Software Development Life Cycle

What are the Software Development Life Cycle (SDLC) phases?

There are various software development approaches defined and designed which are used/employed during development process of software, these approaches are also referred as "Software Development Process Models" (e.g. Waterfall model, incremental model, V-model, iterative model, etc.). Each process model follows a particular life cycle in order to ensure success in process of software development.

Software life cycle models describe phases of the software cycle and the order in which those phases are executed. Each phase produces deliverables required by the next phase in the life cycle. Requirements are translated into design. Code is produced according to the design which is called development phase. After coding and development the testing verifies the deliverable of the implementation phase against requirements.

There are following six phases in every Software development life cycle model:

1. Requirement gathering and analysis
2. Design
3. Implementation or coding
4. Testing
5. Deployment
6. Maintenance

1) Requirement gathering and analysis: Business requirements are gathered in this phase. This phase is the main focus of the project managers and stake holders. Meetings with managers, stake holders and users are held in order to determine the requirements like; Who is going to use the system? How will they use the system? What data should be input into the system? What data should be output by the system? These are general questions that get answered during a requirements gathering phase. After requirement gathering these requirements are analyzed for their validity and the possibility of incorporating the requirements in the system to be development is also studied.

Finally, a Requirement Specification document is created which serves the purpose of guideline for the next phase of the model.

2) Design: In this phase the system and software design is prepared from the requirement specifications which were studied in the first phase. System Design helps in specifying hardware and system requirements and also helps in defining overall system architecture. The system design specifications serve as input for the next phase of the model.

3) Implementation / Coding: On receiving system design documents, the work is divided in modules/units and actual coding is started. Since, in this phase the code is produced so it is the main focus for the developer. This is the longest phase of the software development life cycle.

4) Testing: After the code is developed it is tested against the requirements to make sure that the product is actually solving the needs addressed and gathered during the requirements phase. During this phase unit testing, integration testing, system testing, acceptance testing are done.

5) Deployment: After successful testing the product is delivered / deployed to the customer for their use.

6) Maintenance: Once when the customers starts using the developed system then the actual problems comes up and needs to be solved from time to time. This process where the care is taken for the developed product is known as maintenance.

What are the Software Development Models?

The development models are the various processes or methodologies that are being selected for the development of the project depending on the project's aims and goals. There are many development life cycle models that have been developed in order to achieve different required objectives. The models specify the various stages of the process and the order in which they are carried out.

The selection of model has very high impact on the testing that is carried out. It will define the what, where and when of our planned testing, influence regression testing and largely determines which test techniques to use.

There are various Software development models or methodologies. They are as follows:

1. Waterfall model
2. V model
3. Incremental model
4. RAD model
5. Agile model
6. Iterative model
7. Spiral model

Choosing right model for developing of the software product or application is very important. Based on the model the development and testing processes are carried out.

Different companies based on the software application or product, they select the type of development model whichever suits to their application. But these days in market the 'Agile Methodology' is the most used model. 'Waterfall Model' is the very old model. In 'Waterfall Model' testing starts only after the development is completed. Because of which there are many defects and failures which are reported at the end. So,the cost of fixing these issues are high. Hence, these days people are preferring 'Agile Model'. In 'Agile Model' after every sprint there is a demo-able feature to the customer. Hence customer can see the features whether they are satisfying their need or not.

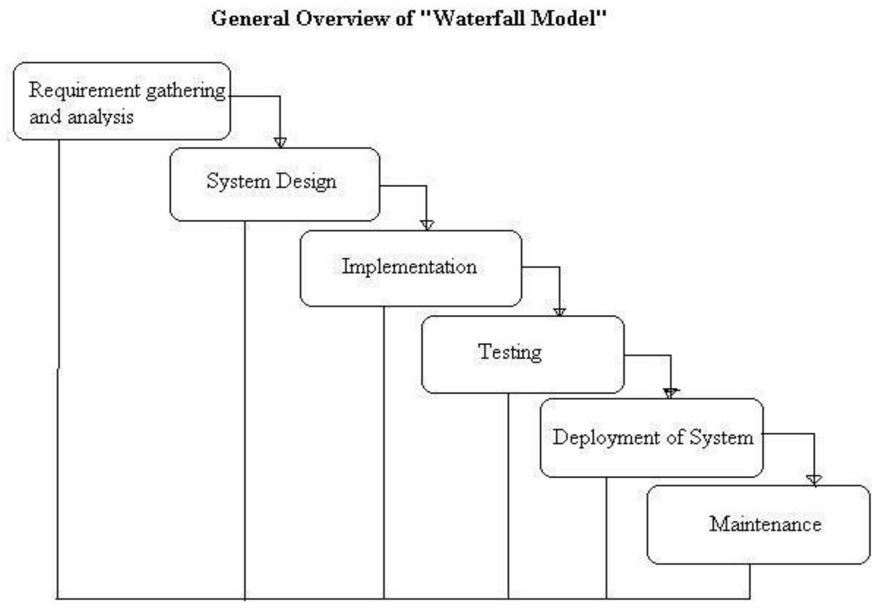
'V-model' is also used by many of the companies in their product. 'V-model' is nothing but 'Verification' and 'Validation' model. In 'V-model' the developer's life cycle and tester's life cycle are mapped to each other. In this model testing is done side by side of the development.

Likewise 'Incremental model', 'RAD model', 'Iterative model' and 'Spiral model' are also used based on the requirement of the customer and need of the product.

What is Waterfall model- advantages, disadvantages and when to use it?

The Waterfall Model was first Process Model to be introduced. It is also referred to as a **linear-sequential life cycle model**. It is very simple to understand and use. In a waterfall model, each phase must be completed fully before the next phase can begin. This type of model is basically used for the for the project which is small and there are no uncertain requirements. At the end of each phase, a review takes place to determine if the project is on the right path and whether or not to continue or discard the project. In this model the testing starts only after the development is complete. In **waterfall model phases** do not overlap.

Diagram of Waterfall-model:


General Overview of "Waterfall Model"
Requirement gathering and analysis
System Design
Implementation
Testing
Deployment of System
Maintenance

Advantages of waterfall model:

- This model is simple and easy to understand and use.
- It is easy to manage due to the rigidity of the model – each phase has specific deliverables and a review process.
- In this model phases are processed and completed one at a time. Phases do not overlap.
- Waterfall model works well for smaller projects where requirements are very well understood.

Disadvantages of waterfall model:

- Once an application is in the testing stage, it is very difficult to go back and change something that was not well-thought out in the concept stage.
- No working software is produced until late during the life cycle.
- High amounts of risk and uncertainty.
- Not a good model for complex and object-oriented projects.
- Poor model for long and ongoing projects.
- Not suitable for the projects where requirements are at a moderate to high risk of changing.

When to use the waterfall model:

- This model is used only when the requirements are very well known, clear and fixed.
- Product definition is stable.
- Technology is understood.
- There are no ambiguous requirements
- Ample resources with required expertise are available freely

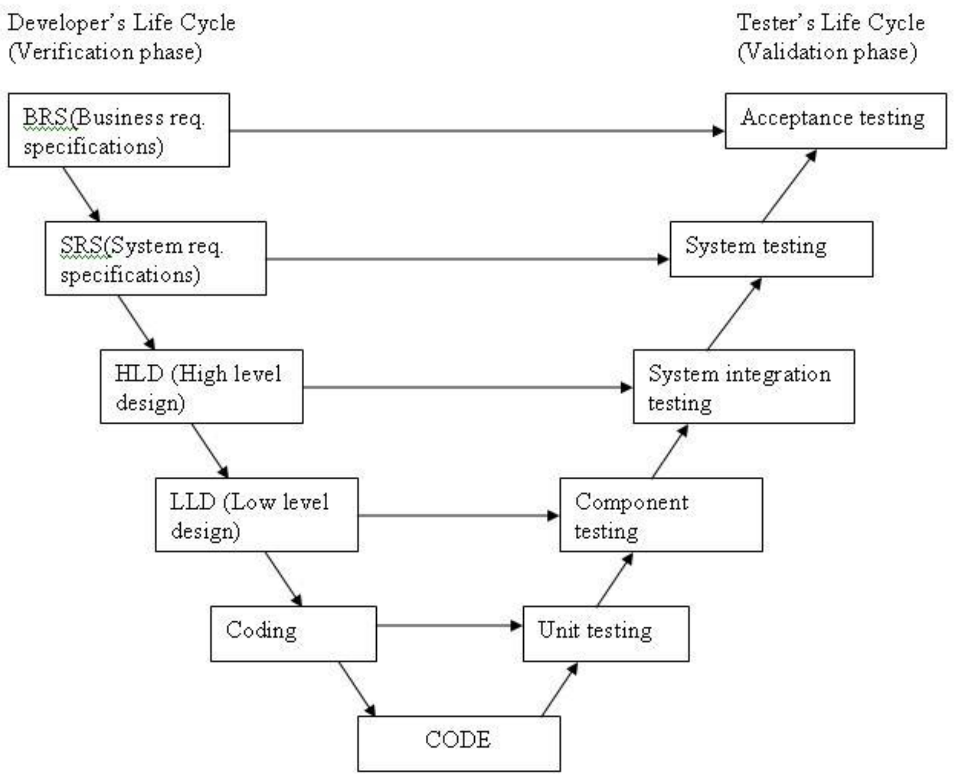
- The project is short.

Very less customer enter action is involved during the development of the product. Once the product is ready then only it can be demoed to the end users. Once the product is developed and if any failure occurs then the cost of fixing such issues are very high, because we need to update everywhere from document till the logic.

What is V-model- advantages, disadvantages and when to use it?

V- model means Verification and Validation model. Just like the waterfall model, the V-Shaped life cycle is a sequential path of execution of processes. Each phase must be completed before the next phase begins. Testing of the product is planned in parallel with a corresponding phase of development.

Diagram of V-model:



The various phases of the V-model are as follows:

Requirements like BRS and SRS begin the life cycle model just like the waterfall model. But, in this model before development is started, a system test plan is created. The test plan focuses on meeting the functionality specified in the requirements gathering.

The high-level design (HLD) phase focuses on system architecture and design. It provide overview of solution, platform, system, product and service/process. An integration test plan is created in this phase as well in order to test the pieces of the software systems ability to work together.

The low-level design (LLD) phase is where the actual software components are designed. It defines the actual logic for each and every component of the system. Class diagram with all the methods and relation between classes comes under LLD. Component tests are created in this phase as well.

The implementation phase is, again, where all coding takes place. Once coding is complete, the path of execution continues up the right side of the V where the test plans developed earlier are now put to use.

Coding: This is at the bottom of the V-Shape model. Module design is converted into code by developers.

Advantages of V-model:

- Simple and easy to use.
- Testing activities like planning, test designing happens well before coding. This saves a lot of time. Hence higher chance of success over the waterfall model.
- Proactive defect tracking – that is defects are found at early stage.
- Avoids the downward flow of the defects.
- Works well for small projects where requirements are easily understood.

Disadvantages of V-model:

- Very rigid and least flexible.
- Software is developed during the implementation phase, so no early prototypes of the software are produced.
- If any changes happen in midway, then the test documents along with requirement documents has to be updated.

When to use the V-model:

- The V-shaped model should be used for small to medium sized projects where requirements are clearly defined and fixed.
- The V-Shaped model should be chosen when ample technical resources are available with needed technical expertise.

High confidence of customer is required for choosing the V-Shaped model approach. Since, no prototypes are produced, there is a very high risk involved in meeting customer expectations.

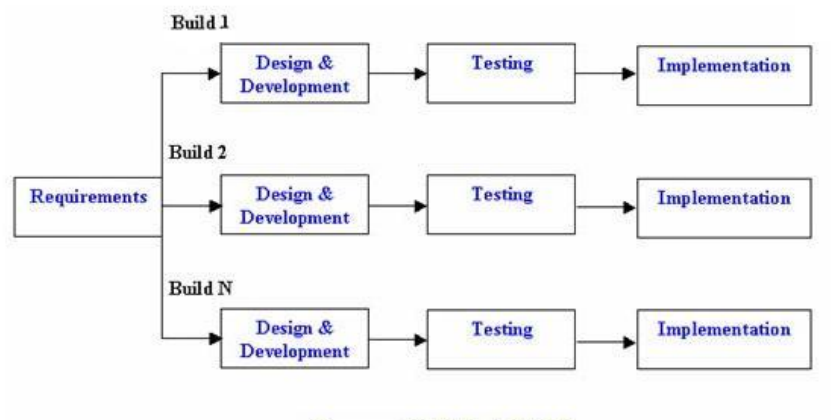
What is Incremental model- advantages, disadvantages and when to use it?

In incremental model the whole requirement is divided into various builds. Multiple development cycles take place here, making the life cycle a "multi-waterfall" cycle. Cycles are divided up into smaller, more easily managed modules. Each module passes through the requirements, design, implementation and testing phases. A working version of software is produced during the first module, so you have working software early on during the software life cycle. Each subsequent release of the module adds function to the previous release. The process continues till the complete system is achieved.

For example:

In the diagram above when we work **incrementally** we are adding piece by piece but expect that each piece is fully finished. Thus keep on adding the pieces until it's complete. As in the image above a person has thought of the application. Then he started building it and in the first iteration the first module of the application or product is totally ready and can be demoed to the customers. Likewise in the second iteration the other module is ready and integrated with the first module. Similarly, in the third iteration the whole product is ready and integrated. Hence, the product got ready step by step.

Diagram of Incremental model:



Incremental Life Cycle Model

Advantages of Incremental model:

- Generates working software quickly and early during the software life cycle.
- This model is more flexible – less costly to change scope and requirements.
- It is easier to test and debug during a smaller iteration.
- In this model customer can respond to each built.
- Lowers initial delivery cost.
- Easier to manage risk because risky pieces are identified and handled during it'd iteration.

Disadvantages of Incremental model:

- Needs good planning and design.
- Needs a clear and complete definition of the whole system before it can be broken down and built incrementally.
- Total cost is higher than waterfall.

When to use the Incremental model:

- This model can be used when the requirements of the complete system are clearly defined and understood.
- Major requirements must be defined; however, some details can evolve with time.
- There is a need to get a product to the market early.
- A new technology is being used
- Resources with needed skill set are not available
- There are some high risk features and goals.

What is RAD model- advantages, disadvantages and when to use it?

RAD model is Rapid Application Development model. It is a type of incremental model. In RAD model the components or functions are developed in parallel as if they were mini projects. The developments are time boxed, delivered and then assembled into a working prototype. This can quickly give the customer something to see and use and to provide feedback regarding the delivery and their requirements.

Diagram of RAD-Model:

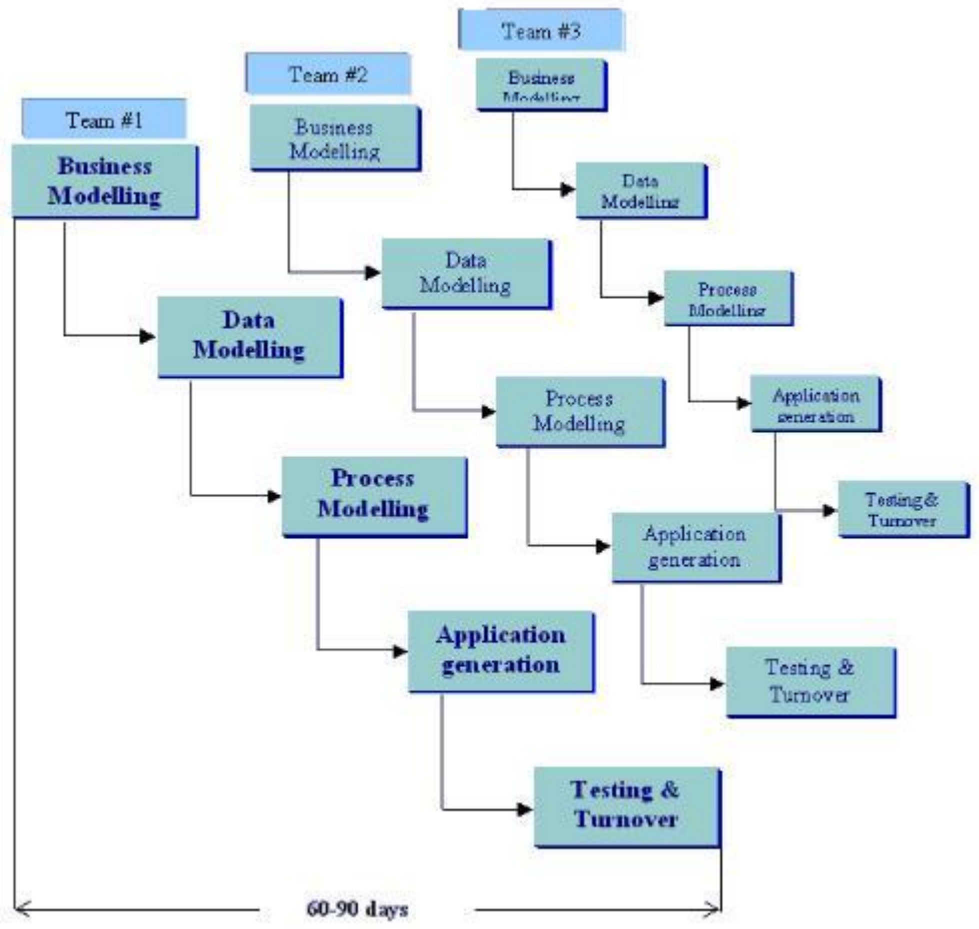


Figure 1.5 – RAD Model

The phases in the rapid application development (RAD) model are:

Business modeling: The information flow is identified between various business functions.
Data modeling: Information gathered from business modeling is used to define data objects that are needed for the business.
Process modeling: Data objects defined in data modeling are converted to achieve the business information flow to achieve some specific business objective. Description are identified and created for CRUD of data objects.
Application generation: Automated tools are used to convert process models into code and the actual system.
Testing and turnover: Test new components and all the interfaces.

Advantages of the RAD model:

- Reduced development time.
- Increases reusability of components
- Quick initial reviews occur
- Encourages customer feedback
- Integration from very beginning solves a lot of integration issues.

Disadvantages of RAD model:

- Depends on strong team and individual performances for identifying business requirements.
- Only system that can be modularized can be built using RAD
- Requires highly skilled developers/designers.
- High dependency on modeling skills
- Inapplicable to cheaper projects as cost of modeling and automated code generation is very high.

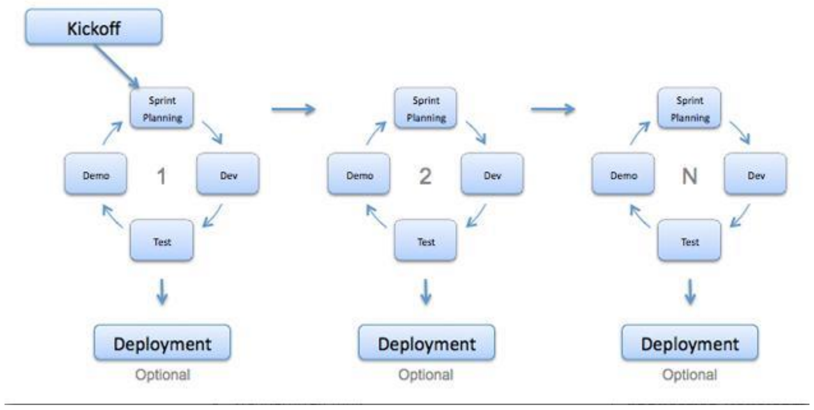
When to use RAD model:

- RAD should be used when there is a need to create a system that can be modularized in 2-3 months of time.
- It should be used if there's high availability of designers for modeling and the budget is high enough to afford their cost along with the cost of automated code generating tools.
- RAD SDLC model should be chosen only if resources with high business knowledge are available and there is a need to produce the system in a short span of time (2-3 months).

What is Agile model – advantages, disadvantages and when to use it?

Agile development model is also a type of Incremental model. Software is developed in incremental, rapid cycles. This results in small incremental releases with each release building on previous functionality. Each release is thoroughly tested to ensure software quality is maintained. It is used for time critical applications. Extreme Programming (XP) is currently one of the most well known agile development life cycle model.

Diagram of Agile model:



Advantages of Agile model:

- Customer satisfaction by rapid, continuous delivery of useful software.

- People and interactions are emphasized rather than process and tools. Customers, developers and testers constantly interact with each other.
- Working software is delivered frequently (weeks rather than months).
- Face-to-face conversation is the best form of communication.
- Close, daily cooperation between business people and developers.
- Continuous attention to technical excellence and good design.
- Regular adaptation to changing circumstances.
- Even late changes in requirements are welcomed

Disadvantages of Agile model:

- In case of some software deliverables, especially the large ones, it is difficult to assess the effort required at the beginning of the software development life cycle.
- There is lack of emphasis on necessary designing and documentation.
- The project can easily get taken off track if the customer representative is not clear what final outcome that they want.
- Only senior programmers are capable of taking the kind of decisions required during the development process. Hence it has no place for newbie programmers, unless combined with experienced resources.

When to use Agile model:

- When new changes are needed to be implemented. The freedom agile gives to change is very important. New changes can be implemented at very little cost because of the frequency of new increments that are produced.
- To implement a new feature the developers need to lose only the work of a few days, or even only hours, to roll back and implement it.
- Unlike the waterfall model in agile model very limited planning is required to get started with the project. Agile assumes that the end users' needs are ever changing in a dynamic business and IT world. Changes can be discussed and features can be newly effected or removed based on feedback. This effectively gives the customer the finished system they want or need.
- Both system developers and stakeholders alike, find they also get more freedom of time and options than if the software was developed in a more rigid sequential way. Having options gives them the ability to leave important decisions until more or better data or even entire hosting programs are available; meaning the project can continue to move forward without fear of reaching a sudden standstill.

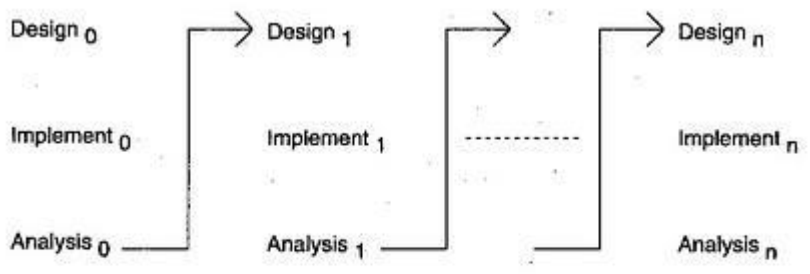
What is Iterative model- advantages, disadvantages and when to use it?

An iterative life cycle model does not attempt to start with a full specification of requirements. Instead, development begins by specifying and implementing just part of the software, which can then be reviewed in order to identify further requirements. This process is then repeated, producing a new version of the software for each cycle of the model.

For example:

In the diagram above when we work **iteratively** we create rough product or product piece in one iteration, then review it and improve it in next iteration and so on until it's finished. As shown in the image above, in the first iteration the whole painting is sketched roughly, then in the second iteration colors are filled and in the third iteration finishing is done. Hence, in iterative model the whole product is developed step by step.

Diagram of Iterative model:



Advantages of Iterative model:

- In iterative model we can only create a high-level design of the application before we actually begin to build the product and define the design solution for the entire product. Later on we can design and built a skeleton version of that, and then evolved the design based on what had been built.
- In iterative model we are building and improving the product step by step. Hence we can track the defects at early stages. This avoids the downward flow of the defects.
- In iterative model we can get the reliable user feedback. When presenting sketches and blueprints of the product to users for their feedback, we are effectively asking them to imagine how the product will work.
- In iterative model less time is spent on documenting and more time is given for designing.

Disadvantages of Iterative model:

- Each phase of an iteration is rigid with no overlaps
- Costly system architecture or design issues may arise because not all requirements are gathered up front for the entire lifecycle

When to use iterative model:

- Requirements of the complete system are clearly defined and understood.
- When the project is big.
- Major requirements must be defined; however, some details can evolve with time.

What is Spiral model- advantages, disadvantages and when to use it?

The spiral model is similar to the incremental model, with more emphasis placed on risk analysis. The spiral model has four phases: Planning, Risk Analysis, Engineering and Evaluation. A software project repeatedly passes through these phases in iterations (called Spirals in this model). The baseline spiral, starting in the planning phase, requirements are gathered and risk is assessed. Each subsequent spirals builds on the baseline spiral.

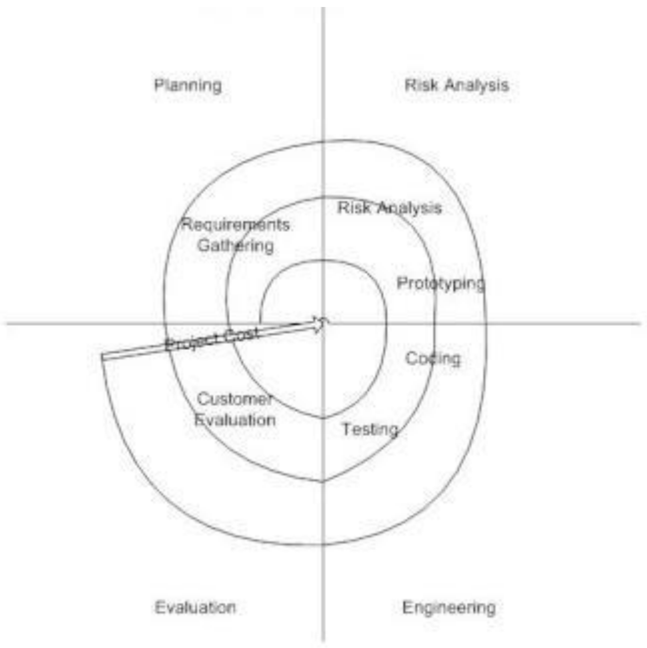
Planning Phase: Requirements are gathered during the planning phase. Requirements like 'BRS' that is 'Bussiness Requirement Specifications' and 'SRS' that is 'System Requirement specifications'.

Risk Analysis: In the **risk analysis phase**, a process is undertaken to identify risk and alternate solutions. A prototype is produced at the end of the risk analysis phase. If any risk is found during the risk analysis then alternate solutions are suggested and implemented.

Engineering Phase: In this phase software is **developed**, along with testing at the end of the phase. Hence in this phase the development and testing is done.

Evaluation phase: This phase allows the customer to evaluate the output of the project to date before the project continues to the next spiral.

Diagram of Spiral model:



Advantages of Spiral model:

- High amount of risk analysis hence, avoidance of Risk is enhanced.
- Good for large and mission-critical projects.
- Strong approval and documentation control.
- Additional Functionality can be added at a later date.

- Software is produced early in the software life cycle.

Disadvantages of Spiral model:

- Can be a costly model to use.
- Risk analysis requires highly specific expertise.
- Project's success is highly dependent on the risk analysis phase.
- Doesn't work well for smaller projects.

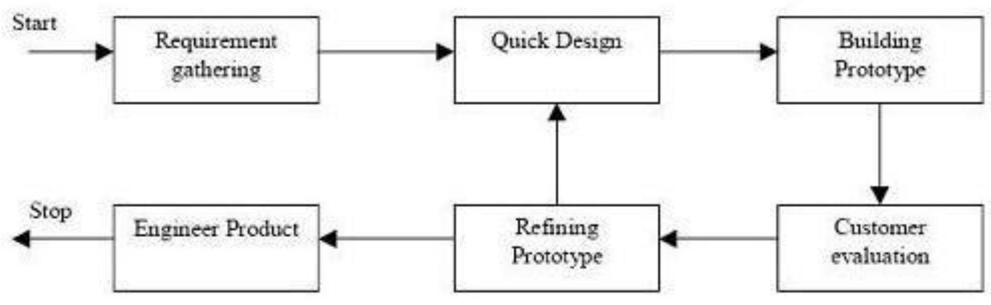
When to use Spiral model:

- When costs and risk evaluation is important
- For medium to high-risk projects
- Long-term project commitment unwise because of potential changes to economic priorities
- Users are unsure of their needs
- Requirements are complex
- New product line
- Significant changes are expected (research and exploration)

What is Prototype model- advantages, disadvantages and when to use it?

The basic idea here is that instead of freezing the requirements before a design or coding can proceed, a throwaway prototype is built to understand the requirements. This prototype is developed based on the currently known requirements. By using this prototype, the client can get an "actual feel" of the system, since the interactions with prototype can enable the client to better understand the requirements of the desired system. Prototyping is an attractive idea for complicated and large systems for which there is no manual process or existing system to help determining the requirements. The prototype are usually not complete systems and many of the details are not built in the prototype. The goal is to provide a system with overall functionality.

Diagram of Prototype model:



Prototyping Model

Advantages of Prototype model:

- Users are actively involved in the development

- Since in this methodology a working model of the system is provided, the users get a better understanding of the system being developed.
- Errors can be detected much earlier.
- Quicker user feedback is available leading to better solutions.
- Missing functionality can be identified easily
- Confusing or difficult functions can be identified
- Requirements validation, Quick implementation of, incomplete, but functional, application.

Disadvantages of Prototype model:

- Leads to implementing and then repairing way of building systems.
- Practically, this methodology may increase the complexity of the system as scope of the system may expand beyond original plans.
- Incomplete application may cause application not to be used as the full system was designed
- Incomplete or inadequate problem analysis.

When to use Prototype model:

- Prototype model should be used when the desired system needs to have a lot of interaction with the end users.
- Typically, online systems, web interfaces have a very high amount of interaction with end users, are best suited for Prototype model. It might take a while for a system to be built that allows ease of use and needs minimal training for the end user.
- Prototyping ensures that the end users constantly work with the system and provide a feedback which is incorporated in the prototype to result in a useable system. They are excellent for designing good human computer interface systems.

What are Software Testing Levels?

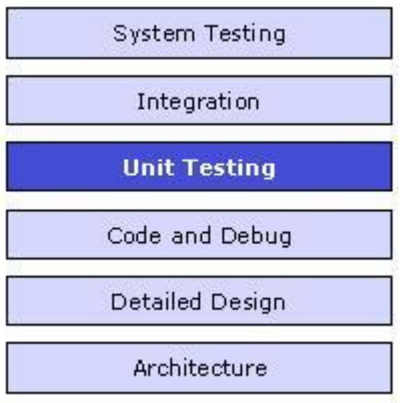
Testing levels are basically to identify missing areas and prevent overlap and repetition between the development life cycle phases. In software development life cycle models there are defined phases like requirement gathering and analysis, design, coding or implementation, testing and deployment. Each phase goes through the testing. Hence there are various levels of testing. The various levels of testing are:

1. Unit testing: It is basically done by the developers to make sure that their code is working fine and meet the user specifications. They test their piece of code which they have written like classes, functions, interfaces and procedures.
2. Component testing: It is also called as module testing. The basic difference between the unit testing and component testing is in unit testing the developers test their piece of code but in component testing the whole component is tested. For example, in a student record application there are two modules one which will save the records of the students and other module is to upload the results of the students. Both the modules are developed separately and when they are tested one by one then we call this as a component or module testing.
3. Integration testing: Integration testing is done when two modules are integrated, in order to test the behavior and functionality of both the modules after integration. Below are few types of integration testing:
    - Big bang integration testing
    - Top down
    - Bottom up
    - Functional incremental

4. Component integration testing: In the example above when both the modules or components are integrated then the testing done is called as Component integration testing. This testing is basically done to ensure that the code should not break after integrating the two modules.
5. System integration testing: System integration testing (SIT) is a testing where testers basically test that in the same environment all the related systems should maintain data integrity and can operate in coordination with other systems.
6. System testing: In system testing the testers basically test the compatibility of the application with the system.
7. Acceptance testing: Acceptance testing are basically done to ensure that the requirements of the specification are met.
8. Alpha testing: Alpha testing is done at the developers site. It is done at the end of the development process
9. Beta testing: Beta testing is done at the customers site. It is done just before the launch of the product.

What is Unit testing?

- A unit is the smallest testable part of an application like functions, classes, procedures, interfaces. Unit testing is a method by which individual units of source code are tested to determine if they are fit for use.
- **Unit tests are basically written and executed by software developers** to make sure that code meets its design and requirements and behaves as expected.
- The goal of unit testing is to segregate each part of the program and test that the individual parts are working correctly.
- This means that for any function or procedure when a set of inputs are given then it should return the proper values. It should handle the failures gracefully during the course of execution when any invalid input is given.
- A unit test provides a written contract that the piece of code must assure. Hence it has several benefits.
- Unit testing is basically done before integration as shown in the image below.



Method Used for unit testing: White Box Testing method is used for executing the unit test.

When Unit testing should be done?

Unit testing should be done before Integration testing.

By whom unit testing should be done?

Unit testing should be done by the developers.

Advantages of Unit testing:

1. Issues are found at early stage. Since unit testing are carried out by developers where they test their individual code before the integration. Hence the issues can be found very early and can be resolved then and there without impacting the other piece of codes.

2. Unit testing helps in maintaining and changing the code. This is possible by making the codes less interdependent so that unit testing can be executed. Hence chances of impact of changes to any other code gets reduced.

3. Since the bugs are found early in unit testing hence it also helps in reducing the cost of bug fixes. Just imagine the cost of bug found during the later stages of development like during system testing or during acceptance testing.

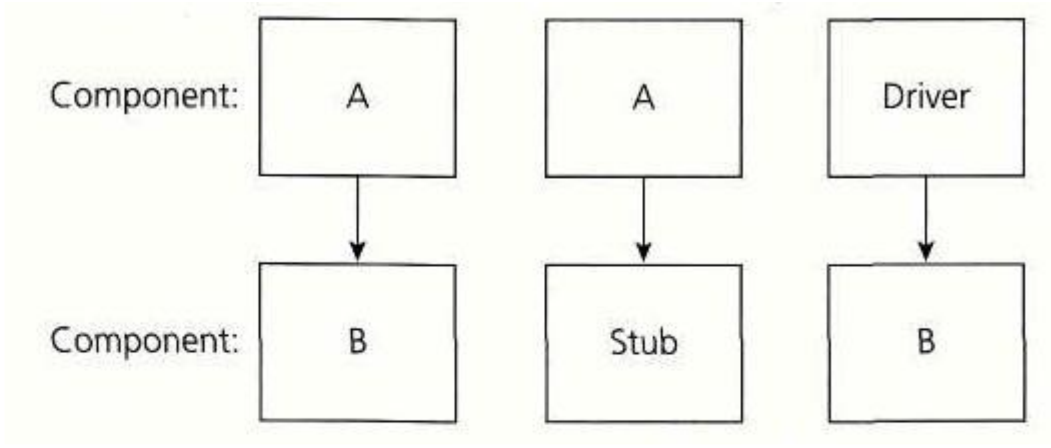
4. Unit testing helps in simplifying the debugging process. If suppose a test fails then only latest changes made in code needs to be debugged.

What is Component testing?

What is Component testing?: Component testing is a method where testing of each component in an application is done separately. Suppose, in an application there are 5 components. Testing of each 5 components separately and efficiently is called as component testing.

- Component testing is also known as module and program testing. It finds the defects in the module and verifies the functioning of software.
- Component testing is done by the tester.
- Component testing may be done in isolation from rest of the system depending on the development life cycle model chosen for that particular application. In such case the missing software is replaced by **Stubs** and **Drivers** and simulate the interface between the software components in a simple manner.
- Let's take an example to understand it in a better way. Suppose there is an application consisting of three modules say, module A, module B and module C. The developer has developed the module B and now wanted to test it. But in order to test the module B completely few of it's functionalities are dependent on module A and few on module C. But the module A and module C has not been developed yet. In that case to test the module B completely we can replace the module A and module C by stub and drivers as required.
- **Stub:** A stub is called from the software component to be tested. As shown in the diagram below 'Stub' is called by 'component A'.
- **Driver:** A driver calls the component to be tested. As shown in the diagram below 'component B' is called by the 'Driver'.

Below is the diagram of the component testing:



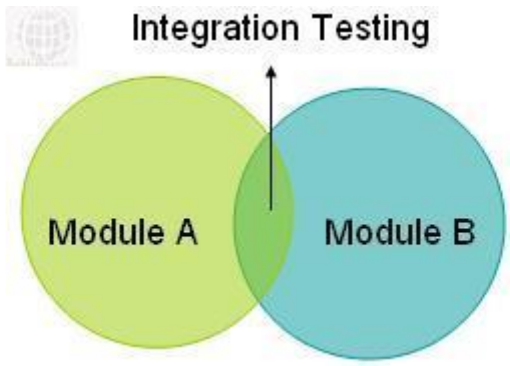
As discussed in the previous article of the 'Unit testing' it is done by the developers where they do the testing of the individual functionality or procedure. After unit testing is executed, component testing comes into the picture. Component testing is done by the testers.

Component testing plays a very important role in finding the bugs. Before we start with the integration testing it's always preferable to do the component testing in order to ensure that each component of an application is working effectively.

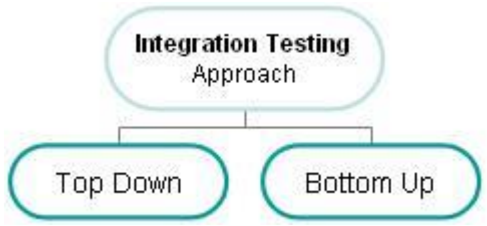
Integration testing is followed by the component testing.

What is Integration testing?

- Integration testing tests integration or interfaces between components, interactions to different parts of the system such as an operating system, file system and hardware or interfaces between systems.
- Also after integrating two different components together we do the integration testing. As displayed in the image below when two different modules 'Module A' and 'Module B' are integrated then the integration testing is done.



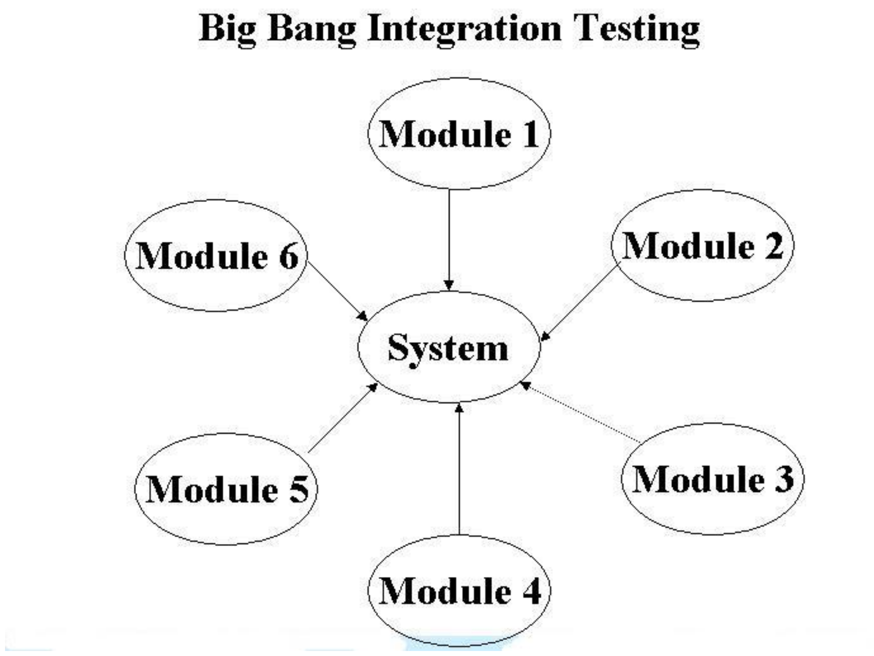
- Integration testing is done by a specific integration tester or test team.
- Integration testing follows two approach known as 'Top Down' approach and 'Bottom Up' approach as shown in the image below:



Below are the integration testing techniques:

1. Big Bang integration testing:

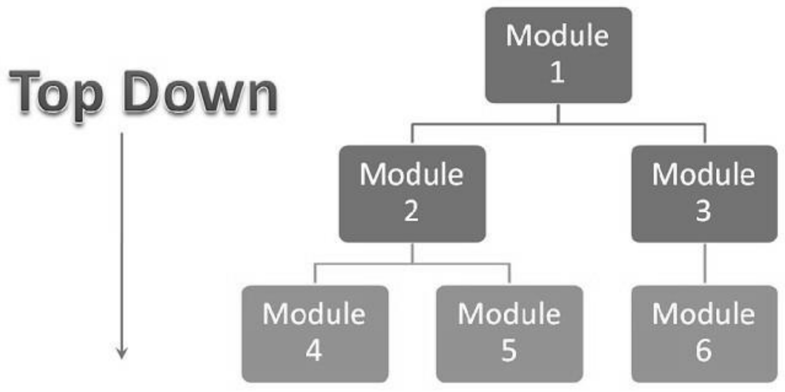
In Big Bang integration testing all components or modules are integrated simultaneously, after which everything is tested as a whole. As per the below image all the modules from 'Module 1′ to 'Module 6′ are integrated simultaneously then the testing is carried out.



Advantage: Big Bang testing has the advantage that everything is finished before integration testing starts.

Disadvantage: The major disadvantage is that in general it is time consuming and difficult to trace the cause of failures because of this late integration.

2. Top-down integration testing: Testing takes place from top to bottom, following the control flow or architectural structure (e.g. starting from the GUI or main menu). Components or systems are substituted by stubs. Below is the diagram of 'Top down Approach':



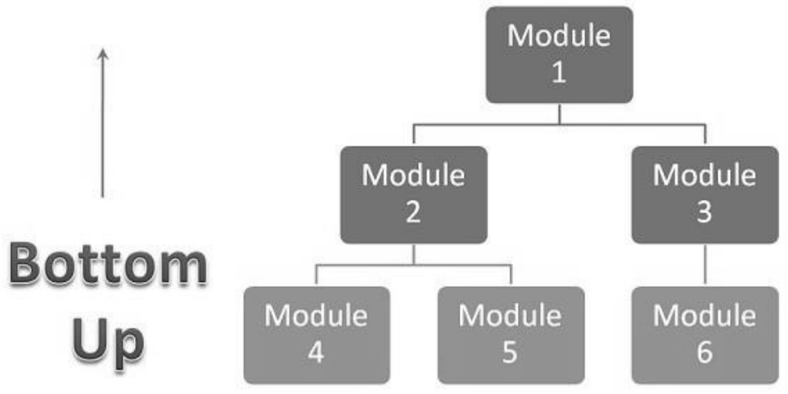
Advantages of Top-Down approach:

- The tested product is very consistent because the integration testing is basically performed in an environment that almost similar to that of reality
- Stubs can be written with lesser time because when compared to the drivers then Stubs are simpler to author.

Disadvantages of Top-Down approach:

- Basic functionality is tested at the end of cycle

3. Bottom-up integration testing: Testing takes place from the bottom of the control flow upwards. Components or systems are substituted by drivers. Below is the image of 'Bottom up approach':



Advantage of Bottom-Up approach:

- In this approach development and testing can be done together so that the product or application will be efficient and as per the customer specifications.

Disadvantages of Bottom-Up approach:

- We can catch the Key interface defects at the end of cycle
- It is required to create the test drivers for modules at all levels except the top control

Incremental testing:

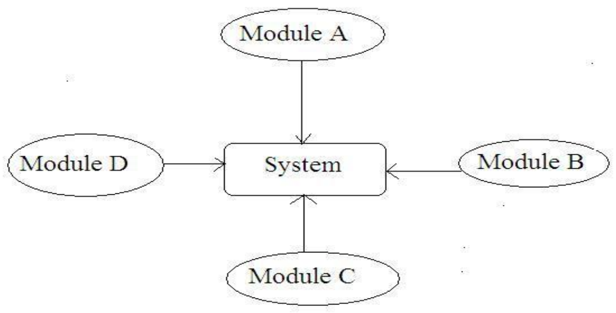
Another extreme is that all programmers are integrated one by one, and a test is carried out after each step. The incremental approach has the advantage that the defects are found early in a smaller assembly when it is relatively easy to detect the cause.
A disadvantage is that it can be time-consuming since stubs and drivers have to be developed and used in the test.
Within incremental integration testing a range of possibilities exist, partly depending on the system architecture.

Functional incremental: Integration and testing takes place on the basis of the functions and functionalities, as documented in the functional specification.

What is Big Bang integration testing?

- In Big Bang integration testing all components or modules are integrated simultaneously, after which everything is tested as a whole.
- In this approach individual modules are not integrated until and unless all the modules are ready.
- In Big Bang integration testing all the modules are integrated without performing any integration testing and then it's executed to know whether all the integrated modules are working fine or not.
- This approach is generally executed by those developers who follows the 'Run it and see' approach.
- Because of integrating everything at one time if any failures occurs then it become very difficult for the programmers to know the root cause of that failure.
- In case any bug arises then the developers has to detach the integrated modules in order to find the actual cause of the bug.

Below is the image of the big bang integration testing:



Suppose a system consists of four modules as displayed in the diagram above. In big bang integration all the four modules 'Module A, Module B, Module C and Module D' are integrated simultaneously and then the testing is performed. Hence in this approach no individual integration testing is performed because of which the chances of critical failures increases.

Advantage of Big Bang Integration:

- Big Bang testing has the **advantage** that everything is finished before integration testing starts.

Disadvantages of Big Bang Integration:

- The major **disadvantage** is that in general it is very time consuming
- It is very difficult to trace the cause of failures because of this late integration.
- The chances of having critical failures are more because of integrating all the components together at same time.
- If any bug is found then it is very difficult to detach all the modules in order to find out the root cause of it.
- There is high probability of occurrence of the critical bugs in the production environment

What is Incremental testing in software?

- Another extreme is that all programmers are integrated one by one, and a test is carried out after each step.
- The incremental approach has the advantage that the defects are found early in a smaller assembly when it is relatively easy to detect the cause.
- A disadvantage is that it can be time-consuming since stubs and drivers have to be developed and used in the test.
- Within incremental integration testing a range of possibilities exist, partly depending on the system architecture:
  - **Top down:** Testing takes place from top to bottom, following the control flow or architectural structure (e.g. starting from the GUI or main menu). Components or systems are substituted by stubs.
  - **Bottom up:** Testing takes place from the bottom of the control flow upwards. Components or systems are substituted by drivers.
  - **Functional incremental:** Integration and testing takes place on the basis of the functions nad functionalities, as documented in the functional specification.

What is Component integration testing?

- It tests the interactions between software components and is done after component testing.
- The software components themselves may be specified at different times by different specification groups, yet the integration of all of the pieces must work together.
- It is important to cover negative cases as well because components might make assumption with respect to the data.

What is System integration testing?

- It tests the interactions between different systems and may be done after system testing.
- It verifies the proper execution of software components and proper interfacing between components within the solution.
- The objective of SIT Testing is to validate that all software module dependencies are functionally correct and that data integrity is maintained between separate modules for the entire solution.

- As testing for dependencies between different components is a primary function of SIT Testing, this area is often most subject to Regression Testing.

What is 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.
- System testing is carried out by specialists testers or independent testers.
- System testing should investigate both functional and non-functional requirements of the testing.

What is Acceptance testing?

- After the system test has corrected all or most defects, the system will be delivered to the user or customer for acceptance testing.
- Acceptance testing is basically done by the user or customer although other stakeholders may be involved as well.
- The goal of acceptance testing is to establish confidence in the system.
- Acceptance testing is most often focused on a validation type testing.
- Acceptance testing may occur at more than just a single level, for example:

    - A **Commercial Off the shelf (COTS)** software product may be acceptance tested when it is installed or integrated.
    - **Acceptance testing of the usability of the component** may be done during component testing.
    - **Acceptance testing of a new functional enhancement** may come before system testing.

- The **types of acceptance testing** are:

    - The **User Acceptance test:** focuses mainly on the functionality thereby validating the fitness-for-use of the system by the business user. The user acceptance test is performed by the users and application managers.
    - The **Operational Acceptance test:** also known as Production acceptance test validates whether the system meets the requirements for operation. In most of the organization the operational acceptance test is performed by the system administration before the system is released. The operational acceptance test may include testing of backup/restore, disaster recovery, maintenance tasks and periodic check of security vulnerabilities.
    - **Contract Acceptance testing**: It is performed against the contract's acceptance criteria for producing custom developed software. Acceptance should be formally defined when the contract is agreed.
    - **Compliance acceptance testing:** It is also known as regulation acceptance testing is performed against the regulations which must be adhered to, such as governmental, legal or safety regulations.

What is Alpha testing?

Alpha testing is one of the most common software testing strategy used in software development. Its specially used by product development organizations.

- This **test takes place at the developer's site**. Developers observe the users and note problems.
- Alpha testing is testing of an application when development is about to complete. Minor design changes can still be made as a result of alpha testing.
- Alpha testing is typically performed by a group that is independent of the design team, but still within the company, e.g. in-house software test engineers, or software QA engineers.
- Alpha testing is final testing before the software is released to the general public. It has two phases:
    - In the **first phase** of alpha testing, the software is tested by in-house developers. They use either debugger software, or hardware-assisted debuggers. The goal is to catch bugs quickly.
    - In the **second phase** of alpha testing, the software is handed over to the software QA staff, for additional testing in an environment that is similar to the intended use.

- Alpha testing is simulated or actual operational testing by potential users/customers or an independent test team at the developers' site. Alpha testing is often employed for off-the-shelf software as a form of internal acceptance testing, before the software goes to beta testing.

What is Beta testing?

- It is also known as field testing. It takes place at **customer's site**. It sends the system to users who install it and use it under real-world working conditions.
- A beta test is the second phase of software testing in which a sampling of the intended audience tries the product out. (Beta is the second letter of the Greek alphabet.) Originally, the term *alpha test* meant the first phase of testing in a software development process. The first phase includes unit testing, component testing, and system testing. Beta testing can be considered "pre-release testing.

- The goal of beta testing is to place your application in the hands of real users outside of your own engineering team to discover any flaws or issues from the user's perspective that you would not want to have in your final, released version of the application.

Open and closed beta:
Developers release either a **closed beta** or an **open beta**;

- *Closed beta versions are released to a select group of individuals for a user test and are invitation only, while*
- *Open betas are from a larger group to the general public and anyone interested. The testers report any bugs that they find, and sometimes suggest additional features they think should be available in the final version.*

Advantages of beta testing:

- You have the opportunity to get your application into the hands of users prior to releasing it to the general public.
- Users can install, test your application, and send feedback to you during this beta testing period.
- Your beta testers can discover issues with your application that you may have not noticed, such as confusing application flow, and even crashes.

- Using the feedback you get from these users, you can fix problems before it is released to the general public.
- The more issues you fix that solve real user problems, the higher the quality of your application when you release it to the general public.
- Having a higher-quality application when you release to the general public will increase customer satisfaction.
- These users, who are early adopters of your application, will generate excitement about your application.

What are Software Test Types?

Test types are introduced as a means of clearly defining the objective of a certain level for a program or project. A test type is focused on a particular test objective, which could be the testing of the function to be performed by the component or system; a non-functional quality characteristics, such as reliability or usability; the structure or architecture of the component or system; or related to changes, i.e confirming that defects have been fixed (confirmation testing or retesting) and looking for unintended changes (regression testing). Depending on its objectives, testing will be organized differently. Hence there are four software test types:

1. Functional testing
2. Non-functional testing
3. Structural testing

What is Functional testing (Testing of functions) in software?

In functional testing basically the testing of the functions of component or system is done. It refers to activities that verify a specific action or function of the code. Functional test tends to answer the questions like “can the user do this” or “does this particular feature work”. This is typically described in a requirements specification or in a functional specification.

The techniques used for functional testing are often specification-based. Testing functionality can be done from two perspective:

- **Requirement-based testing:** In this type of testing the requirements are prioritized depending on the risk criteria and accordingly the tests are prioritized. This will ensure that the most important and most critical tests are included in the testing effort.
- **Business-process-based testing:** In this type of testing the scenarios involved in the day-to-day business use of the system are described. It uses the knowledge of the business processes. For example, a personal and payroll system may have the business process along the lines of: someone joins the company, employee is paid on the regular basis and employee finally leaves the company.

What is Non-functional testing (Testing of software product characteristics)?

In non-functional testing the quality characteristics of the component or system is tested. Non-functional refers to aspects of the software that may not be related to a specific function or user action such as scalability or security. Eg. How many people can log in at once? Non-functional testing is also performed at all levels like functional testing.

Non-functional testing includes:

- Functionality testing
- Reliability testing
- Usability testing
- Efficiency testing
- Maintainability testing
- Portability testing
- Baseline testing
- Compliance testing
- Documentation testing
- Endurance testing
- Load testing
- Performance testing
- Compatibility testing
- Security testing
- Scalability testing
- Volume testing
- Stress testing
- Recovery testing
- Internationalization testing and Localization testing

- ▪ **Functionality testing:** Functionality testing is performed to verify that a software application performs and functions correctly according to design specifications. During functionality testing we check the core application functions, text input, menu functions and installation and setup on localized machines, etc.
- ▪ **Reliability testing:** Reliability Testing is about exercising an application so that failures are discovered and removed before the system is deployed. The purpose of reliability testing is to determine product reliability, and to determine whether the software meets the customer's reliability requirements.
- ▪ **Usability testing:** In usability testing basically the testers tests the ease with which the user interfaces can be used. It tests that whether the application or the product built is user-friendly or not.

Usability testing includes the following five components:

1. **Learnability:** How easy is it for users to accomplish basic tasks the first time they encounter the design?
2. **Efficiency:** How fast can experienced users accomplish tasks?
3. **Memorability:** When users return to the design after a period of not using it, does the user remember enough to use it effectively the next time, or does the user have to start over again learning everything?
4. **Errors:** How many errors do users make, how severe are these errors and how easily can they recover from the errors?
5. **Satisfaction:** How much does the user like using the system?

2. **Efficiency testing:** Efficiency testing test the amount of code and testing resources required by a program to perform a particular function. Software Test Efficiency is number of test cases executed divided by unit of time (generally per hour).
3. **Maintainability testing:** It basically defines that how easy it is to maintain the system. This means that how easy it is to analyze, change and test the application or product.

4. **Portability testing:** It refers to the process of testing the ease with which a computer software component or application can be moved from one environment to another, e.g. moving of any application from Windows 2000 to Windows XP. This is usually measured in terms of the maximum amount of effort permitted. Results are measured in terms of the time required to move the software and complete the and documentation updates.
5. **Baseline testing:** It refers to the validation of documents and specifications on which test cases would be designed. The requirement specification validation is baseline testing.
6. **Compliance testing:** It is related with the IT standards followed by the company and it is the testing done to find the deviations from the company prescribed standards.
7. **Documentation testing:** As per the IEEE Documentation describing plans for, or results of, the testing of a system or component, Types include test case specification, test incident report, test log, test plan, test procedure, test report. Hence the testing of all the above mentioned documents is known as documentation testing.
8. **Endurance testing:** Endurance testing involves testing a system with a significant load extended over a significant period of time, to discover how the system behaves under sustained use. For example, in software testing, a system may behave exactly as expected when tested for 1 hour but when the same system is tested for 3 hours, problems such as memory leaks cause the system to fail or behave randomly.
9. **Load testing:** A load test is usually conducted to understand the behavior of the application under a specific expected load. Load testing is performed to determine a system's behavior under both normal and at peak conditions. It helps to identify the maximum operating capacity of an application as well as any bottlenecks and determine which element is causing degradation. E.g. If the number of users are in creased then how much CPU, memory will be consumed, what is the network and bandwidth response time
10. **Performance testing:** Performance testing is testing that is performed, to determine how fast some aspect of a system performs under a particular workload. It can serve different purposes like it can demonstrate that the system meets performance criteria. It can compare two systems to find which performs better. Or it can measure what part of the system or workload causes the system to perform badly.
11. **Compatibility testing:** Compatibility testing is basically the testing of the application or the product built with the computing environment. It tests whether the application or the software product built is compatible with the hardware, operating system, database or other system software or not.
12. **Security testing:** Security testing is basically to check that whether the application or the product is secured or not. Can anyone came tomorrow and hack the system or login the application without any authorization. It is a process to determine that an information system protects data and maintains functionality as intended.
13. **Scalability testing:** It is the testing of a software application for measuring its capability to scale up in terms of any of its non-functional capability like load supported, the number of transactions, the data volume etc.
14. **Volume testing:** Volume testing refers to testing a software application or the product with a certain amount of data. E.g., if we want to volume test our application with a specific database size, we need to expand our database to that size and then test the application's performance on it.
15. **Stress testing:** It involves testing beyond normal operational capacity, often to a breaking point, in order to observe the results. It is a form of testing that is used to determine the stability of a given system. It put greater emphasis on robustness, availability, and error handling under a heavy load, rather than on what would be considered correct behavior under normal circumstances. The goals of such tests may be to ensure the software does not crash in conditions of insufficient computational resources (such as memory or disk space).

16. **Recovery testing:** Recovery testing is done in order to check how fast and better the application can recover after it has gone through any type of crash or hardware failure etc. Recovery testing is the forced failure of the software in a variety of ways to verify that recovery is properly performed. For example, when an application is receiving data from a network, unplug the connecting cable. After some time, plug the cable back in and analyze the application's ability to continue receiving data from the point at which the network connection got disappeared. Restart the system while a browser has a definite number of sessions and check whether the browser is able to recover all of them or not.
17. **Internationalization testing and Localization testing:** Internationalization is a process of designing a software application so that it can be adapted to various languages and regions without any changes. Whereas Localization is a process of adapting internationalized software for a specific region or language by adding local specific components and translating text.

What is functionality testing in software?

Functionality testing is performed to verify that a software application performs and functions correctly according to design specifications. During functionality testing we check the core application functions, text input, menu functions and installation and setup on localized machines, etc.

The following is needed to be checked during the functionality testing:

- Installation and setup on localized machines running localized operating systems and local code pages.
- Text input, including the use of extended characters or non-Latin scripts.
- Core application functions.
- String handling, text, and data, especially when interfacing with non-Unicode applications or modules.
- Regional settings defaults.
- Text handling (such as copying, pasting, and editing) of extended characters, special fonts, and non-Latin scripts.
- Accurate hot-key shortcuts without any duplication.

Functionality testing verifies that an application is still fully functional after localization. Even applications which are professionally internationalized according to world-readiness guidelines require functionality testing.

What is reliability testing in software?

- Reliability Testing is about exercising an application so that failures are discovered and removed before the system is deployed. The purpose of reliability testing is to determine product reliability, and to determine whether the software meets the customer's reliability requirements.
- According to ANSI, Software Reliability is defined as: the probability of failure-free software operation for a specified period of time in a specified environment. Software Reliability is not a direct function of time. Electronic and mechanical parts may become "old" and wear out with time and usage, but software will not rust or wear-out during its life cycle. Software will not change over time unless intentionally changed or upgraded.
- Reliability refers to the consistency of a measure. A test is considered reliable if we get the same result repeatedly. Software Reliability is the probability of failure-free software operation for a

specified period of time in a specified environment. Software Reliability is also an important factor affecting system reliability.
- Reliability testing will tend to uncover earlier those failures that are most likely in actual operation, thus directing efforts at fixing the most important faults.
- Reliability testing may be performed at several levels. Complex systems may be tested at component, circuit board, unit, assembly, subsystem and system levels.

Software reliability is a key part in software quality. The study of software reliability can be categorized into three parts:

1. Modeling
2. Measurement
3. Improvement

1. **Modeling:** Software reliability modeling has matured to the point that meaningful results can be obtained by applying suitable models to the problem. There are many models exist, but no single model can capture a necessary amount of the software characteristics. Assumptions and abstractions must be made to simplify the problem. There is no single model that is universal to all the situations.

2. **Measurement:** Software reliability measurement is naive. Measurement is far from commonplace in software, as in other engineering field. "How good is the software, quantitatively?" As simple as the question is, there is still no good answer. Software reliability can not be directly measured, so other related factors are measured to estimate software reliability and compare it among products. Development process, faults and failures found are all factors related to software reliability.

3. **Improvement:** Software reliability improvement is hard. The difficulty of the problem stems from insufficient understanding of software reliability and in general, the characteristics of software. Until now there is no good way to conquer the complexity problem of software. Complete testing of a moderately complex software module is infeasible. Defect-free software product can not be assured. Realistic constraints of time and budget severely limits the effort put into software reliability improvement.

What is Usability testing in software and it's benefits to end user?

- In usability testing basically the testers tests the ease with which the user interfaces can be used. It tests that whether the application or the product built is user-friendly or not.
- Usability Testing is a black box testing technique.
- Usability testing also reveals whether users feel comfortable with your application or Web site according to different parameters - the flow, navigation and layout, speed and content - especially in comparison to prior or similar applications.
- Usability Testing tests the following features of the software.

– How easy it is to use the software.
– How easy it is to learn the software.
– How convenient is the software to end user.

Usability testing includes the following five components:

1. **Learnability:** How easy is it for users to accomplish basic tasks the first time they encounter the design?
2. **Efficiency:** How fast can experienced users accomplish tasks?

3. **Memorability:** When users return to the design after a period of not using it, does the user remember enough to use it effectively the next time, or does the user have to start over again learning everything?
4. **Errors:** How many errors do users make, how severe are these errors and how easily can they recover from the errors?
5. **Satisfaction:** How much does the user like using the system?

Benefits of usability testing to the end user or the customer:

– Better quality software
– Software is easier to use
– Software is more readily accepted by users
– Shortens the learning curve for new users

Advantages of usability testing:

- Usability test can be modified to cover many other types of testing such as functional testing, system integration testing, unit testing, smoke testing etc.
- Usability testing can be very economical if planned properly, yet highly effective and beneficial.
- If proper resources (experienced and creative testers) are used, usability test can help in fixing all the problems that user may face even before the system is finally released to the user. This may result in better performance and a standard system.
- Usability testing can help in discovering potential bugs and potholes in the system which generally are not visible to developers and even escape the other type of testing.

Usability testing is a very wide area of testing and it needs fairly high level of understanding of this field along with creative mind. People involved in the usability testing are required to possess skills like patience, ability to listen to the suggestions, openness to welcome any idea, and the most important of them all is that they should have good observation skills to spot and fix the issues or problems.

What is Efficiency testing in software?

Efficiency testing test the amount of code and testing resources required by a program to perform a particular function. Software Test Efficiency is number of test cases executed divided by unit of time (generally per hour).

It is internal in the organization how much resources were consumed how much of these resources were utilized.

Here are some formulas to calculate **Software Test Efficiency** (for different factors):

- Test efficiency = (total number of defects found in unit+integration+system) / (total number of defects found in unit+integration+system+User acceptance testing)
- Testing Efficiency = (No. of defects Resolved / Total No. of Defects Submitted)* 100

Software Test Effectiveness covers three aspects:

– How much the customer's requirements are satisfied by the system.
– How well the customer specifications are achieved by the system.
– How much effort is put in developing the system.

What is Maintainability testing in software?

It basically defines that how easy it is to maintain the system. This means that how easy it is to analyze, change and test the application or product.

Maintainability testing shall use a model of the maintainability requirements of the software/system. The maintainability testing shall be specified in terms of the effort required to effect a change under each of the following four categories:

- **Corrective maintenance -** Correcting problems. The maintainability of a system can be measured in terms of the time taken to diagnose and fix problems identified within that system.
- **Perfective maintenance -** Enhancements. The maintainability of a system can also be measured in terms of the effort taken to make required enhancements to that system. This can be tested by recording the time taken to achieve a new piece of identifiable functionality such as a change to the database, etc. A number of similar tests should be run and an average time calculated. The outcome will be that it is possible to give an average effort required to implement specified functionality. This can be compared against a target effort and an assessment made as to whether requirements are met.
- **Adaptive maintenance -** Adapting to changes in environment. The maintainability of a system can also be measured in terms on the effort required to make required adaptations to that system. This can be measured in the way described above for perfective maintainability testing.
- **Preventive maintenance -** Actions to reduce future maintenance costs. This refers to actions to reduce future maintenance costs.

What is Portability testing in software?

It refers to the process of testing the ease with which a computer software component or application can be moved from one environment to another, e.g. moving of any application from Windows 2000 to Windows XP. This is usually measured in terms of the maximum amount of effort permitted. Results are measured in terms of the time required to move the software and complete the and documentation updates.

Being able to move software from one machine platform to another either initially or from an existing environment. It refers to system software or application software that can be recompiled for a different platform or to software that is available for two or more different platforms.

The iterative and incremental development cycle implies that portability testing is regularly performed in an iterative and incremental manner.

Portability testing must be automated if adequate regression testing is to occur.

The objectives of Portability testing are to:

- Partially validate the system (i.e., to determine if it fulfills its portability requirements):
    - Determine if the system can be ported to each of its required environments:
        - Hardware ram and disk space
        - Hardware processor and processor speed
        - Monitor resolution
        - Operating system make and version
        - Browser make and version

    - Determine if the look and feel of the webpages is similar and functional in the various browser types and their versions.
- Cause failures concerning the portability requirements that help identify defects that are not efficiently found during unit and integration testing.
- Report these failures to the development teams so that the associated defects can be fixed.
- Help determine the extent to which the system is ready for launch.
- Help provide project status metrics (e.g., percentage of use case paths successfully tested).
- Provide input to the defect trend analysis effort.

Portability tests include tests for:

Installability: Installability testing is conducted on the software used to install other software on its target environment.

Co-existence or compatibility: Co-existence is the software product's capability to co-exists with other independent software products in a common environments sharing common resources**.**

Adaptability: Adaptability is the capability of the software product to be adapted to different specified environments without applying actions or means other than those provided for this purpose for the system.

Replaceability: Replaceability is the capability of the product to be used in place of another specified product for the same purpose in the same environment.

Examples of portability testing of an application that is to be portable across multiple:

- Hardware platforms (including clients, servers, network connectivity devices, input devices, and output devices).
- Operating systems (including versions and service packs).
- Browsers (including both types and versions).

What is Baseline testing in software?

- It is one of the type of non-functional testing.
- It refers to the validation of documents and specifications on which test cases would be designed. The requirement specification validation is baseline testing.
- Generally a baseline is defined as a line that forms the base for any construction or for measurement, comparisons or calculations.
- Baseline testing also helps a great deal in solving most of the problems that are discovered. A majority of the issues are solved through baseline testing.

What is Compliance testing in software testing?

- It is a type of non-functional software testing.
- It is related with the IT standards followed by the company and it is the testing done to find the deviations from the company prescribed standards.
- It determines,whether we are implementing and meeting the defined standards.
- We should take care while doing this testing,Is there any drawbacks in standards implementation in our project and need to do analysis to improve the standards.
- Its basically an audit of a system carried out against a known criterion.

What is documentation testing in software testing?

Documentation testing is a non-functional type of software testing.

- It is a type of non-functional testing.
- Any written or pictorial information describing, defining, specifying, reporting, or certifying activities, requirements, procedures, or results'. Documentation is as important to a product's success as the product itself. If the documentation is poor, non-existent, or wrong, it reflects on the quality of the product and the vendor.
- As per the IEEE Documentation describing plans for, or results of, the testing of a system or component, Types include test case specification, test incident report, test log, test plan, test procedure, test report. Hence the testing of all the above mentioned documents is known as documentation testing.
- This is one of the most cost effective approaches to testing. If the documentation is not right: there will be major and costly problems. The documentation can be tested in a number of different ways to many different degrees of complexity. These range from running the documents through a spelling and grammar checking device, to manually reviewing the documentation to remove any ambiguity or inconsistency.
- Documentation testing can start at the very beginning of the software process and hence save large amounts of money, since the earlier a defect is found the less it will cost to be fixed.

What is Endurance testing in software testing?

Endurance testing is a non functional type of software testing.

- It is a type of non-functional testing.
- It is also known as Soak testing.
- Endurance testing involves testing a system with a significant load extended over a significant period of time, to discover how the system behaves under sustained use. For example, in software testing, a system may behave exactly as expected when tested for 1 hour but when the same system is tested for 3 hours, problems such as memory leaks cause the system to fail or behave randomly.
- The goal is to discover how the system behaves under sustained use. That is, to ensure that the throughput and/or response times after some long period of sustained activity are as good or better than at the beginning of the test.
- It is basically used to check the memory leaks.

What is Load testing in software testing?

- Load testing is a type of non-functional testing.
- A load test is type of software testing which is conducted to understand the behavior of the application under a specific expected load.
- Load testing is performed to determine a system's behavior under both normal and at peak conditions.
- It helps to identify the maximum operating capacity of an application as well as any bottlenecks and determine which element is causing degradation. E.g. If the number of users are increased then how much CPU, memory will be consumed, what is the network and bandwidth response time.
- Load testing can be done under controlled lab conditions to compare the capabilities of different systems or to accurately measure the capabilities of a single system.
- Load testing involves simulating real-life user load for the target application. It helps you determine how your application behaves when multiple users hits it simultaneously.

- Load testing differs from stress testing, which evaluates the extent to which a system keeps working when subjected to extreme work loads or when some of its hardware or software has been compromised.
- The primary goal of load testing is to define the maximum amount of work a system can handle without significant performance degradation.
- Examples of load testing include:
    - Downloading a series of large files from the internet.
    - Running multiple applications on a computer or server simultaneously.
    - Assigning many jobs to a printer in a queue.
    - Subjecting a server to a large amount of traffic.
    - Writing and reading data to and from a hard disk continuously.

What is Performance testing in software?

- It is a type of non-functional testing.
- Performance testing is testing that is performed, to determine how fast some aspect of a system performs under a particular workload.
- It can serve different purposes like it can demonstrate that the system meets performance criteria.
- It can compare two systems to find which performs better. Or it can measure what part of the system or workload causes the system to perform badly.
- This process can involve quantitative tests done in a lab, such as measuring the response time or the number of MIPS (millions of instructions per second) at which a system functions.
- Why to do performance testing:

    - Improve user experience on sites and web apps
    - Increase revenue generated from websites
    - Gather metrics useful for tuning the system
    - Identify bottlenecks such as database configuration
    - Determine if a new release is ready for production
    - Provide reporting to business stakeholders regarding performance against expectations

What is Compatibility testing in software testing?

- It is a type of non-functional testing.
- Compatibility testing is a type of software testing used to ensure compatibility of the system/application/website built with various other objects such as other web browsers, hardware platforms, users (in case if it’s very specific type of requirement, such as a user who speaks and can read only a particular language), operating systems etc. This type of testing helps find out how well a system performs in a particular environment that includes hardware, network, operating system and other software etc.
- It is basically the testing of the application or the product built with the computing environment.
- It tests whether the application or the software product built is compatible with the hardware, operating system, database or other system software or not.

What is Security testing in software testing?

- It is a type of non-functional testing.

- Security testing is basically a type of software testing that's done to check whether the application or the product is secured or not. It checks to see if the application is vulnerable to attacks, if anyone hack the system or login to the application without any authorization.
- It is a process to determine that an information system protects data and maintains functionality as intended.
- The security testing is performed to check whether there is any information leakage in the sense by encrypting the application or using wide range of software's and hardware's and firewall etc.
- Software security is about making software behave in the presence of a malicious attack.
- The six basic security concepts that need to be covered by security testing are: confidentiality, integrity, authentication, availability, authorization and non-repudiation.

What is Scalability testing in software testing?

- It is a type of non-functional testing.
- It is a type of software testing that test the ability of a system, a network, or a process to continue to function well, when it is changed in size or volume in order to meet a growing need.
- It is the testing of a software application for measuring its capability to scale up in terms of any of its non-functional capability like load supported, the number of transactions, the data volume etc.

What is Volume testing in software testing?

- It is a type of non-functional testing.
- Volume testing refers to testing a software application or the product with a certain amount of data. E.g., if we want to volume test our application with a specific database size, we need to expand our database to that size and then test the application's performance on it.
- "Volume testing" is a term given and described in Glenford Myers' *The Art of Software Testing*, 1979. Here's his definition: **"Subjecting the program to heavy volumes of data. The purpose of volume testing is to show that the program cannot handle the volume of data specified in its objectives"** – p. 113.
- The purpose of **volume testing** is to determine system performance with increasing volumes of data in the database.

What is Stress testing in software testing?

- It is a type of non-functional testing.
- It involves testing beyond normal operational capacity, often to a breaking point, in order to observe the results.
- It is a form of software testing that is used to determine the stability of a given system.
- It put greater emphasis on robustness, availability, and error handling under a heavy load, rather than on what would be considered correct behavior under normal circumstances.
- The goals of such tests may be to ensure the software does not crash in conditions of insufficient computational resources (such as memory or disk space).

Difference between Volume, Load and stress testing in software

Very simply we can put the difference between Volume, Load and stress testing as:

Volume Testing = Large amounts of data
Load Testing = Large amount of users
Stress Testing = Too many users, too much data, too little time and too little room

What is Recovery testing in software?

- It is a type of non-functional testing.
- Recovery testing is done in order to check how fast and better the application can recover after it has gone through any type of crash or hardware failure etc.
- Recovery testing is the forced failure of the software in a variety of ways to verify that recovery is properly performed. For example, when an application is receiving data from a network, unplug the connecting cable. After some time, plug the cable back in and analyze the application's ability to continue receiving data from the point at which the network connection got disappeared. Restart the system while a browser has a definite number of sessions and check whether the browser is able to recover all of them or not.

What is Internationalization testing and Localization testing in software?

- It is a type of non-functional testing.
- Internationalization is a process of designing a software application so that it can be adapted to various languages and regions without any changes.
- Whereas Localization is a process of adapting internationalized software for a specific region or language by adding local specific components and translating text.

What is Confirmation testing in software?

Confirmation testing or re-testing: When a test fails because of the defect then that defect is reported and a new version of the software is expected that has had the defect fixed. In this case we need to execute the test again to confirm that whether the defect got actually fixed or not. This is known as confirmation testing and also known as re-testing. It is important to ensure that the test is executed in exactly the same way it was the first time using the same inputs, data and environments.

Hence, when the change is made to the defect in order to fix it then confirmation testing or re-testing is helpful.

What is Regression testing in software?

Regression testing: During confirmation testing the defect got fixed and that part of the application started working as intended. But there might be a possibility that the fix may have introduced or uncovered a different defect elsewhere in the software. The way to detect these '**unexpected side-effects**' of fixes is to do regression testing. The purpose of a regression testing is to verify that modifications in the software or the environment have not caused any unintended adverse side effects and that the system still meets its requirements. Regression testing are mostly automated because in order to fix the defect the same test is carried out again and again and it will be very tedious to do it manually. Regression tests are executed whenever the software changes, either as a result of fixes or new or changed functionality.

What is Structural testing (Testing of software structure/architecture)?

- The structural testing is the testing of the structure of the system or component.
- Structural testing is often referred to as 'white box' or 'glass box' or 'clear-box testing' because in structural testing we are interested in what is happening 'inside the system/application'.
- In structural testing the testers are required to have the knowledge of the internal implementations of the code. Here the testers require knowledge of how the software is implemented, how it works.
- During structural testing the tester is concentrating on how the software does it. For example, a structural technique wants to know how loops in the software are working. Different test cases may be derived to exercise the loop once, twice, and many times. This may be done regardless of the functionality of the software.
- Structural testing can be used at all levels of testing. Developers use structural testing in component testing and component integration testing, especially where there is good tool support for code coverage. Structural testing is also used in system and acceptance testing, but the structures are different. For example, the coverage of menu options or major business transactions could be the structural element in system or acceptance testing.

What is Maintenance Testing?

Once a system is deployed it is in service for years and decades. During this time the system and its operational environment is often corrected, changed or extended. Testing that is provided during this phase is called maintenance testing.

Usually maintenance testing is consisting of two parts:

- **First** one is, testing the changes that has been made because of the correction in the system or if the system is extended or because of some additional features added to it.
- **Second** one is regression tests to prove that the rest of the system has not been affected by the maintenance work.

What is Impact analysis in software testing?

Impact analysis is basically analyzing the impact of the changes in the deployed application or product.

It tells us about the parts of the system that may be unintentionally got affected because of the change in the application and therefore need careful regression testing. This decision is taken together with the stakeholders.