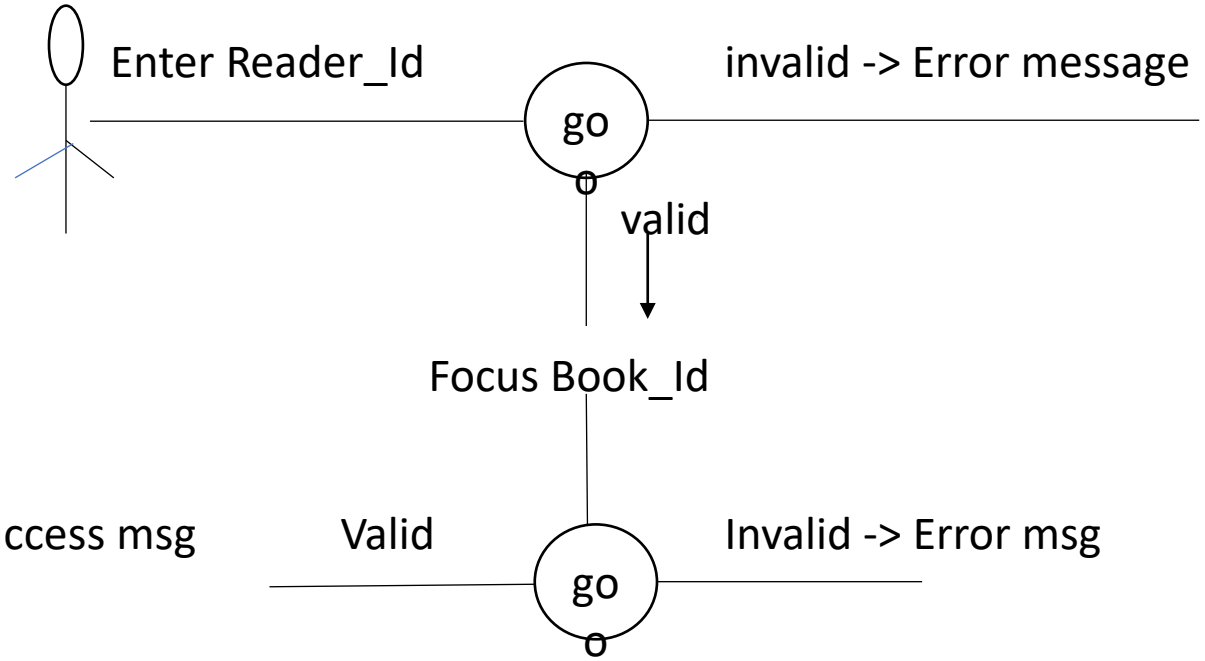
1) Usecase Id : UC_Book_Issue
2) Description : book issue operation will be used by library employee to issue a book to valid reader
3) Actors : book issue operation can take 2 inputs such as “Reader_Id” and “Book_Id”
4) Precondition : readers are already registered and book information is already feeded

1) Events list :

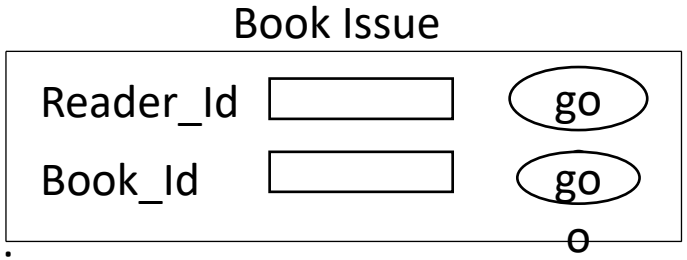
Events	Outcome
Enter Reader_Id ,click go	Book_Id focused for valid reader and error message for invalid reader.
Enter Book_Id, click go	Book issued message for valid Book_Id and when book is available. -Sorry books already issued message, when book is valid but books unavailable. - Wrong Book_Id message when Book_Id invalid

1) Active flow diagram (AFD)





7) Prototype :



8) Post Condition :

After book issue operation, corresponding library employee can go to next or logout.

9) Alternative Events “

10) Related Usecases :

UC_Reader, Register_UC, Book_Feed.

Prepare testcases for Book_Issue operation.

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

Review Testcases

Once testcases are prepared by Testing Engineer, in order to check whether testcase prepared as per client requirement or not, perform below reviews.

1. Self review
2. Peer review
3. Lead review
4. Manager review
5. Client review

Tips to prepare good Testcases (or) How to prepare effective testcases

(1) Understand requirement from end to end.

(2) Identify every requirement scenario as a test scenario.

(3) Prepare test case in Ms-Excel.

(4) Follow IEEE829 Testcase format.

(5) Every test scenario should be cover as testcase.

(6) Testcase should be understandable/reusable.

(7) Every testcase might be cover positive/negative steps.

(8) Prepare test data for every testcase by using BVA/ECP/DT.

(9) Test data should be cover min 4 to max 10 steps.

(10) Every testcase should be reviewed by Test Lead.

(11) Testcase should be update into RTM document.

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).

automation).

Whenever testcases are prepared and reviewed by Test Engineer, as per release date, build released by developer to compare expected value and actual value, every testcase execute on build, called

Test Execution.

During test execution, tester need to follow below format.

(1) Formal Meeting

(2) Configuration Repository Management (CRM)

(3) Test Execution Levels

1. Formal Meeting

Test Execution starts with formal meeting, which is conducted by Product Manager along with development team and testing team, to confirm test management tools, bug tracking tools and version control tools.

(a) Test Management Tool :

This tool used to perform testing activities like Test Plan, Requirement documents, analysing requirements, prepare scenarios and cases, update reports, test execution etc.

Eg : ALM/QC, Jira, Q-Test, QA Coverage.

(b) Bug Tracking Tool :

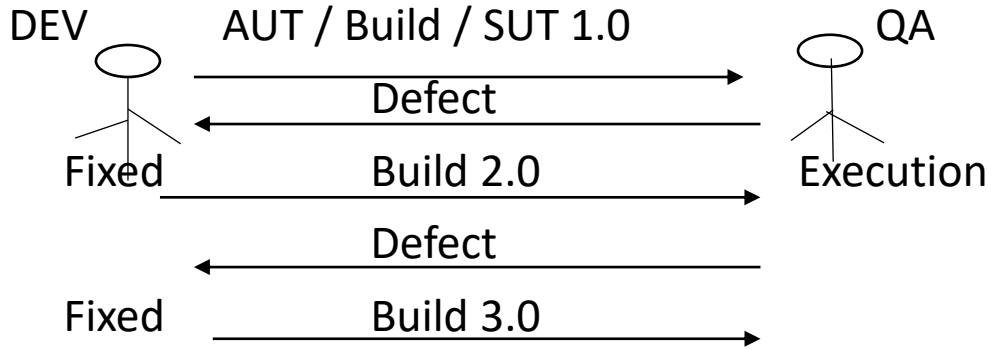
is used to track defects in between developers and testing engineers called Bug tracking tool.

Eg: ALM/QC, Jira, BackLog, RedMine, etc.,.

© Version Control Tool :

is used to maintain build versions in between developers and testing engineers.

Eg: Git, ANT , Maven.



2. **CRM** :

Every software engineer to store working related document, Project Manager, will create folder structure in server system called Configuration Management Repository. It contains mains 3 folders such as-

i. Development base

ii. Soft base and

iii. Test base.

- Development base contains development related documents such as development requirements, plan, design,coding, unit and integration testing documents.
- Softbase contains build versions in between development and testing engineer.
- Testbase contains test related documents such as FRS, Use case, Test scenarios, test case, test data, RTM, defect documents.

Every software engineer, connecting to server performs operations based on their roles and responsibilities.

Developer has read and write permissions on development base and **softbase** has only read permission on test base. Tester has read and write permission on test base and **softbase** and only read permission on development base.

3. **Test Execution Levels** :

Smoke Testing, Sanity Testing, Re/Regression Testing

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.

IEEE829 Defect Report Template

Project Name										
Defect_Id	Defect Description	Reproductable [Y/N]	Reproductive steps	Defect Severity	Defect Priority	Defect Status New/Reopen	Detected By	Dected Date	Detected in Version	Fixed By

- Project Name: Name of the current project
- Defect_Id: One unique id or name to identify document in future.
- Defect Description: Prepare step by step description of defect.
- Reproducable [Y/N]: While executing testcases if defect producing everytime, update status as “Yes” or while executing defect producing only sometimes, then update status as “No”.
- Reproducable Steps: If reproducable is “Yes”, then attach testcase
  If reproducable is “No”, then attach testcase and screenshot.
- Defect Severity: It define impact of the defect as Blocker, Critical, Major and Minor severities.
- Defect Priority: It indicates urgency to fix defect as a High, Medium, Low.
- Defect Status: If defect occurs as first time, then update status as “New”. Once defect is fixed also, same defect is repetiting again and again, then update status as Reopen.
- Detected By: Name of the testing engineer.
- Detected Date: Date of defect identified.
- Detected in Version: Name/Number of build version
- Fixed By/Fixed Date: Both will be updated by developer after defect fixing.

- Date of Closure: Once defect fixed by developer, during regression testing, if defect is fixed as per requirement, then update date of closure or update status as reopen by testing engineer.

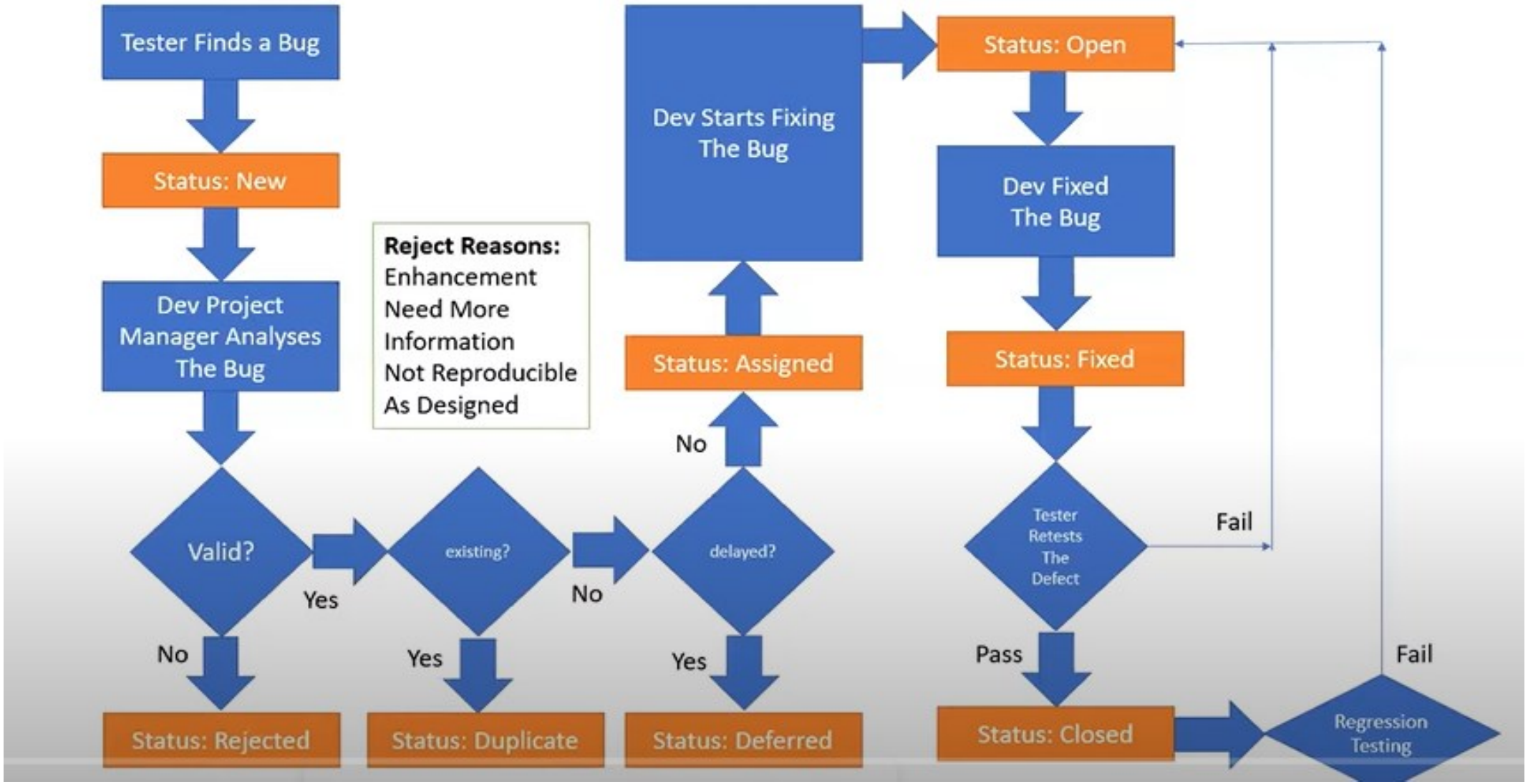
Defect Resolution

- 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

Note:

What kind of opinion did developer having on defect is called Defect Resolution.

Bug Life Cycle



Once defect identify by Tester, those defect will be documented in a Defect Report Template or Bug Tracking Tool, with a status “New”.

Once all new defects reported to developer, then developer will analyse

- If reported defect is invalid, update status as “Rejected”.
- If reported defected is valid, then based on priority, developer update status as “Open”.
- If developer need clarification about the defect, he update the status as “Hold”.
- If defect is postponed to next phase, then status will be “Differed”.
- If defect if already reported, then status will be updated as “Duplicate”.

Once this analysing is completed, developer start fixing defects. Once defects are fixed, then developer update status as Fixed or Resolved.

Once all defects are fixed, before release to testing engineer, modify build version and update as a modified build.

Tester will start Re/Regression testing on build to confirm either defect fixed as per client requirement or not. If defect is fixed in Regression testing, update status as “Close”, else status will be updated as “Re-open”.

If any new defects identified in the modified build, the same will be documented with the status “New”, and the cycle will continue till all defects are closed.

Note:

suppose reported bug is fixed and closed in one build and then in upcoming builds this defect is detected again , then we should raise a **new bug by specifying new build number** , but we cant reopen the closed bug.

Test Closer / SignOff

Whenever all testcases are executed successfully and all major defects are fixed, before signoff current project, Test Lead will perform final Regression testing on doubtful functionalities.

During this testing, if any defect identified then it called as “Golden Defects”. Those defects as soon as possible report to developer during signoff from the project. Test Lead will prepare test summary document.

Final Test Summary Documents:

- Test summary document
- Testcase summary document
- Test execution summary document
- Defect summary document.

Test Metrics:

Whenever we do task we need to measure it. We have to track the work. To know how much we are progressing and where are we now and where we have to go. It helps the management team to check how much work is going , how much is pending, status of the work.

Before writing metrics we need to collecting the data. i.e to calculate the metrics we need to collect the data like - That data is –

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

Defect Related Matrics :

- **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.

Principles of Software Testing:

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.

8. Absence of Error Fallacy: If we find mor no.of defects without fullfuilling user needs, then those defects are not helpful.