**Manual Testing:**

* Manual Testing is a testing process in which test cases are **executed manually** without using any automation tool.
* Test cases are generated and planned and implemented manually and it is done according to the perspective of end users.
* Manual Testing is **mandatory** for every newly deployed software **before Automation Testing**. It requires the great efforts and time but it gives surety of 100% bug-free software.
* Manual Testing requires good knowledge of manual testing techniques but not of any automated testing tool.
* Manual Testing is essential because one of the basic fundamentals **of Software testing is 100% Automation is not Possible.**

**Why Manual Testing:**

* To give **stable, bug-free and good quality** product to the client.
* If the test engineer does manual testing, he/she can test the application as an **end-user perspective** and get more familiar with the product, which helps them to write the correct test cases of the application and give the quick feedback of the application.

**Types of Manual Testing:**

* White Box Testing
* Black Box Testing
* Gray Box Testing

**White Box Testing:**

* The white box testing **done by developer**. Developer will test each and every line of code and give it to the tester.
* Since the **code is visible to the developer** so it is called White box testing.

**Block Box Testing:**

* It is done by tester. The tester will check the functionalities of the software according to the client’s needs.
* Since the code is not visible to the tester so it is called black box testing.

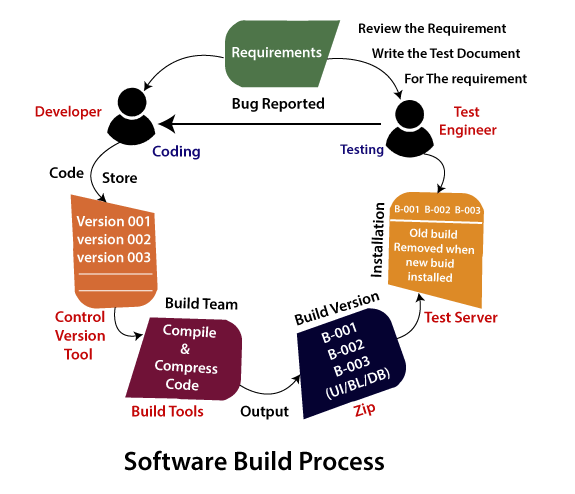
**Gray Box Testing:**

* The person who has good knowledge of **testing as well as coding** will do the code testing and functionality Testing.
* Since it is a combination of white box and black box testing so it is called Gray Box Testing

**How to Perform Manual Testing:**

* First Tester **observe** given **software related documents** to select testing areas.
* Tester analyses the requirement document to **cover all requirements** stated by **the customer**
* Tester develops **testing cases** according to requirement document
* All test cases are **executed manually** by using white box testing and block box testing.
* If bugs occurred the test team will inform to the development team
* The development team will rectify the problem and return to the testing team for retest.

**Software Build Process:**



1. Whenever the requirement is collected, it will provide to the two different team called Developer and Testing Team.
2. Then developer team will start writing the code.
3. Meantime the tester will go through the requirements and prepares the necessary documents. Up to now the developer may complete the code and store into the **control vison tool**.
4. After that, the code changes in the **UI**, and these changes handle by one separate team, which is known as the **build team**.
5. Then they start compile and compress the code with help of build tool. The output which we get will stored as zip file is called as **Build (App or Software)**
6. Then Build will be installed into the **test server**. After that the testing engineer will start do testing by accessing the test server through **Test URL**
7. If the Testing Engineer found any bug means they will inform to the concern developer.
8. Then, the developer will modify the bug by accessing the test URL. Once they fix the bug, they will again upload the new file and they will delete the old one.
9. The process will be going until the **build is getting stabled**.
10. Once the Build is stable, they deliver the Build or App to the Customer.

**Advantages of Manual Testing**

* It does **not require programming knowledge** while using the Black box method.
* It is used to test dynamically changing GUI designs.
* Tester interacts with software as a real user so that they are able to discover usability and user interface issues.
* It ensures that the software is a hundred percent bug-free.
* It is cost-effective.
* Easy to learn for new testers.

**Disadvantages of Manual Testing**

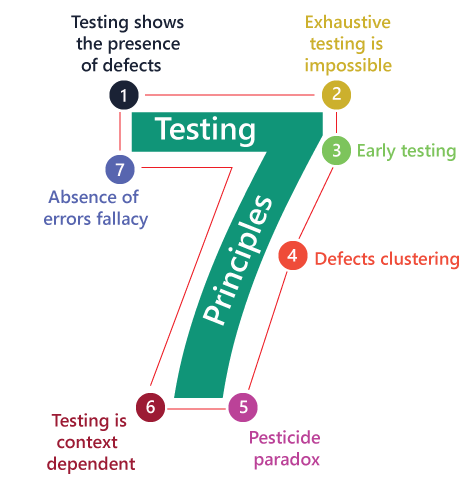
* It requires a large number of human resources.
* It is very time-consuming.
* Tester develops test cases based on their skills and experience. There is no evidence that they have covered all functions or not.
* Test cases cannot be used again. Need to develop separate test cases for each new software.

It does not provide testing on all aspects of testing.

* Since two teams work together, sometimes it is difficult to understand each other's motives, it can mislead the process.

**Software Testing Principles:**

* Testing shows the presence of defects
* Exhaustive Testing is not possible
* Early Testing
* Defect Clustering
* Pesticide Paradox
* Testing is context-dependent
* Absence of errors fallacy



**Testing Shows the presence of defects:**

Testing Engineer test the application for delivering the application bug or defect free. So, primary purpose of testing is to **address the bugs**.

And, by doing testing we can decrease the errors but it doesn’t mean that the whole application is free because we can address the error even during the deployment.

**Exhaustive Testing is not possible:**

Sometimes it seems to be very hard to test all the modules and their features with effective and non- effective combinations of the inputs data throughout the actual testing process.

Most of the time it takes more time and eventually our hard work could become unsuccessful. So better complete the testing, by testing modules according to its importance.

**Early Testing:**

The Early testing means finding the bugs in earlier stage. To achieve this, we **need requirement document** at earlier stage. So that we can find the error at earlier stage it would cause less amount of cost as compared to finding errors at final stage.

**Defect Clustering:**

Defect Clustering is the process of addressing the uncertain modules which has 80% of the bug in a whole project. Which could be because of its complication.

**Pesticide Paradox:**

* This principle defined that if we are executing the same set of test cases again and again over a particular time, then these kinds of the test will not be able to find the new bugs in the software or the application.
* To get over these pesticide paradoxes, it is very significant to review all the test cases frequently.
* And the new and different tests are necessary to be written for the implementation of multiple parts of the application or the software, which helps us to find more bugs.

**Testing is Context-Dependent:**

The testing is context – dependent states that we have a various field in the market and each has its own functionalities and features so, for testing these, kind of applications or software we have to get aid from various technology, method and different approaches to test the application. Therefore, testing depends on the context of the application

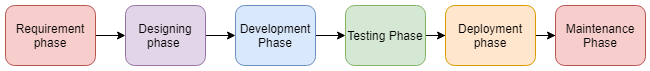
**Absence of Error Fallacy:**

The absence of error fallacy means **identifying and fixing** the bugs **would not help** if the application is **impractical (not practical)** and not able to accomplish the client's requirements and needs.

**Software Development Life Cycle (SDLC):**

SDLC is a process that creates a **structure of development** of software. There are different phases within SDLC, and each phase has its various activities. It makes the development team able to **design, create, and delive**r a high-quality product.

It simply gives the order of execution of the phases.



**Requirement Phase:**

* This is a crucial phase
* The phase in which we are collecting the requirements from the client.
* We have to get the information about the product like how will be it used and who will use to determine the load operations.

**Designing Phase:**

* This is high priority case.
* The detailed analysis of new software according to requirement phase will be happen here.
* Here logical designing converted into physical design.
* The decision like which programming language, data bases, and the combination of software and hardware to provide the platform to run the software that we have to use those, kind of decision is taken in this phase only.
* There are several techniques and tools, such as data flow diagrams, flowcharts, decision tables, and decision trees, Data dictionary, and the structured dictionary are used for describing the system design

**Development/Build Phase:**

* Once the design completed then it will be given to the development team where the work will divide into small units.
* Frontend -developer will do all the GUI work.
* Back end -developer will do all the backend operations according to the requirements.
* Since this is the coding phase, it takes the longest time and more focused approach for the developer in the software development life cycle.

**Testing Phase:**

* Testing determines whether the software is actually giving the result as per the requirements addressed in the requirement phase or not.
* Develop Team makes the test plan. This plan includes all type of essential testing such as integration testing, system testing, unit testing, acceptance testing. Non -functional testing also done in this phase.

**Deployment/Deliver Phase:**

* Whenever the testing phase is completed and it gives satisfying result, then it becomes ready to deliver to the customer.
* Once we delivered to the customer, we recommend them to do beta testing. In beta testing customer can require any changes which are not present in the software but mentioned in the document or changes in GUI to make it user friendly.
* Besides if any problem occurs it will be informed to the development team and they will fix it instantly if it is severe or otherwise they will simply said wait for the next version.

**Maintenance:**

* The maintenance phase is the last and long-lasting phase of SDLC because it is the process which **continues** until the **software's life cycle** comes to an end.
* When a customer starts using software, then actual problems start to occur, and at that time there's a need to solve these problems.
* This phase also includes making **changes in hardware and software** to **maintain** its operational **effectiveness** like to improve its **performance**, enhance security features and according to customer's requirements with upcoming time.
* This process to take care of product time to time is called maintenance.

**Software Testing Life Cycle (STLC):**

The procedure of software testing is also known as STLC (Software Testing Life Cycle) which includes phases of the testing process. The testing process is executed in a well-planned and systematic manner. All activities are done to improve the quality of the software product.

**Steps**:

1. [Requirement Analysis](https://www.javatpoint.com/software-testing-life-cycle#requirement-analysis) – detail analysis of requirement document
2. [Test Plan Creation](https://www.javatpoint.com/software-testing-life-cycle#test-plan-creation) – developing test cases to execute
3. [Environment setup](https://www.javatpoint.com/software-testing-life-cycle#environment-setup) – set the testing environment. It is done by senior developers
4. [Test case Execution](https://www.javatpoint.com/software-testing-life-cycle#test-case-execution) – Then executes the test cases for each and every essential functionality.
5. [Defect Logging](https://www.javatpoint.com/software-testing-life-cycle#defect-logging) - Defect logging analysis mainly works to find out **defect distribution depending upon severity and types**. If, any defect is detected, then the software is returned to the development team to fix the defect, then the software is re-tested on all aspects of the testing.
6. [Test Cycle Closure](https://www.javatpoint.com/software-testing-life-cycle#test-cycle-closure) -

* The test cycle closure report includes all the documentation related to software design, development, testing results, and defect reports.
* This phase evaluates the strategy of development, testing procedure, possible defects in order to use these practices in the future if there is a software with the same specification

**SDLC MODELS:**

* Waterfall model
* Spiral model
* Verification and validation model
* Prototype model
* Hybrid model

**Water Model:**

It is a simple model that is easy to use as well as understand. The execution happens in the **sequence order**, which means that the outcome of the one-stage is equal to the input of another stage. That's why it is also known as the **Linear-sequential life cycle model**.

The waterfall model is divided into various stages, which are as follows:

1. Requirement collection
2. Feasibility study
3. Design
4. Coding
5. Testing
6. Installation
7. Maintenance

**Spiral Models:**

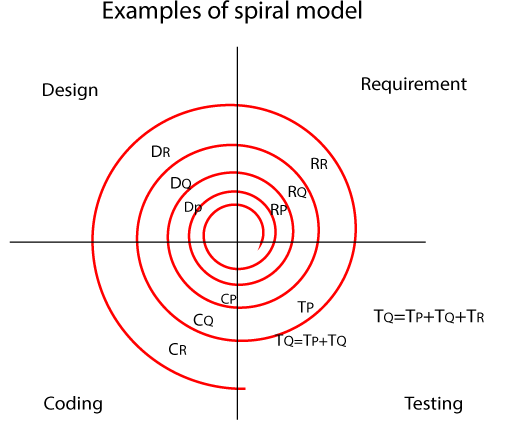
The biggest problem we face in the **waterfall model is that taking a long duration** to complete the product, and the software became outdated. To solve this problem, we have a new approach, which is known as the Spiral model. The spiral model is also known as the cyclic model.

we create the application **module by module** and handed over to the **customer** so that they can start using the application at a very **early stage**

In this model, we develop the application in the stages because sometimes the client gives the requirements in between the process.

The different phases of the spiral model are as follows:

1. Requirement analysis
2. Design
3. Coding
4. Testing and risk analysis

****

**Consider that the project that need 3 modules named P, Q, R to develop a software**

**Module P, Module Q, Module R**

**Requirement analysis = RP, RQ, RR**

**Design = DP, DQ, DR**

**Coding = CP, CQ, CR**

**Testing = TP, TQ, TR**

**Cycle 1:**

**Module 1 = RP 🡪 DP🡪TP**

**Cycle 2:**

**Once the everything is done for Module 1, P, then it will go Module 2, Q**

**Module 1 & Module 2 = RQ🡪 DQ🡪 T = TP+TQ**

**Cycle 3:**

**After the two Cycle, completed, then 3rd cycle executes,**

**Module1&Module2& Module3 = RR🡪DR🡪T = TP+TQ+TR**

**Protype Model:**

The most significant disadvantage of previous models (waterfall and spiral) is that there were lots of **customer rejection** that happens after the application was **developed**, and there was no involvement of the customers in between the project.

The prototype is just the sample or a dummy of the required software product. We should get the **product reviewed by client** then only we started develop the product.

**Protype Model Process:**

1. Requirement analysis
2. feasibility study
3. Create a prototype
4. Prototype testing
5. Customer review and approval
6. Design
7. Coding
8. Testing
9. Installation and maintenance

**V-Model (V&V-Model/Validation &Verification Model):**

* V-Model came up to overcome the drawback of the water model.
* In this model activities goes on downward direction and one point in a time it started going upward direction for testing process thus make the shape of V so, it is called V -Model.

**When We go for this model:**

🡪 Larger Application – Which means app has **n- number** of modules

🡪 Complex Application – Which means app has many dependencies modules

**Requirements:**

It is the document which is collected from the client. For, V and V Model we have two files,

🡪 CRS/BRS – Customer/Business Requirement Specification (English language)

🡪 SRS/FS – Software/Functional Requirement Specification (Coding Language)

**Characteristics of Requirement:**

🡪 Requirement should be in detail (modules, components, functional specification)

🡪 It has to be in sequential order.

🡪 Has to be written in simple language

🡪 It should be measurable and countable

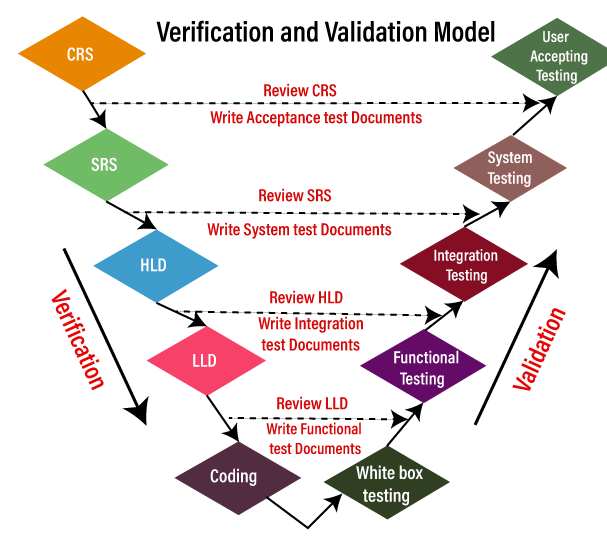
**V and V Model Process:**

**Step 1:**

It will start from collecting the CRS (customer requirement specification) document, from the client by the Business Analyst where the test engineer will check the following scenarios:

* **Review the CRS** based on
  + Incorrect requirements
  + Missing requirements
  + Conflicts in the requirements
* **Write Acceptance Test documents**

Once the test engineer team reviews the CRS and found any bugs or defects, they will send it to the development team for fixing the bugs. After fixing the bugs, the development team updates the CRS and concurrently developing the SRS document.



**Stage 2:**

After completing the CRS, the SRS is sent to the testing team for the review process, and the developers start creating the HLD (high-level design) for the application. And the testing team will test the SRS on the following scenarios:

* **Review the SRS against CRS**
  + Each CRS is transferred to SRS
  + CRS is not transformed properly to SRS
* **Write the system Test documents**

Once the testing team reviews every detail of the SRS and CRS has been converted correctly to SRS, we will move to our next stage.

**Stage 3:**

After the completion of HLD, the developers start creating the LLD (Low-level design) for the application, and in the meantime, the tester will check the following tests on the HLD:

* **Review HLD**
* **Write integration test documents**

**Step 4:**

Once the testing team has done reviewing the HLD, the developers write the coding and develops the application, and the testing team will do the following tasks:

* **Review the LLD**
* **Write functional test documents**

**Stage 5**

After the completion of the coding part, the developers will perform one round of unit testing, which is also called white box testing, and check every line of the code and make sure that the code is correct.

After performing the unit testing, the application is sent to the testing team, where they perform multiple testing such as **functional testing, integration testing, and system testing, and acceptance testing.**

And once the testing part is done, the application will finally deliver to the customer.

**Hybrid Model**

The hybrid model is the combination of two or more primary (traditional) models and modifies them as per the business requirements. This model is dependent on the other SDLC models, such as spiral, V and V, and prototype models. The hybrid model is mainly used for small, medium, and large projects. It focuses on the risk management of the product

The most commonly used combination of two models is as follows:

1. **Spiral and prototype**

**V & V and Prototype**