**Software Testing Solved 2019**

**Section A**

1. **A. bottom up**

**B. Black box testing**

**C. To uncover errors**

**D. Capability maturity model**

**E. Unit testing**

**F. CMMI**

**G. User end**

**H. Size of module**

**I. functional incrementation**

**J. Boundary value analysis**

**Section B**

1. **Explain software process model**

A software process model is an abstraction of the software development process. The models specify the stages and order of a process. So, think of this as a representation of the ****order of activities**** of the process and the ****sequence**** in which they are performed.

****A model will define the following:****

* The tasks to be performed
* The input and output of each task
* The pre and post conditions for each task
* The flow and sequence of each task

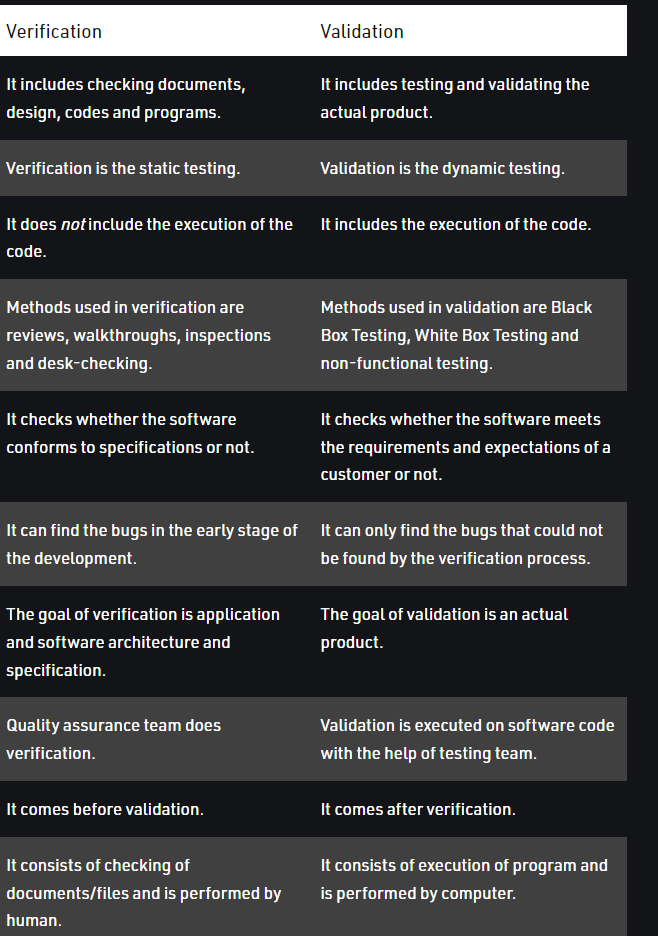
here are many kinds of process models for meeting different requirements. We refer to these as ****SDLC models**** (Software Development Life Cycle models). The most popular and important SDLC models are as follows:

* Waterfall model
* V model
* Incremental model
* RAD model
* Agile model
* Iterative model
* Prototype model
* Spiral model

1. **Describe the terms software verification and validation**

Verification is the process of checking that a software achieves its goal without any bugs. It is the process to ensure whether the product that is developed is right or not. It verifies whether the developed product fulfills the requirements that we have. Verification is static testing.

Validation is the process of checking whether the software product is up to the mark or in other words product has high level requirements. It is the process of checking the validation of product i.e. it checks what we are developing is the right product. it is validation of actual and expected product. Validation is the dynamic testing.



1. **Describe the v process model**

The V-model is an SDLC model where execution of processes happens in a sequential manner in a V-shape. It is also known as **Verification and Validation model**.

The V-Model is an extension of the waterfall model and is based on the association of a testing phase for each corresponding development stage. This means that for every single phase in the development cycle, there is a directly associated testing phase. This is a highly-disciplined model and the next phase starts only after completion of the previous phase.

## V-Model - Design

Under the V-Model, the corresponding testing phase of the development phase is planned in parallel. So, there are Verification phases on one side of the ‘V’ and Validation phases on the other side. The Coding Phase joins the two sides of the V-Model.

The following illustration depicts the different phases in a V-Model of the SDLC.



## V-Model - Verification Phases

There are several Verification phases in the V-Model, each of these are explained in detail below.

### Business Requirement Analysis

This is the first phase in the development cycle where the product requirements are understood from the customer’s perspective. This phase involves detailed communication with the customer to understand his expectations and exact requirement. This is a very important activity and needs to be managed well, as most of the customers are not sure about what exactly they need. The **acceptance test design planning** is done at this stage as business requirements can be used as an input for acceptance testing.

### System Design

Once you have the clear and detailed product requirements, it is time to design the complete system. The system design will have the understanding and detailing the complete hardware and communication setup for the product under development. The system test plan is developed based on the system design. Doing this at an earlier stage leaves more time for the actual test execution later.

### Architectural Design

Architectural specifications are understood and designed in this phase. Usually more than one technical approach is proposed and based on the technical and financial feasibility the final decision is taken. The system design is broken down further into modules taking up different functionality. This is also referred to as **High Level Design (HLD)**.

The data transfer and communication between the internal modules and with the outside world (other systems) is clearly understood and defined in this stage. With this information, integration tests can be designed and documented during this stage.

### Module Design

In this phase, the detailed internal design for all the system modules is specified, referred to as **Low Level Design (LLD)**. It is important that the design is compatible with the other modules in the system architecture and the other external systems. The unit tests are an essential part of any development process and helps eliminate the maximum faults and errors at a very early stage. These unit tests can be designed at this stage based on the internal module designs.

## Coding Phase

The actual coding of the system modules designed in the design phase is taken up in the Coding phase. The best suitable programming language is decided based on the system and architectural requirements.

The coding is performed based on the coding guidelines and standards. The code goes through numerous code reviews and is optimized for best performance before the final build is checked into the repository.

## Validation Phases

The different Validation Phases in a V-Model are explained in detail below.

### Unit Testing

Unit tests designed in the module design phase are executed on the code during this validation phase. Unit testing is the testing at code level and helps eliminate bugs at an early stage, though all defects cannot be uncovered by unit testing.

### Integration Testing

Integration testing is associated with the architectural design phase. Integration tests are performed to test the coexistence and communication of the internal modules within the system.

### System Testing

System testing is directly associated with the system design phase. System tests check the entire system functionality and the communication of the system under development with external systems. Most of the software and hardware compatibility issues can be uncovered during this system test execution.

### Acceptance Testing

Acceptance testing is associated with the business requirement analysis phase and involves testing the product in user environment. Acceptance tests uncover the compatibility issues with the other systems available in the user environment. It also discovers the non-functional issues such as load and performance defects in the actual user environment.

## V- Model ─ Application

V- Model application is almost the same as the waterfall model, as both the models are of sequential type. Requirements have to be very clear before the project starts, because it is usually expensive to go back and make changes. This model is used in the medical development field, as it is strictly a disciplined domain.

The following pointers are some of the most suitable scenarios to use the V-Model application.

Requirements are well defined, clearly documented and fixed.

Product definition is stable.

Technology is not dynamic and is well understood by the project team.

There are no ambiguous or undefined requirements.

The project is short.

## V-Model - Pros and Cons

The advantage of the V-Model method is that it is very easy to understand and apply. The simplicity of this model also makes it easier to manage. The disadvantage is that the model is not flexible to changes and just in case there is a requirement change, which is very common in today’s dynamic world, it becomes very expensive to make the change.

The advantages of the V-Model method are as follows −

This is a highly-disciplined model and Phases are completed one at a time.

Works well for smaller projects where requirements are very well understood.

Simple and easy to understand and use.

Easy to manage due to the rigidity of the model. Each phase has specific deliverables and a review process.

The disadvantages of the V-Model method are as follows −

High 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.

Once an application is in the testing stage, it is difficult to go back and change a functionality.

No working software is produced until late during the life cycle

1. **What is test plan and content available in a test plan ?**

# Test Plan

A test plan is a detailed document which describes software testing areas and activities. It outlines the test strategy, objectives, test schedule, required resources (human resources, software, and hardware), test estimation and test deliverables.

The test plan is a base of every software's testing. It is the most crucial activity which ensures availability of all the lists of planned activities in an appropriate sequence.

The test plan is a template for conducting software testing activities as a defined process that is fully monitored and controlled by the testing manager. The test plan is prepared by the Test Lead (60%), Test Manager(20%), and by the test engineer(20%).

## Types of Test Plan

There are three types of the test plan

* Master Test Plan
* Phase Test Plan
* Testing Type Specific Test Plans

### Master Test Plan

Master Test Plan is a type of test plan that has multiple levels of testing. It includes a complete test strategy.

### Phase Test Plan

A phase test plan is a type of test plan that addresses any one phase of the testing strategy. For example, a list of tools, a list of test cases, etc.

### Specific Test Plans

Specific test plan designed for major types of testing like security testing, load testing, performance testing, etc. In other words, a specific test plan designed for non-functional testing.

1. **Explain the environment of testing**

The test environment is a collection of hardware and software, which helps us to execute the test cases.

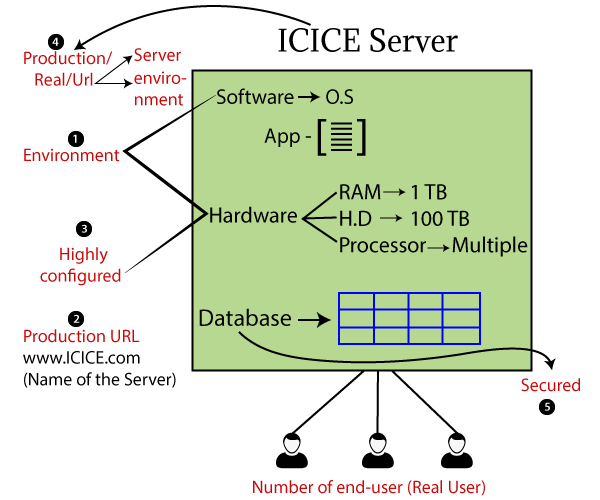
Once we get the requirement from the customer, the developer starts developing the code.

When the coding phase is completed, the application should be installed in the test server. And that application can be accessible with the help of the URL, and the URL could be like this:

****HTTP://COMPUTER NAME/PROJECT/COMPANY NAME****

Suppose we have the ICICE Server, which has been configured with software, hardware, database, and the application.

* The test environment has a software configuration (operating systems), hardware configuration ([RAM](https://www.javatpoint.com/ram), Hard Disk, and Processor), and the test console, which help us to execute the test cases.
* The environment setting is an important part of the testing process because if it is not set correctly, the testing team cannot start the testing process, and the application might be collapse.
* And the Production [URL](https://www.javatpoint.com/url-full-form) could be ****the name of a server**** such as [ICICE.com](http://www.icice.com/" \t "https://www.javatpoint.com/_blank), and ****the environment****.
* The database is secured; that's why we can use it to create, modify, and delete the data.
* This server can be used by n-number of the end-users (real user).



## Software server

Here, we have four types of servers, which are as follows:

* ****Production server****
* ****Development server****
* ****QA Main server****
* ****Staging server****

### Production Server

The server, which consists of software, hardware, and the application development environment, is known as the ****production server****. It is a core server in which the users access any web application or website.

The production server configuration is similar to the staging server, and the application should be debugged and tested on the staging server before dumped into the production server.

****Features of the production server****

* The end-user or real user use this server.
* Security is given to the production server since it contains real user data.
* Highly configured system since it is used by n-number of users.
* The production URL accesses the production server.
* Once the application is developed/tested/stable, then it is deployed into the production environment.

### Development server

Generally, it is accessed by developers for writing new lines of code as well as bug fixes. The dev server gives a run-time environment where the program developed and debugged.

### QA Main server

The test engineers access it for conducting all the types of testing and also obtain by developers for reproducing the bugs. In this server, we will perform the unit and integration testing.

### Test (Staging) server

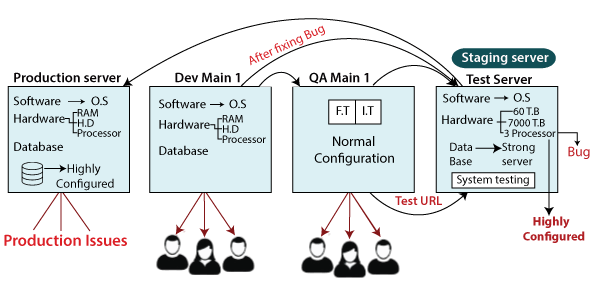
We take another server to check the compatibility of the server because the production server is highly configured, so we need a similar server where we can perform the system testing.

Checking the application compatibility [bugs](https://www.javatpoint.com/bug-in-software-testing) is known as a staging server.

This server helps us to identify the software performance, behavior, bugs, and other issues before the application is deployed into the production server.

## Process of test environment

* Once the requirement is given by the customer and developers finish the coding, we start the initial round of testing on the test server like ****functional/unit, integration**** testing with having normal configuration.
* Here, all the servers software should be the same, but the hardware and the database are similar to the ****production server**** and the ****staging server****.
* For the [system testing](https://www.javatpoint.com/system-testing), we need a highly configured server, which is similar to the production server (similarity to the software and hardware and database) that is called the ****Staging server.****
* In the staging server, we are using the ****test URL**** and perform the ****system testing****.
* If we find any bug, we will be handed over to the developers, and they will fix the bug, and it will directly store in the ****staging test server****.



* In a company, we will use only two environments, which are as follows:
  + Developer environment
  + Testing environment
* In the testing process, we will move to the database from ****QA Main 1**** to the ****Staging server**** to save time, instead of creating a new one using the already existing dev database.
* And then move the application from the test server (staging server) to the Production server.

1. What is the role of quality assurance in software development

[Quality assurance (QA for short)](https://itcraftapps.com/services/quality-assurance/) is the area of production responsible for detecting issues and preventing failures. The goal is to provide the highest quality to the customers and to create a product or service that is compatible with requirements and expectations.

This division is not only present in software houses – it is a part of many other industries, as every company wants to deliver ideal solutions to their customers. Having the manufacturers scan the results of their work to find out if there are any problems with it would take a lot of time and burden them with additional duties.

## The process of quality assurance

What are the elements of quality assurance? Here are essential points to take care of:

* **Planning – establishing a plan for every process that is required to deliver the highest quality we can.**
* **Selecting tools – every tool usage needs to be justified by the requirements of the project.**
* **Training – preparing the team and teaching them about the quality standards they should aspire to.**
* **Inspection – controlling the process while it is happening and looking for improvements on every level.**

1. **Explain the boundary value analysis technique with suitable example**

Boundary value analysis is one of the widely used case design technique for black box testing. It is used to test boundary values because the input values near the boundary have higher chances of error.

Whenever we do the testing by boundary value analysis, the tester focuses on, while entering boundary value whether the software is producing correct output or not.

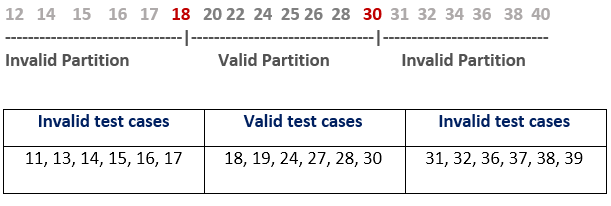
Boundary values are those that contain the upper and lower limit of a variable. Assume that, age is a variable of any function, and its minimum value is 18 and the maximum value is 30, both 18 and 30 will be considered as boundary values.

The basic assumption of boundary value analysis is, the test cases that are created using boundary values are most likely to cause an error.

There is 18 and 30 are the boundary values that's why tester pays more attention to these values, but this doesn't mean that the middle values like 19, 20, 21, 27, 29 are ignored. Test cases are developed for each and every value of the range.

****Let's understand via practical:****

Imagine, there is a function that accepts a number between 18 to 30, where 18 is the minimum and 30 is the maximum value of valid partition, the other values of this partition are 19, 20, 21, 22, 23, 24, 25, 26, 27, 28 and 29. The invalid partition consists of the numbers which are less than 18 such as 12, 14, 15, 16 and 17, and more than 30 such as 31, 32, 34, 36 and 40. Tester develops test cases for both valid and invalid partitions to capture the behavior of the system on different input conditions.



The software system will be passed in the test if it accepts a valid number and gives the desired output, if it is not, then it is unsuccessful. In another scenario, the software system should not accept invalid numbers, and if the entered number is invalid, then it should display error massage.

If the software which is under test, follows all the testing guidelines and specifications then it is sent to the releasing team otherwise to the development team to fix the defects.

1. **What is graph matrix ?how is it related to a connection matrix**

A graph matrix is a data structure that can assist in developing a tool for automation of path testing. Properties of graph matrices are fundamental for developing a test tool and hence graph matrices are very useful in understanding software testing concepts and theory.

What is a Graph Matrix ?

A graph matrix is a square matrix whose size represents the number of nodes in the control flow graph. If you do not know what control flow graphs are, then read this article. Each row and column in the matrix identifies a node and the entries in the matrix represent the edges or links between these nodes. Conventionally, nodes are denoted by digits and edges are denoted by letters.

Ye aadha hai aage wala likh k send karo

**Section c**

1. **what is cyclomatic complexity ?discuss different ways to compute it with example**

Cyclomatic complexity of a code section is the quantitative measure of the number of linearly independent paths in it. It is a software metric used to indicate the complexity of a program. It is computed using the Control Flow Graph of the program. The nodes in the graph indicate the smallest group of commands of a program, and a directed edge in it connects the two nodes i.e. if second command might immediately follow the first command.

For example, if source code contains no control flow statement then its cyclomatic complexity will be 1 and source code contains a single path in it. Similarly, if the source code contains one **if condition** then cyclomatic complexity will be 2 because there will be two paths one for true and the other for false.

Mathematically, for a structured program, the directed graph inside control flow is the edge joining two basic blocks of the program as control may pass from first to second.   
So, cyclomatic complexity M would be defined as,

***M = E – N + 2P***

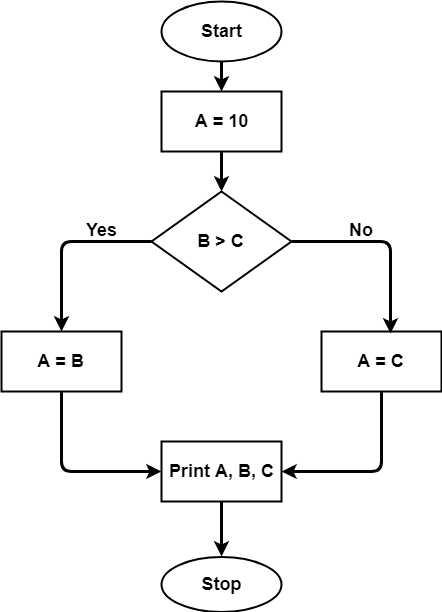
*where,   
E = the number of edges in the control flow graph   
N = the number of nodes in the control flow graph   
P = the number of connected components*

Steps that should be followed in calculating cyclomatic complexity and test cases design are: 

* Construction of graph with nodes and edges from code.
* Identification of independent paths.
* Cyclomatic Complexity Calculation
* Design of Test Cases

Let a section of code as such: 

**Control Flow Graph** of above code



The cyclomatic complexity calculated for above code will be from control flow graph. The graph shows seven shapes(nodes), seven lines(edges), hence cyclomatic complexity is 7-7+2 = 2.

**Use of Cyclomatic Complexity:** 

* Determining the independent path executions thus proven to be very helpful for Developers and Testers.
* It can make sure that every path have been tested at least once.
* Thus help to focus more on uncovered paths.
* Code coverage can be improved.
* Risk associated with program can be evaluated.
* These metrics being used earlier in the program helps in reducing the risks.

**Advantages of Cyclomatic Complexity:**.

* It can be used as a quality metric, gives relative complexity of various designs.
* It is able to compute faster than the Halstead’s metrics.
* It is used to measure the minimum effort and best areas of concentration for testing.
* It is able to guide the testing process.
* It is easy to apply.

**Disadvantages of Cyclomatic Complexity:**

* It is the measure of the programs’s control complexity and not the data complexity.
* In this, nested conditional structures are harder to understand than non-nested structures.
* In case of simple comparisons and decision structures, it may give a misleading figure.

1. **Describe the various types of testing during software development**

### **Accessibility Testing**

Accessibility testing is the practice of ensuring your mobile and web apps are working and usable for users without and with disabilities such as vision impairment, hearing disabilities, and other physical or cognitive conditions.

### **Acceptance Testing**

Acceptance testing ensures that the end-user (customers) can achieve the goals set in the business requirements, which determines whether the software is acceptable for delivery or not. It is also known as user acceptance testing (UAT).

### **Black Box Testing**

Black box testing involves testing against a system where the code and paths are invisible.

### **End to End Testing**

End to end testing is a technique that tests the application’s workflow from beginning to end to make sure everything functions as expected.

### **Functional Testing**

Functional testing checks an application, website, or system to ensure it’s doing exactly what it’s supposed to be doing.

### **Interactive Testing**

Also known as manual testing, interactive testing enables testers to create and facilitate manual tests for those who do not use automation and collect results from external tests.

### **Integration Testing**

Integration testing ensures that an entire, integrated system meets a set of requirements. It is performed in an integrated hardware and software environment to ensure that the entire system functions properly.

### **Load Testing**

This type of non-functional software testing process determines how the software application behaves while being accessed by multiple users simultaneously.

### **Non Functional Testing**

Non functional testing verifies the readiness of a system according to nonfunctional parameters (performance, accessibility, UX, etc.)  which are never addressed by functional testing.

### **Performance Testing**

Performance testing examines the speed, stability, reliability, scalability, and resource usage of a software application under a specified workload.

### **Regression Testing**

Regression testing is performed to determine if code modifications break an application or consume resources.

### **Sanity Testing**

Performed after bug fixes, sanity testing determines that the bugs are fixed and that no further issues are introduced to these changes.

### **Security Testing**

Security testing unveils the vulnerabilities of the system to ensure that the software system and application are free from any threats or risks. These tests aim to find any potential flaws and weaknesses in the software system that could lead to a loss of data, revenue, or reputation per employees or outsides of a company.

### **Single User Performance Testing**

Single user performance testing checks that the application under test performs fine according to specified threshold without any system load. This benchmark can be then used to define a realistic threshold when the system is under load.

### **Smoke Testing**

This type of software testing validates the stability of a software application, it is performed on the initial software build to ensure that the critical functions of the program are working.

### **Stress Testing**

Stress testing is a software testing activity that tests beyond normal operational capacity to test the results.

### **Unit Testing**

Unit testing is the process of checking small pieces of code to ensure that the individual parts of a program work properly on their own, speeding up testing strategies and reducing wasted tests.

### **White Box Testing**

White box testing involves testing the product's underlying structure, architecture, and code to validate input-output flow and enhance design, usability, and security.

1. **What are the different types of structural testing techniques?Discuss any two technique with the with the help of example**

Another type of **[software testing](https://www.javatpoint.com/software-testing-tutorial)** technique is ****Structural testing****, which is used to test the internal design of the software or structure of the coding for the particular software.

In this testing, the development team members are included in the testing team to execute the software's internal design. The working of structural testing is opposite to ****Behavioral testing****.

In other words, we can say that structural testing tests the different features of an application based on its types.

Structural testing is also known as white-box testing, ****glass box testing****, and ****clear-box testing****. Developers mostly implement it to identify the issue and fix them quickly.

The structural testing process requires an in-depth knowledge of the programming language and is opposite to Functional Testing.

The knowledge of the code's internal executions and how the software is implemented is a necessity for the test engineer to implement the structural testing.

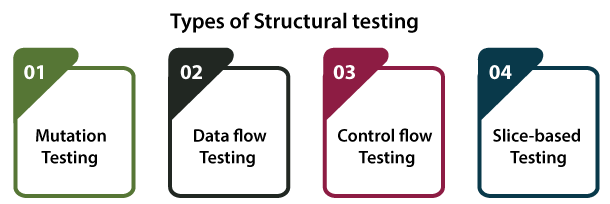
Throughout the structural testing, the test engineer intends on how the software performs, and it can be used at all levels of testing.

****For example****, the coverage of menu options or significant business transactions could be the system's structural element or acceptance testing.

## Types of Structural Testing

Structural testing is divided into four different categories, which are as follows:

* ****Mutation testing****
* ****Data flow testing****
* ****Control flow testing****
* ****Slice-based testing****



### Mutation testing

* It is used to check the quality of the test case that should fail the mutant code.
* Mutation testing involves the development of new tests to be implemented on the software for its testing process.
* When we identify various errors, it implies that either the program is correct or the test case is inefficient in locating the fault.
* In the mutation testing, the developers make small modifications to the previously accessible software tests and generate a mutant of the old software test.
* It used to cause an error in the program, which implies that the mutation testing is performed to evaluate the test case's productivity.

### Data flow testing

* It is a group of testing approaches used to observe the control flow of programs to discover the sequence of variables as per the series of events.
* It implements a control flow graph and analysis the points where the codes can change the data.
* If we execute the data flow testing technique, the information is kept safe and unchanged during the code's implementation.

### Control flow testing

* The ****control flow testing**** is the basic model of ****Structural testing****.
* It is to check the implementation order of commands or statements of the code over a control structure.
* In the control flow testing, a specific part of an extensive program is selected by the test engineer to set the testing path.
* Generally, the control flow testing technique is used in unit testing.
* In this testing, the entire test is based on how the control is executed during the code.
* The complete information of all the software's features and logic is necessary to execute the control flow testing.

### Slice-based testing

* It was initially created and established to keep the software.
* The basic idea is to sort the complete code into small chunks and then evaluate each portion carefully.
* The slice-based testing is very beneficial for the maintenance of the software along with fixing the application too.

## Structural Testing Tools

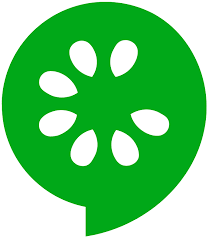
Like other testing has their tools, structural testing also contains some open-source and commercial tools that have their functionality.

****Some of the most commonly used tools for structural testing are as follows:****

* ****Cucumber****
* ****JBehave****
* ****Cfix****
* ****JUnit****

Let understand them one by one in details:

### Cucumber



It is a widely used tool for behavior-driven Development as it delivers an easily understandable testing script for ****system acceptance**** and automation testing.

It is a software tool used by the test engineer to develop test cases for the testing of the software's behavior.

For more information about the cucumber tool, we can refer to the following link: <https://www.javatpoint.com/cucumber-testing-introduction>

### JBehave

It is a framework for BDD (****Behaviour-Driven Development****). The BDD is a development of ****TDD (test-driven Development****) and acceptance-test-driven design. It is planned to create these practices more accessible and spontaneous to beginners and professionals alike.



****Features of JBehave****

Some of the commonly used features of JBehave are as follows:

* It is purely executing in Java programming language, which plays well with Java-based enterprises.
* In this, we can write the user stories in ****JBehave syntax/ Gherkin syntax****.
* The user stories can be implemented as ****JUnit****
* It provides the Annotation-based configuration and steps class specifications.
* It allows stories to be executed through Ant task.

### Cfix

Another structural testing tool is ****Cfix,**** an XUnit testing framework supported by the [C](https://www.javatpoint.com/c-programming-language-tutorial)/[C++ programming language](https://www.javatpoint.com/cpp-tutorial).

In this tool, the unit tests are compiled and linked into a DLL. It has been designed to work well in combination with ****Windows Debuggers**** such as ****Visual Studio**** and ****WinDBG****.

****Characteristic of Cfix****

Following are the commonly used features of Cfix:

* It supports development of both user and kernel mode unit tests.
* The failing test case can be highly customized in case of implementation.

### JUnit



JUnit is one of the essential tools of structural testing. It is an open-source unit testing framework, which was written in Java language.

It will help us to enhance the developer's efficiency, which provides the consistency of the development code and reduces the time of the debugging.

For more details about the ****JUnit,**** refers to the below link: <https://www.javatpoint.com/junit-tutorial>

## Advantages and Disadvantages of Structural Testing

Below are the benefits and drawback of structural testing:

### Advantages of Structural Testing

The benefits of Structural testing are as follows:

* Structural testing does not require a lot of manual work as it is an automated process.
* Structural testing is not a time-consuming process.
* All the early defects can easily be identified
* It removes the dead code (extra code) or statements easily.
* It provides easy coding and implementation.
* It delivers detailed testing of the software.

### Disadvantages of Structural Testing

The ****drawback**** of the structural testing are as follows:

* To perform structural testing, in-depth knowledge of programming languages is required.
* Even though structural testing is automatic, it might turn out very difficult because it involves training in the tool used for testing.
* It is expensive in respect of money because sometimes resources are necessary to efficiently perform structural testing.
* There is also a small chance that some commands, statements or branches could be missed unintentionally.

1. **What are the various type of functional testing technique ?**

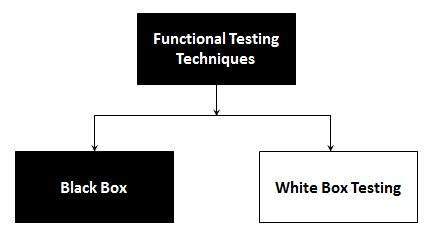
**Also explain the boundary value analysis technique with suitable example**

## What is Functional Testing?

Functional Testing is a testing technique that is used to test the features/functionality of the system or Software, should cover all the scenarios including failure paths and boundary cases.

## Functional Testing Techniques:

There are two major Functional Testing techniques as shown below:



The other major Functional Testing techniques include:

Unit Testing

Integration Testing

Smoke Testing

User Acceptance Testing

Localization Testing

Interface Testing

Usability Testing

System Testing

Regression Testing

Globalization Testing

## Boundary Value Analysis (BVA):

BVA is another Black Box Test Design Technique, which is used to find the errors at boundaries of input domain (tests the behavior of a program at the input boundaries) rather than finding those errors in the centre of input. So, the basic idea in boundary value testing is to select input variable values at their: minimum, just above the minimum, just below the minimum, a nominal value, just below the maximum, maximum and just above the maximum. That is, for each range, there are two boundaries, the lower boundary (start of the range) and the upper boundary (end of the range) and the boundaries are the beginning and end of each valid partition. We should design test cases which exercise the program functionality at the boundaries, and with values just inside and outside the boundaries. Boundary value analysis is also a part of stress and negative testing.

Suppose, if the input is a set of values between A and B, then design test cases for A, A+1, A-1 and B, B+1, B-1.

Example:

