Programming Fundamentals

# **SoftwarE EngineerinG:**

**Software** is a set of programs written to perform a desired functionality. Engineering is designing and building something that serves a particular purpose and finds a cost-effective solution to problems.

Software Engineering uses various tools and methods required for designing, developing, testing, and maintaining software.

# **Software Development life Cycle:**

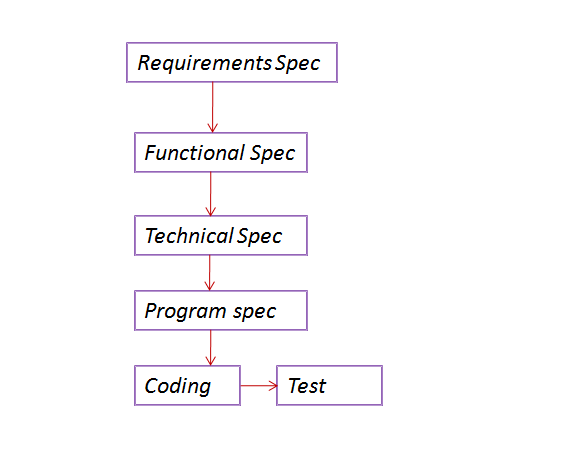
SDLC is a process of developing software that follows methods to ensure that Quality, reliable, and cost-effective software is developed.

There are different types of SDLC model:

* Waterfall or sequential model
* V model
* Iterative and incremental models

## Waterfall Model:

In the traditional waterfall model, which is a linear or sequential model, each activity in the software development life cycle is complete before the next stage begins.



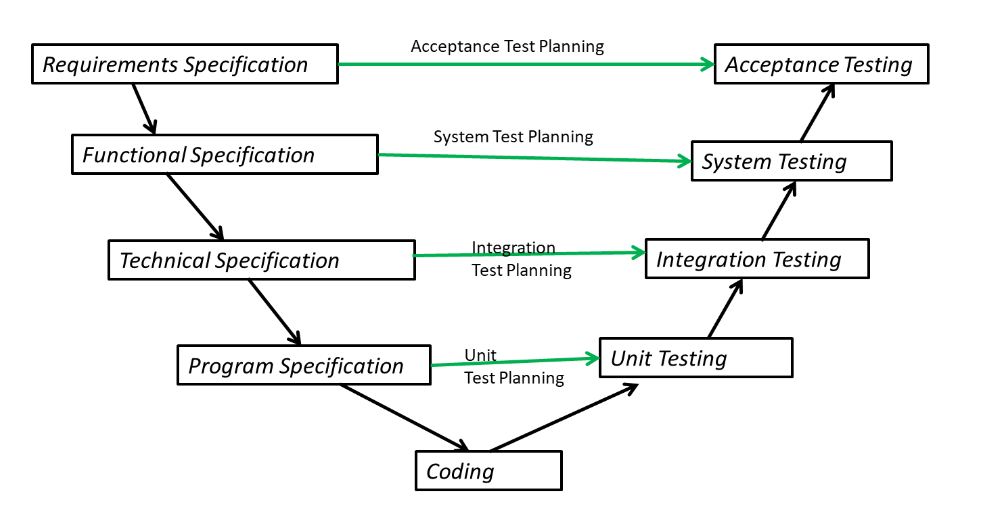
Customer

* Each stage is completed and frozen before moving on to the next.
* Testing is carried out only once the code has been fully developed
* There are no quality checks done other than the testing at the end of the lifecycle.
* Customer sees the product only after the Testing phase.

### Disadvantages:

* Evaluation of the software at the end of the life cycle against user requirements. Pushing the testing to the end of the life cycle is risky as the test cycles get squeezed.
* All testing activities come late in the project. When testing is done so late in the project it can reveal costly errors.
* Project plan includes testing at later stages and physical deliverables such as code can be tested.
* Difficult to incorporate changes late in the project life cycle.
* Finding faults and fixing them at later stages may cause the system to become unstable and even unusable.

## V - Model



The V-model is an extension of the waterfall model but this time from the viewpoint of testing activities that need to be completed later in SDLC. It follows the principle of early testing. So, for every development activity, there will be a corresponding testing activity. Test phase for a given level begins along with corresponding software development activity.

* Testers become involved in reviewing documents as soon as drafts are available in the development life cycle. So during the Requirements phase, testers review the Requirement specification - checking whether the requirement has enough details to be tested, there is no ambiguity, and if non-functional requirements are mentioned as well.
* When the Functional Specification phase begins, Testers get involved, reviewing the drafts and again making sure all the features have been captured, are clear and there is enough details to test the Features. The Functional Test Plan is the outcome of this phase.
* During the other phases downwards, which is primarily a Development Team effort, developers come up with the Technical Specification & a corresponding Integration Test Plan. The Integration Test Plan details how the modules or Programs will be integrated, data that will be passed through the interfaces and the approach to Integration.
* Then the Program Specification and Unit Test plans are developed.

### Validation and Verification

The V-model introduces two checks called Verification and Validation as Early testing activities.

There are two checks for each work product - a static testing check and dynamic testing check.

**Verification** is checking if we are **building the product in the right way** and it meets the requirements set out for it. All reviews are Verification activities.

Reviews are done to:

* Check if all the requirements are covered
* Check internal standards, guidelines etc
* Verify completion criteria for the work-product
* Check if there is sufficient info to build the next product in the life cycle.

**Validation** checks whether the product meets the user's needs. This answers the question "**did we build the right product**", Does the product meet the user requirements?

When a customer looks at a system we are building on their behalf, they are much less interested in comparing it to the documented requirements and much more concerned about whether the product is going to solve their problem, be usable by their people, and run on their computers.

|  |  |
| --- | --- |
| Verification | Validation |
| Checks if the product meets the specific requirements of that phase | Checks the final product to see if it meets the user needs |
| It tests the requirements, architecture, design and code of the software | It tests the usability, reliability and functionality of the software |
| Are we building the product right? | Are we building the right product? |

### Advantage of V-model

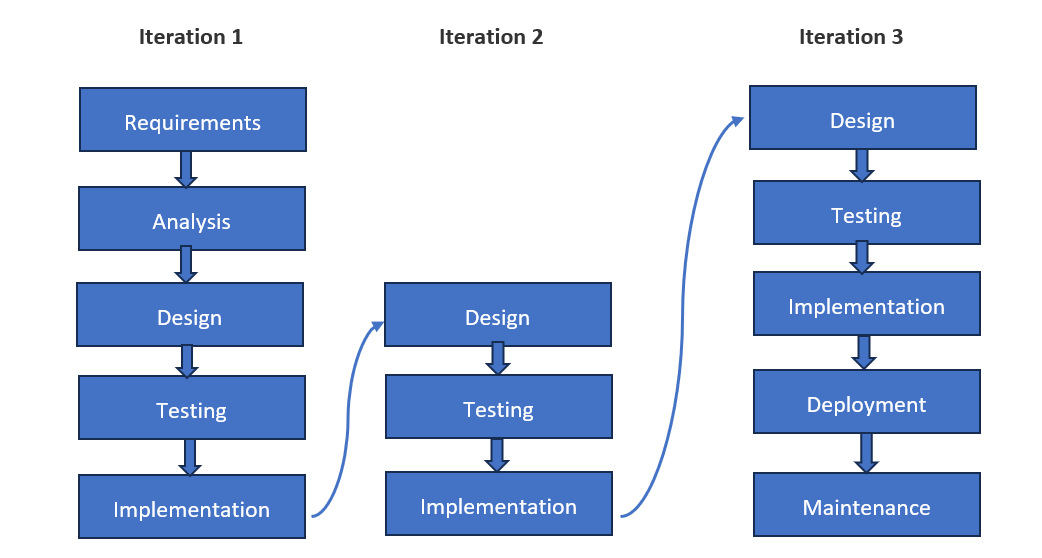
* Each work product is tested/reviewed at the end of each stage.
* Early test design: Planning for testing starts with each work-product.
* Defects can be identified as early as possible in the life cycle.
* Testers are involved in reviewing the requirements.

### Disadvantages of V-Model

* Pushes the validation of the system by the user representatives right to the end of the life cycle. i,e, User Acceptance Testing
* If customer needs were not captured accurately or if they changes, then these issues will not be uncovered until the user testing is carried out. Fixing problems at this stage could be very costly; in addition, it is possible that the project could be cancelled altogether.

## Iterative development model

In Iterative model, a sub-set or the most important features are first designed, built, tested and a version is released. Based on customer feedback, features are tweaked, added, dropped and each Release gets bigger and better than the previous. The customer gets a chance to use the product and report changes from the very first Release*.*

****

In Iterative models, like Agile, these stages are more loosely defined, for eg we could do Design and back to Requirements, Coding back to Design for changes etc.  If were writing a book, we would just write a lot, we would review throwing out unnecessary stuff, and start again. Iterations are planned rework.

Summarizing:

* Requirements do not need to be fully defined before coding can start.
* A working version of the code is built in a series of stages, sprints or iterations.
* Each stage encompasses requirements definition, design, code and test.
* Referred to as cyclical as we go round the development life cycle a number of times within the project.

### Advantages

* Involvement of user representatives in testing.
* Having the users represented throughout minimizes the risk of developing an unsatisfactory product.
* The user representatives are empowered to request changes to the software to meet their needs.

### Disadvantages

* Lack of formal documentation makes it difficult to test.
* Developers may make changes without formally recording them. This approach means it is difficult to trace back to requirements or parts of the software that have changed. So we would need a robust process to keep track of the changes.
* Amount of regression testing to be conducted because of changes is more.

## RUP(Rational Unified Process):

Software development process by IBM.

Consists of four steps:

1. **Inception**: The initial Idea or Planning
2. **Elaboration**: Further detailed investigation into resources, architecture and Costs
3. **Construction**: Software is developed, including Testing
4. **Transition**: Software product is released to Customer.

Risk is the primary driver for decisions. Evaluation of Quality continues throughout the development.

The Increments that are produced are larger than what is produced during Agile. Typically would take months rather than weeks to develop.

# Testing

* What is testing and why it is necessary?

Examples of Software Failures:

* 1. Rocket Launch Failures
* 2. Online Transactions:
  + Functional: HMRC Online Tax Filing website where a user could sometimes see the amount the previous user earned.
  + Non-functional: Websites not able to scale, security lapses etc

Incorrect Software can cause harm to

* + People eg: failure of a hospital life support system leading to injury, death
  + Companies eg. incorrect billing systems to loss of money, reputation etc
  + Environment eg. Releasing chemicals or radiation into the atmosphere leading to loss of money, injury, death.

So software failures lead to loss of Money, Time, Business Reputation, Injury and Death.

* Why do failures occur?

## Testing Principles

1. Testing shows the presence of bugs

2. Exhaustive testing is not possible

3. Early Testing

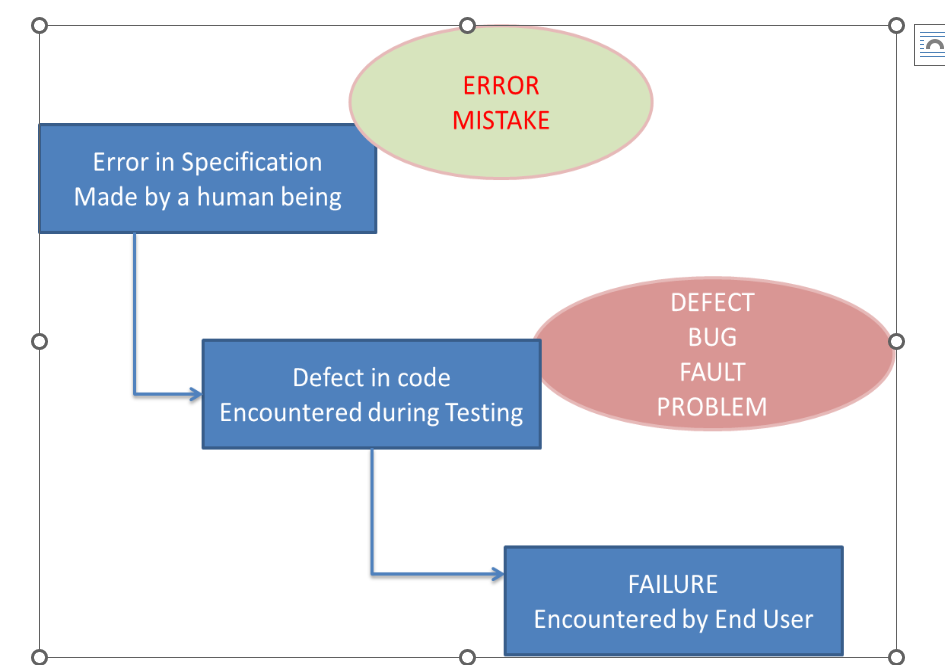
4. Defect Clustering (pareto principle 80-20 rule)

5. The Pesticide Paradox

6. Testing is context dependent

7. Absence of errors fallacy

## Error-Defect-Failure



So to avoid FAILURE

* + We need to avoid errors and faults or identify and correct them.

And Software Testing helps contribute to both avoidance and rectifications. So we need Software testing!

## Qualities of a good tester

**Attention to Detail:** Good testers have a keen eye for detail and can identify even minor discrepancies or inconsistencies in the software they are testing.

**Analytical Skills:** Testers should be able to analyze requirements, specifications, and user stories to create comprehensive test cases and test plans.

**Critical Thinking:**They should think critically to identify potential problem areas and risks in the software, helping to improve its quality.

**Communication Skills:** Effective communication is essential to report and document issues accurately, work collaboratively with developers, and provide clear feedback to the team.

**Technical Proficiency:** Testers should have a good understanding of the technology and tools used in the project, which may include programming languages, testing frameworks, and automation tools.

**Domain Knowledge:** Domain knowledge is valuable for understanding the context of the software being tested, which helps in designing relevant test cases.

**Curiosity:** Good testers are naturally curious and motivated to explore the software thoroughly, not just based on specifications but by thinking from the user’s perspective.

**Adaptability:**Testers need to adapt to different development methodologies (e.g., Agile, Waterfall), tools, and project requirements.

**Patience:** Testing can be repetitive and time-consuming. Testers should exhibit patience and persistence to ensure all scenarios are thoroughly tested.

**Problem-Solving Skills:** When issues are identified, testers should work with the development team to help troubleshoot and find solutions.

## Test Process

We define a test process to ensure that we do not miss the critical steps and do things in the right order.

Test Plan, Test Cases, Testing, Defects, Report, Closure

The fundamental test process consists of the following main activities:

PDERC

* + planning and control;
  + analysis and design;
  + implementation and execution;
  + evaluating exit criteria and reporting;
  + test closure activities.

Test closure activities:

1. Check all defects are fixed and closed
2. Create a summary report
3. Analyze lessons learnt
4. Use the information gathered to improve test process

# Levels of independent testing

In Independent testing, there are several levels of testing which are known as Level of Independence.

1. Testing done by the developer means who developed the item/product.
2. Testing is done by another programmer of the same development team.
3. Testing is done by testers integrated with the developer.
4. Testing is done by some independent testing team from another group of the same organization.
5. Testing is done by some independent testers from other organizations.

### Pros & Cons of Implementing Independent Testing

Independence in the implementation of testing has some key benefits and drawbacks, as described in the following table.

|  |  |  |
| --- | --- | --- |
| **Sr.** | **Benefits** | **Drawbacks** |
| 1 | The tester sees other and different defects to the author | Isolation from the development team (if treated as totally independent), which will mean the tester is totally dependent on the test basis to understand what it is the tester is testing (documentation that is rarely up to date) |
| 2 | The tester is unbiased | The tester may be seen as the bottleneck, as independent test execution is normally the last stage and affected by any delays earlier in the process |
| 3 | The tester can see what has been built rather than what the developer thought had been built | Developers lose a sense of responsibility for quality as it may be assumed that they need not worry about errors because the independent test team will find them |
| 4 | The tester makes no assumptions regarding quality | The fully independent view sets developers and testers on either side of an invisible fence. This can be a hindrance to communication that would in normal circumstances ensure common understanding and effective working. It can also mean that developers are seen to ‘throw’ the software over the fence  Ex: Ego issues, Dev may not give more info on doc change etc |

# Test Levels

* + Unit(component) testing
  + Integration testing
  + System testing
  + Acceptance testing

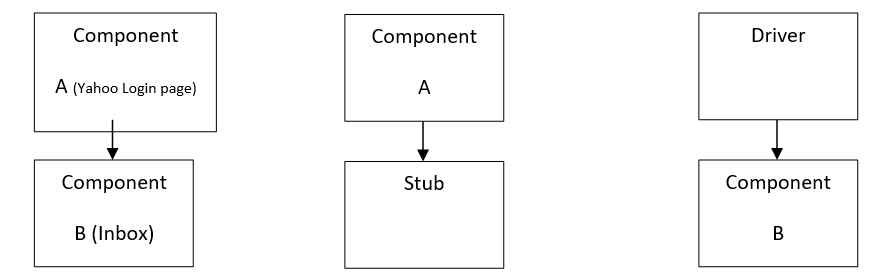
## Unit Testing

Before testing of the code can start, clearly the code has to be written. This is shown at the bottom of the V-model. Generally, the code is written in component parts, or units. The units are usually constructed in isolation, for integration at a later stage.

Units are also called programs, modules or components.

Unit testing is intended to ensure that the code written for the unit meets its specification, prior to its integration with other units.

In addition to checking conformance to the program specification, unit testing would also verify that all of the code that has been written for the unit can be executed.



**Component Testing Typical Defects and Failures:**

* Incorrect Functionality
* Data Flow Problems
* Incorrect Code or Logic

## Integration Testing

Once the units have been written, the next stage would be to put them together to create the system. This is called integration.

It involves building something large from a number of smaller pieces.

During Integration testing**, Control flow and Data flow** are checked.

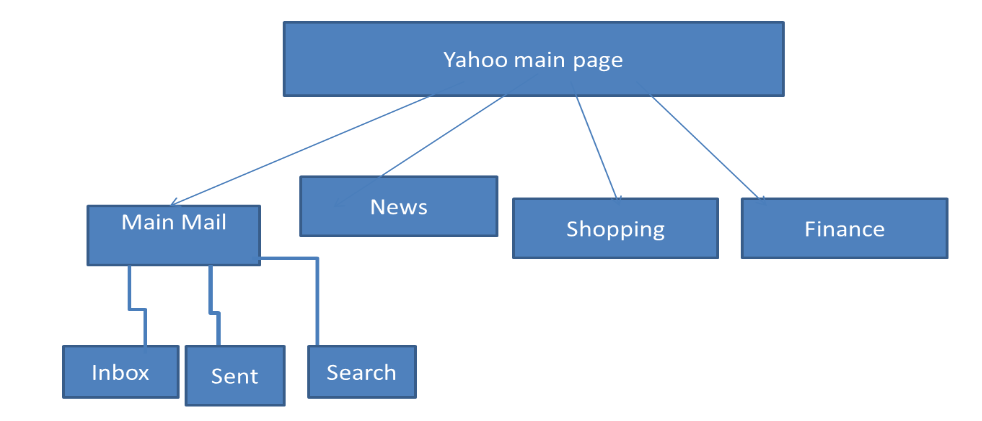
Yahoo Mail page - Main program, Inbox is the child of Yahoo Mail

So to get to Yahoo mail, control needs to go from Yahoo Login to Inbox.

Data also needs to flow - INBOX uses the data (username) to retrieve email for the user.

There are four commonly quoted integration strategies, namely:  
  
1) Big-Bang Integration  
2) Top-Down Integration  
3) Bottom-Up Integration

4) Functional incremental



## System Testing

Having checked that the components all work together at unit level, the next step is to consider the functionality from an end-to-end perspective. This activity is called System testing. (The Functional Specifications are also called System Requirements, hence the name System testing)

* Unit Testing : testing of the individual programs/modules
* Integration : interfaces between modules
* The test cases for unit and integration testing are not representative of the operating conditions in the live environment.
* So far testing has not been conducted end to end.

System testing serves to correct this imbalance by focusing on the behaviour of the whole system/product as defined by the scope of a development project or program, in a representative live environment.

## User Acceptance Test

The purpose of acceptance testing is to provide the end users with confidence that the system will function according to their expectations.

* Unlike system testing, however, the testing conducted here should be independent of any other testing carried out.
* For instance, acceptance testing may assess the system's readiness for deployment and use.
* Acceptance testing is often the responsibility of the customers or users of a system, although other project team members may be involved as well.

# Testing Types

* Functional Testing
* Non Functional Testing
* Whitebox Testing
* Change related testing
* Maintenance Testing

## Functional Testing

This involves testing the application functionality for Ex, placing an order, account creation, etc. Functional testing is often referred to as Black box Testing. Here the tester will focus on the functionality of the software based on the requirement specifications.

## Non Functional Testing

This is where the behavioral aspects of the system are tested. These non-functional requirements are usually documented in a functional specification like functional testing. Thus, mainly black-box testing techniques are used for this type of testing.

Examples of Non-functional testing are performance testing, load testing, stress testing, usability testing, maintainability testing, reliability testing, and portability testing. It is the testing of “how” the system works.

## White box testing

In Whitebox testing, we change our measure to focus on the structural aspects of the system. This could be the code itself.

It is also called as Structural Testing or Glassbox Testing

Unit testing is an example of Whitebox testing

## Testing Related to Changes: Confirmation and Regression Testing

At any level of testing, it can be expected that defects will be discovered.

When these are found and fixed, the quality of the system being delivered can be improved.

After a defect is detected and fixed, the software should be **retested** to **confirm** that the original defect has been successfully removed. This is called **confirmation or retesting**.

When the developer removes the defect, this activity is called **debugging**, which is not a testing activity. Testing finds a defect, debugging fixes it.

The unchanged software should also be retested to ensure that no additional defects have been introduced as a result of changes to the software. This is called **regression testing**.

Regression testing should also be carried out if the environment has changed.

* It is performed when the software, or its environment, is changed.
* The extent of regression testing is based on the risk of not finding defects in software that was working previously.

## Maintenance testing

For many projects the system is eventually released into the live environment. Once deployed, it will be in service as long as intended.

During this deployment, it may become necessary to change the system. Changes may be due to the following reasons:

1) Additional features being required.

2) The system is migrated to a new operating platform or its test environment is corrected, changed or extended or some configuration data is changed

3) The system being retired - data may need to be migrated or archived.

4) Planned upgrade.

5) New faults being found requiring fixing (these can be ‘hot fixes’).

Once changes have been made to the system, they will need to be tested (retesting), and it also will be necessary to conduct regression testing to ensure that the rest of the system has not been adversely affected by the changes. Testing that takes place on a system which is in operation in the live environment is called maintenance testing.

An understanding of the parts of the system which could be affected by the changes can reduce the amount of regression testing. Working this out is called Impact Analysis, , i.e. analyzing the impact of the changes on the system.

Maintenance testing is done on an existing operational system, and is triggered by modifications, migration, or retirement of the software or system.

**Modifications** include planned enhancement changes (e.g., release-based), corrective and emergency changes, and changes of environment, such as planned operating system or database upgrades, planned upgrade, or patches to correct newly exposed or discovered of the operating system.

Maintenance testing for **migration** (e.g., from one platform to another) should include operational tests of the new environment as well as of the changed software.

Maintenance testing for the **retirement** of a system may include the testing of data migration or archiving if long data-retention periods are required.

**Patch** - Publicly released update to fix a known bug/issue

**Hotfix** - update to fix a very specific issue

**Maintenance Release** - Incremental update between service packs or software versions to fix multiple outstanding issues

**Service Pack** - Large Update that fixes many outstanding issues