Testing as a Process

and

Introduction To Unit Testing

Contents

SDLC

Every software development process has a life cycle known as Software development life cycle.This life cycle has different stages

Planning Stage :-

Planning stage involves creating scope of application ,financial as well as technical, how many people to involve ,skills required, budget for entire project ,duration in which all the process has to be completed and software to be handed over to the client

At the planning stage itself certain key issues are discussed with the team

- Objective of intended software system
- different stake holders have to be identified ,
- role of each person involved with the process of development ,
- financial limitations with the team and customer
- Business risk involved

Planning process can include discussion with stake holders ,and team that is going to develop the software

[

Who Is Considered a Stakeholder?

While every development project is unique, there are some universal categories that can be used to guide stakeholder identification.

End users and beneficiaries

These are the people who will be most directly affected by the software. Their buy-in is essential. No matter how flashy or efficient software is, if end users don't like it they won't use it. End users fall into **three main groups:**

Direct users

Those who will use the software directly are usually most concerned with how it will fit into their current workflows. They want to know that it solves a significant problem or otherwise makes their job easier.

Secondary users

Direct users interact with the software itself. Secondary users rely on the products of the software. New software needs to produce results in a format that fits into secondary users' workflows. Forgetting about this group can cause one problem while solving another, like suddenly generating reports in a format secondary users can't **integrate into their analytics**.

Beneficiaries

These are all the people affected by the software's products. The term encompasses a huge base of customers and vendors who focus more on results than process. Their input should revolve around the services or information the software will provide.

]

Analysis stage:-

At this stage all the requirements related to develop the software are gathered ,analyzed and discussed with team and stake holders

Gathering requirements ,involves

talking to stake holders and understanding complete requirement of users

Consultation can be made with a domain expert of different people involved currently in the existing process

A complete analysis and feasibility report ,metrics and analysis documents form critical part of development process ,as they can be accessed anytime to take a design decision, check deviation from intended objective , analyze what is possible within available resources and what exactly the customer wants

Design Phase :=
This phase involves design the actual system that needs to be developed .Designing can involve different methodologies like creating a map of the system ,use cases ,UML Diagrams ,developing prototypes .
The purpose of this phase is to ensure that all requirements analyzed during requirement analysis phase are finally filtered and put on paper ,or prototyped and the team can visualize the final outcome that they will get once the development process is over

Development phase:

In this phase ,the software that has been designed is put to code .Different programming languages and technologies can be used for coding ,This depends on methodology to be used for programming .Two popular methodologies are procedural and Object Oriented programming
Most of the time it is decided at the design stage itself ,which methodology has to be used to code for intended software system.
Systems are mostly developed today around Object Oriented Programming

Different programming languages that target Object Oriented Programming are
C++,Java,C#

Testing and Integration phase :

This phase involves testing of system developed.The objective of testing is to identify bugs and issues in the system.Is the intended system as per requirements or has it deviated from the intended purpose .

The intention of testing is not to stall the development process ,or negate the system built ,but to identify bugs that can damage the hole process of development .

The tesing process has it’s own lifecycle called software testing life cycle

Deployment and Maintenance :

The system developed through the software development process has to be delivered to the client ,either as a package if it is a x`x`desktop application ,or through different stores ,if it is a mobile application or through web it is a web application

Maintenance involves different things like

- updating software from time to time
- providing patches to plug undetected bugs
- cleaning up unwanted data collected through interaction of user with software

- monitoring software for security related risks
- scaling application to meet increasing customer demand

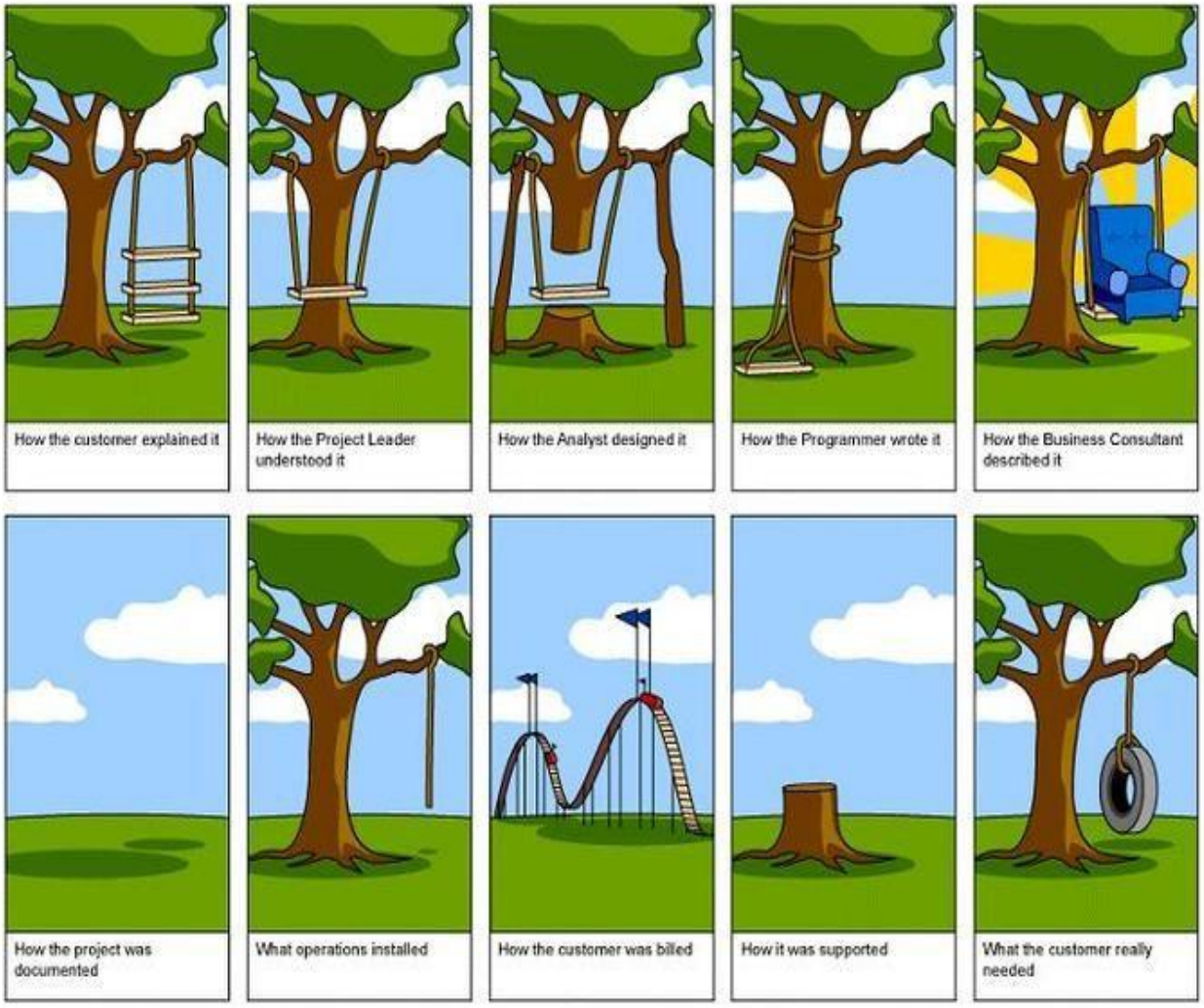
- enhancing security needs

The six stages of software development appear based on type of model selected for development

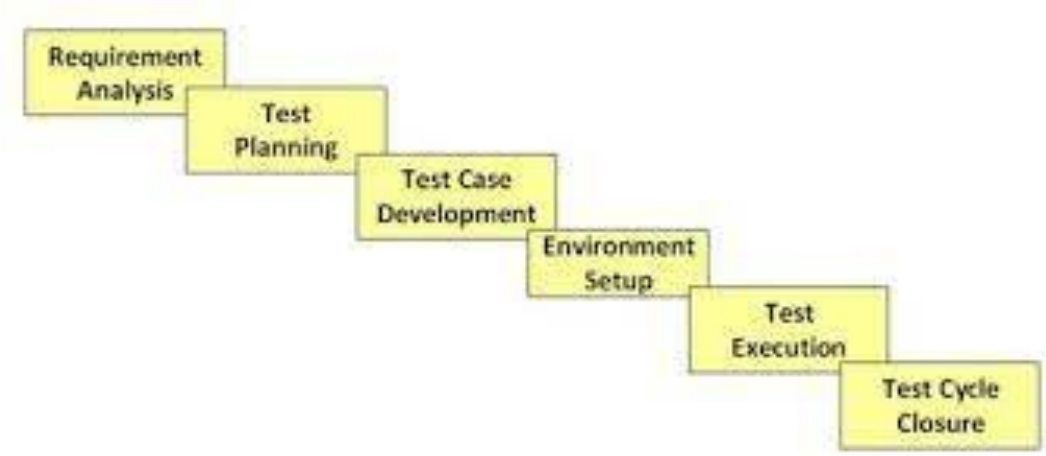
different models that can be used to implement are Waterfall model, Spiral Model ,Win-Win Spiral modal ,agile development

Let’s Discuss

Why Testing is important


How the customer explained it
How the Project Leader understood it
How the Analyst designed it
How the Programmer wrote it
How the Business Consultant described it
How the project was documented
What operations installed
How the customer was billed
How it was supported
What the customer really needed

Software Testing Life Cycle



Requirement Analysis

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

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

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

Test Planning

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

Test Case Development

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

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

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

Test Environment setup

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

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

After the creation of Test data the Smoke testing is executed on the build in order to check whether the basic functionalities are working

fine or not. This phase can be done in parallel with the Test Case Development phase.

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

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

Test Execution

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

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

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

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

Test Closure

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

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

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

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

Types of Testing

There are different types of testing involved at different stages .These are unit testing ,Integration testing ,System testing and acceptance testing

Let us discuss

Quality Control vs Quality Assurance

Types of testing

Unit Testing

It is a type of software testing where individual units or components of a software are tested. The purpose is to validate that each unit of the software code performs as expected. Unit Testing is done during the development (coding phase) of an application by the developers. Unit Tests isolate a section of code and verify its correctness. A unit may be an individual function, method, procedure, module, or object.

In SDLC, STLC, V Model, Unit testing is first level of testing done before integration testing.

Integration Testing

Integration testing is the second level of the software testing process comes after unit testing. In this testing, units or individual components of the software are tested in a group. The focus of the

integration testing level is to expose defects at the time of interaction between integrated components or units.

Unit testing uses modules for testing purpose, and these modules are combined and tested in integration testing. The Software is developed with a number of software modules that are coded by different coders or programmers. The goal of integration testing is to check the correctness of communication among all the modules

System Testing

In system testing the behavior of whole system/product is tested as defined by the scope of the development project or product.

It may include tests based on risks and/or requirement specifications, business process, use cases, or other high level descriptions of system behavior, interactions with the operating systems, and system resources.

System testing is most often the final test to verify that the system to be delivered meets the specification and its purpose. It is carried out by specialists testers or independent testers.

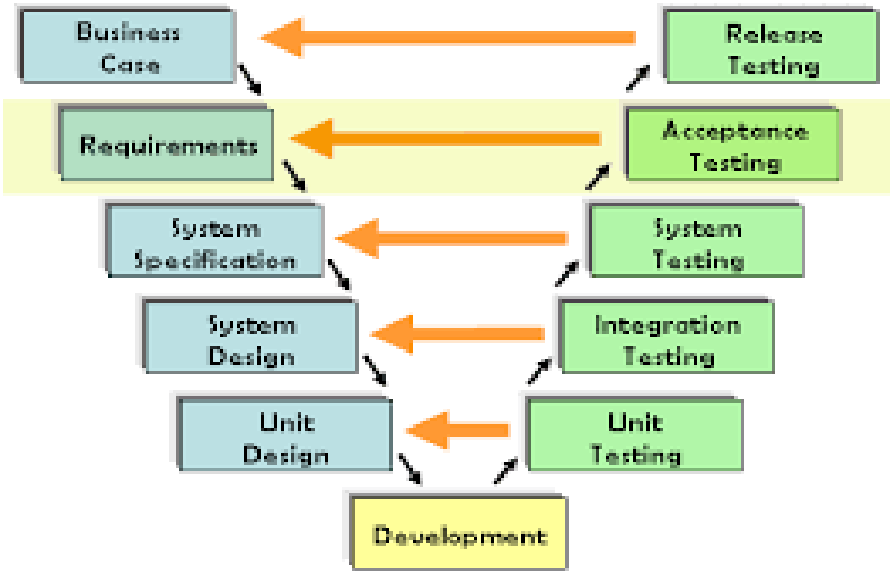
System testing should investigate both functional and non-functional requirements of the testing.

Acceptance Testing

Acceptance testing, a testing technique performed to determine whether or not the software system has met the requirement specifications. The main purpose of this test is to evaluate the system's compliance with the business requirements and verify if it is has met the required criteria for delivery to end users.

There are various forms of acceptance testing:

- User acceptance Testing
- Business acceptance Testing
- Alpha Testing
- Beta Testing



* **Regression testing** is re-running functional and non-functional tests to ensure that previously developed and tested software still performs after a change.[2] If not, that would be called a *regression*. Changes that may require regression testing include bug fixes, software enhancements, configuration changes, and even substitution of electronic components.[3] As regression test suites tend to grow with each found defect, test automation is frequently involved. Sometimes a change impact analysis is performed to determine an appropriate subset of tests

Unit Testing

Unit testing is about verifying the behavior of the smallest units in your application.

Technically, that would be a class or even a class method in object-oriented languages, and a procedure or function in procedural and functional languages.

Advantages of Unit Testing

The process becomes agile

This is the main benefit of unit testing. When you add more features to any software, you might need to make changes to the old design and code, and this can be expensive as well as risky. If you use the unit testing methodology, then this can save a lot of time and can make the whole process much faster and easier.

Quality of Code

Unit testing significantly improves code quality. It helps developers to identify the smallest defects that might be present in the units before they go for integration testing.

Find Software Bugs Easily

Unit testing helps identify all kinds of issues with the software at a very early stage. Software developers can then work on those issues first before progressing any further. The main advantage of this is when the issues are resolved at an early stage, no other part of the software is impacted. This leads to increased efficiency, reduced downtime, and reduced costs that would otherwise arise as a result of the whole design process stalling.

Facilitates Change

Refactoring the code or updating the system library becomes much easier when you test each component of the software individually. If there are any problems, they are detected early on and making changes to the system thus becomes much easier. The accuracy of each unit is verified before it moves on to the next phase. This means that the unit is proven to be in proper working order before it’s integrated with other modules.

Provides Documentation

Unit testing takes into consideration the documentation of the entire system. Developers who want to learn about the functionality of a certain program or application can easily learn about the system by reading the documentation of each individual module. It allows them to develop a thorough understanding of the system and what each individual component does.

Debugging Process

The debugging process can be simplified to a great extent by unit testing. If a certain test fails, then only the latest changes that have been made to the code need to be debugged.

Design

Unit testing allows software developers to actually think through the design of the software and what has to be done before they write the code. This can help them to stay focused and can also help them to create much better designs. Testing a code early on can help to define what that piece of code is really responsible for.

Reduce Costs

Any problems or bugs in the system are identified in the early stages through unit testing, and because of that the cost of bug fixes is significantly reduced. If these bugs are discovered later, then it will be much more expensive to fix them.

When bugs are detected at the later stages, they are usually the results of many changes that have already been made to the system. If the software has already been developed, finding out the exact code that caused these bugs will be a major problem.

Why Do Unit Testing ?

Unit tests make it safer and easier to refactor the code by putting tests into place that make sure refactoring occurs without problems and disruption. It takes the risk out of changing older source code.

Doing unit tests is essentially doing quality assurance of the code. It shows problems and bugs before the product has an integration test. Creating a testing process before the coding is completed solves issues and challenges creators to write better code.

UT helps find problems and resolve them before further testing so they won't impact other bits of code. This includes bugs in a programmer's execution and issues with a specification for the unit itself.

UT allows the refactoring of code and makes integration simpler. It finds changes and helps maintain and adjust code, reducing bugs and defects, and verifying the accuracy of each unit. It makes sure the later testing is easier once the integration process begins.

This type of testing maps a system and creates documentation. It helps understand the unit's interface.

UT makes the process of debugging easier.

UT forces better code and design whether you are using C#, Java, Python, JavaScript, or Php. It means you have a well-defined code with high cohesion.

Using a unit test and good unit testing tools means you reduce the overall cost of a project. Early bug detection means fewer late changes and easier to spot issues than if it is done at a later stage.

Every process has some disadvantages

With UT, you have to increase the amount of code that needs to be written. You usually have to write one or more unit tests depending on how complex things are. It is suggested to have at least three so you don't just get a yes and a no that contradicts each other. While the test code should be fairly simple, this testing method is still more work and more code which means more hours and more cost.

Unit tests are problematic when you need to test your user interface (UI). They are good for when you need to test business logic implementation but not great for UI.

There is a school of thought that unit tests are problematic for a product's structural design. They solidify the structure of code which means change can be problematic when needed.

In comparison to those who say UT improves code, others say it makes it worse and ends up adding indirection that is pointless. Changing code and adding new code can mean navigational issues and more time spent before integration testing is even started.

UT cannot and will not catch all errors in a program. There is no way it can test every execution path or find integration errors and full system issues.

Unit tests have to be realistic. You want the unit you're testing to act as it will as part of the full system. If this doesn't happen, the test value and accuracy are compromised

Creating Test Cases

A **TEST CASE** is a set of actions executed to verify a particular feature or functionality of your software application. A Test Case contains test steps, test data, precondition, postcondition developed for

specific test scenario to verify any requirement. The test case includes specific variables or conditions, using which a testing engineer can compare expected and actual results to determine whether a software product is functioning as per the requirements of the customer.

Test Cases are based on test scenarios

For example to check the functionality of a Login Page for a customer

Test Case 1:Check the Login functionality with valid login id and password

Test Case 2:Check the Login functionality with invalid login Id and password

Test Case 3 :Check the Login with empty fields

Test Case 4:Leave the login id field blank

Test Case 5: Leave the password field blank

There is test data associated with every test case

Test Data is values supplied to test case

A Sample Test case

Test Case Id	Test Case	Test Steps	Test Data	Expected Result	Actual Result	Pass/Fail
1	Check Login with valid Email /Password	1.Login to site 2.Enter user Id 3.Enter Password 4.Click Submit	User ID=Peter Password =peter123	User Logged in successfully	Invalid details	Fail

Exercise

creating test cases