**Notes for MCA-II (Semester- III)**

**Subject :- Software Testing & Quality Assurance**

**(Subject Code:- IT-33)**

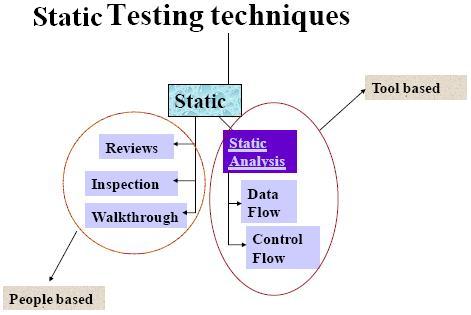
**Chapter: 3] Static Testing**

* **3.1. Static Techniques – Review :-**
* **3.1.1 Review Process (Informal & Formal) :-**

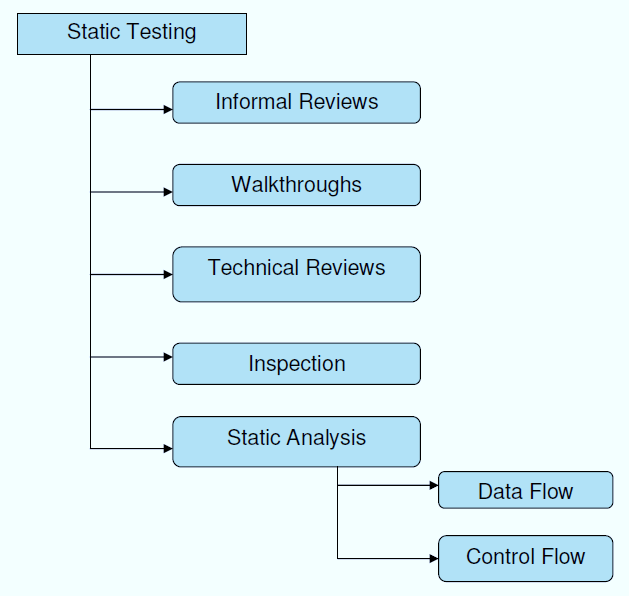
**Static Testing** is a software testing technique which is used to check defects in software application without executing the code. Static testing is done to avoid errors at an early stage of development as it is easier to identify the errors and solve the errors.

The two main types of static testing techniques are

* **Manual examinations**: Manual examinations include analysis of code done manually, also known as **REVIEWS.**
* **Automated analysis using tools:** Automated analysis are basically static analysis which is done using tools.
* Under Static Testing code is not executed. Rather it manually checks the code, requirement documents, and design documents to find errors. Hence, the name "static".
* Main objective of this testing is to improve the quality of software products by finding errors in early stages of the development cycle. This testing is also called as Non-execution technique or verification testing.
* Static testing involves manual or automated reviews of the documents. This review is done during initial phase of testing to catch defect early in STLC. It examines work documents and provides review comments.



A review in a Static Testing is a process or meeting conducted to find the potential defects in the design of any program. Another significance of review is that all the team members get to know about the progress of the project and sometimes the diversity of thoughts may result in excellent suggestions.



Software review is an important part of [**"Software Development Life Cycle (SDLC)"**](https://www.professionalqa.com/software-development-life-cycle) that assists software engineers in validating the quality, functionality, and other vital features and components of the software. it is a complete process that involves testing the software product and ensuring that it meets the requirements stated by the client.

Software Review is systematic inspection of a software by one or more individuals who work together to find and resolve errors and defects in the software during the early stages of Software Development Life Cycle (SDLC). Software review is an essential part of Software Development Life Cycle (SDLC) that helps software engineers in validating the quality, functionality and other vital features and components of the software. It is a whole process that includes testing the software product and it makes sure that it meets the requirements stated by the client.

Usually performed manually, software review is used to verify various documents like requirements, system designs, codes, test plans and test cases.

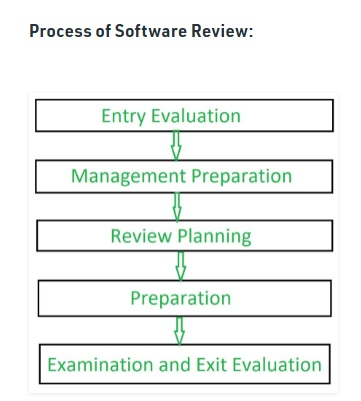
**Objectives of Software Review:**

1. To improve the productivity of the development team.

1. To make the testing process time and cost effective.

1. To make the final software with fewer defects.

1. To eliminate the inadequacies.



**Why is Software Review Important?**

* It improves the productivity of the development team.
* Makes the process of testing time & cost effective, as more time is spent on testing the software during the initial development of the product.
* Fewer defects are found in the final software, which helps reduce the cost of the whole process.
* The reviews provided at this stage are found to be cost effective, as they are identified at the earlier stage, as the cost of rectifying a defect in the later stages would be much more than doing it in the initial stages.
* In this process of reviewing software, often we train technical authors for defect detection process as well as for [**"defect prevention process"**](https://www.professionalqa.com/defect-analysis-and-prevention).
* It is only at this stage the inadequacies are eliminated.
* Elimination of defects or errors can benefit the software to a great extent. Frequent check of samples of work and identification of small time errors can lead to low error rate.
* As a matter of fact, this process results in dramatic reduction of time taken in producing a technically sound document.

### Formal Review:

A type of peer review, [**"formal review"**](https://www.professionalqa.com/activities-of-formal-review) follows a formal process and has a specific formal agenda. It has a well structured and regulated process, which is usually implemented at the end of each life cycle. During this process, a formal review panel or board considers the necessary steps for the next life cycle.

#### Features of Formal Review:

* This evaluates conformance to specification and various standards.
* Conducted by a group of 3 or more individuals.
* The review team petitions the management of technical leadership to act on the suggested recommendations.
* Here, the leader verifies that the action documents are verified and incorporated into external processes.
* Formal review consists of six important steps, which are:
  + Planning.
  + Kick-off.
  + Preparation.
  + Review meeting.
  + Rework.
  + Follow up.

### Informal Review:

Unlike Formal Reviews, Informal reviews are applied multiple times during the early stages of software development process. The major difference between the **formal and informal reviews** is that the former follows a formal agenda, whereas the latter is conducted as per the need of the team and follows an informal agenda. Though time saving, this process is not documented and does not require any entry criteria or large group of members.

Informal Review:

* Informal reviews are applied many times during the early stages of the life cycle of the document.
* A two person team can conduct an informal review.
* In later stages these reviews often involve more people and a meeting. The goal is to keep the author and to improve the quality of the document.
* The most important thing to keep in mind about the informal reviews is that they are not documented.
* Key characteristics of Informal Review:
  + Generally one to one meeting
  + No agenda
  + Results are not formally reported
  + Occurs as needed throughout each phase.

#### Features of Informal Review:

* Conducted by a group of 2-7 members, which includes the designer an any other interested party.
* Here the team identifies errors & issues as well as examine alternatives.
* It is a forum for learning.
* All the changes are made by the software designer.
* These changes are verified by other project controls.
* The role of informal review is to keep the author informed and to improve the quality of the product.

## Process of Software Review:

The process of software review is a simple one and is common for all its types. It is usually implemented by following a set of activities, which are laid down by IEEE Standard 1028. All these steps are extremely important and need to be followed rigorously, as skipping even a single step can lead to a complication with the development process, which can further affect the quality of the end product.

1. **Entry Evaluation:**

A standard check-list is used by entry criteria in order to ensure an ideal condition for a successful review.

1. **Management Preparation:**

During this stage of the process, a responsible management ensures that the software review has all the required resources, which includes things like staff, time, materials, and tools.

1. **Review Planning:**

To undergo a software review, an objective is identified. Based on the objective, a recognized team of resources is formed.

1. **Preparation:**

The reviewers are held responsible for preparing group examination to do the reviewing task.

1. **Examination and Exit Evaluation:**

In the end, the result made by each reviewer is combined all together. Before the review is finalized, verification of all activities is done that are considered necessary for an efficacious software review.

|  |  |
| --- | --- |
| **Informal reviews** | **Formal reviews** |
| It is a type of review that typically occurs spontaneously among teams. | It is a type of review that is done by a team of software testers or the reviewer’s team. |
| It is generally done by Software developers or engineers. | It is done by the Reviewers and they are responsible for this all review. |
| Here, no report is created as it is informal. | Here, a report is created as it is formal. |

**3.1.2 Desk Checking :-** Desk checking is an informal manual test that programmers can use to verify coding and algorithm logic before a program launch. This enables them to spot errors that might prevent a program from working as it should. Modern debugging tools make desk checking less essential than it was in the past, but it can still be a useful way of spotting logic errors.

## Desk Checking Overview

Desk checking is a similar process to proofreading; in this exercise, the programmer runs through lines of code to identify errors and to check logic. Typically, the programmer will print out the code and go through it in a pencil and paper exercise. He may run a manual test on algorithms, checking that they work correctly and contain no coding errors. This usually involves creating a table with columns containing line numbers, variables, conditions, and inputs and outputs, depending on the checks he is making.

## Advantages of Desk Checking :-

Even experienced programmers make mistakes -- a desk check may help catch and fix them before a program goes through a formal run. Running a desk check is quick and inexpensive. The programmer who wrote the code typically checks it herself; if she identifies issues, she can fix them on the spot before the project moves onto the next stage. If she doesn't desk check and an error causes problems later down the line, it might delay a project. Errors may also be harder to identify at a later stage.

## Disadvantages of Desk Checking:-

A desk check does not guarantee that a programmer will find mistakes. It is also subject to human error. Programmers may miss things that need to be fixed, simply because they wrote the code themselves and are too close to it to be objective. Getting a different programmer to desk check may solve this issue. However, the person running the check also needs to understand the requirements behind the code before he can evaluate if it will work.

## Desk Checking and the Structured Walkthrough :-

Desk checking is sometimes part of a broader testing process. In a structured walkthrough, for example, the programmer is part of a peer group that reviews and analyses the work prior to launch. The programmer typically gives the materials for review to group members before the meeting. During the meeting itself, she walks the group through the code. Ideally, the group will spot errors if they exist or make viable suggestions for improvement. Projects may have one or more walkthrough stages, checking issues such as the understanding of requirements and coding accuracy.

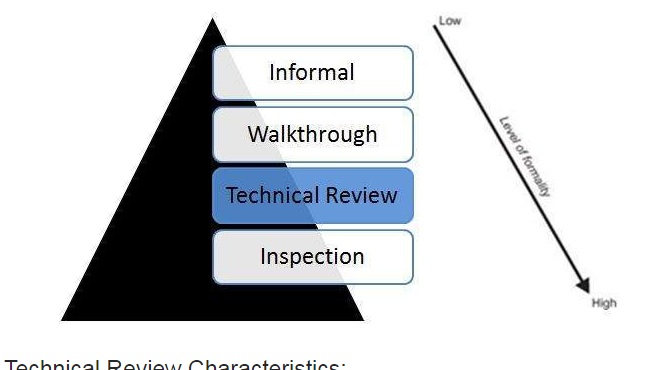
* **3.1.3 Technical or Peer Review :-**

What is Technical review?

A Technical review is a static white-box testing technique which is conducted to spot the defects early in the life cycle that cannot be detected by black box testing techniques.

A software technical review is examined by a team of qualified software engineers for the suitability of the software product. This process can also be defined as a critical evaluation of an object in the software. Through the software technical review process, we can identify the errors or defects in the software product in the early phase itself.

Technical Review - Static Testing:



Technical Review Characteristics:

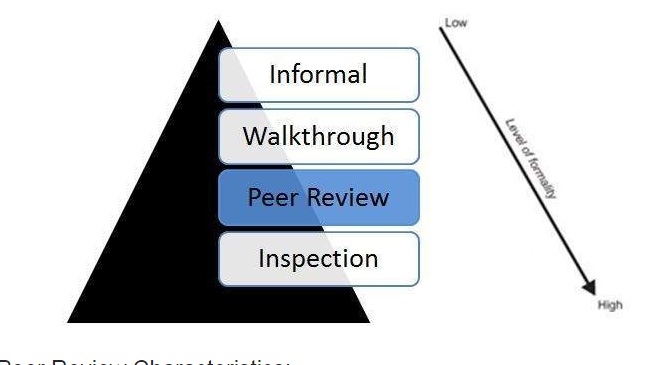
* Technical Reviews are documented and uses a defect detection process that has peers and technical specialist as part of the review process.
* The Review process doesn't involve management participation.
* It is usually led by trained moderator who is NOT the author.
* The report is prepared with the list of issues that needs to be addressed.

## What is Peer Review?

A peer review, a review technique, which is a static white-box testing which are conducted to spot the defects early in the life cycle that cannot be detected by black box testing techniques.

Peer review is the process of assessing the technical content and quality of the product and it is usually conducted by the author of the work product along with some other developers.

## Peer Review - Static Testing

****

## Peer Review Characteristics:

* Peer Reviews are documented and uses a defect detection process that has peers and technical specialist as part of the review process.
* The Review process doesn't involve management participation.
* It is usually led by trained moderator who is NOT the author.
* The report is prepared with the list of issues that needs to be addressed.

**Technical or peer review:**

* A technical review is a discussion meeting that focuses on achieving consensus about the technical content of a document.
* Compared to inspections, technical reviews are less formal and there is little or no focus on defect identification on the basis of referenced documents, intended readership and rules.
* During technical reviews defects are found by experts, who focus on the content of the document.
* The experts that are needed for a technical review are:
  + architects
  + chief designers
  + key users.
* In practice, technical reviews vary from quite informal to very formal.
* **The goals of a technical review are to**:
  + assess the value of technical concepts and alternatives in the product and project environment;
  + establish consistency in the use and representation of technical concepts;
  + ensure, at an early stage, that technical concepts are used correctly;
  + Inform participants of the technical content of the document.
* **Key characteristics of a technical review are**:
  + It is a documented defect-detection process that involves peers and technical experts.
  + It is often performed as a peer review without management participation.
  + Ideally it is led by a trained moderator, but possibly also by a technical expert.
  + A separate preparation is carried out during which the product is examined and the defects are found.
  + More formal characteristics such as the use of checklists and a logging list or issue log are optional.
* **3.1.4 Walkthrough :-**

Walkthrough is a method of conducting informal group/individual review. In a walkthrough, author describes and explain work product in a informal meeting to his peers or supervisor to get feedback. Here, validity of the proposed solution for work product is checked.

It is cheaper to make changes when design is on the paper rather than at time of conversion. Walkthrough is a static method of quality assurance. Walkthrough are informal meetings but with purpose.

in which a designer or programmer leads members of the development team and other interested parties through a software product, and the participants ask questions and make comments about possible errors, violation of development standards, and other problems.

* A walkthrough is characterized by the author of the document under review guiding the participants through the document and his or her thought processes, to achieve a common understanding and to gather feedback.
* This is especially useful if people from outside the software discipline are present, who are not used to, or cannot easily understand software development documents.
* The content of the document is explained step by step by the author, to reach consensus on changes or to gather information.
* Within a walkthrough the author does most of the preparation.
* The participants, who are selected from different departments and backgrounds, are not required to do a detailed study of the documents in advance.
* Because of the way the meeting is structured, a large number of people can participate and this larger audience can bring a great number of diverse viewpoints regarding the contents of the document being reviewed as well as serving an educational purpose.
* If the audience represents a broad cross-section of skills and disciplines, it can give assurance that no major defects are 'missed' in the walk-through.
* A walkthrough is especially useful for higher-level documents, such as requirement specifications and architectural documents.
* The specific goals of a walkthrough depend on its role in the creation of the document. In general the following goals can be applicable:
  + to present the document to stakeholders both within and outside the soft ware discipline, in order to gather information regarding the topic under documentation;
  + to explain (knowledge transfer) and evaluate the contents of the document;
  + to establish a common understanding of the document;
  + to examine and discuss the validity of proposed solutions and the viability of alternatives, establishing consensus.
* Key characteristics of walkthroughs are:
  + The meeting is led by the authors; often a separate scribe is present.
  + Scenarios and dry runs may be used to validate the content.
  + Separate pre-meeting preparation for reviewers is optional.

Review Meeting:

* This meeting typically consists of the following elements:-
  + -logging phase
  + -discussion phase
  + -decision phase.
* During the logging phase the issues, e.g. defects, that have been identified during the preparation are mentioned page by page, reviewer by reviewer and are logged either by the author or by a scribe. This phase is for just jot down all the issues not to discuss them in detail.
* If an issue needs discussion, the item is logged and then handled in the discussion phase.
* A detailed discussion on whether or not an issue is a defect is not very meaningful, as it is much more efficient to simply log it and proceed to the next one.
* The issues classified as discussion items will be handled during discussion phase.
* Participants can take part in the discussion by bringing forward their comments and reasoning.
* The moderator also paces this part of the meeting and ensures that all discussed items either have an outcome by the end of the meeting, or are defined as an action point if a discussion cannot be solved during the meeting.
* The outcome of discussions is documented for future reference.
* At the end of the meeting, a decision on the document under review has to be made by the participants, sometimes based on formal exit criteria.
* The most important exit criterion is the average number of critical and major defects found per page.
* If the number of defects found per page exceeds a certain level, the document must be reviewed again, after it has been reworked. If the document complies with the exit criteria, the document will be checked during follow-up by the moderator or one or more participants. Subsequently, the document can leave or exit the review process.

## Walkthrough Participants:

* **Author** - The Author of the document under review.
* **Presenter** - The presenter usually develops the agenda for the walkthrough and presents the output being reviewed.
* **Moderator** - The moderator facilitates the walkthrough session, ensures the walkthrough agenda is followed, and encourages all the reviewers to participate.
* **Reviewers** - The reviewers evaluate the document under test to determine if it is technically accurate.
* **Scribe** - The scribe is the recorder of the structured walkthrough outcomes who records the issues identified and any other technical comments, suggestions, and unresolved questions.
* **3.1.5 Inspection :-**

An inspection is defined as formal, rigorous, in depth group review designed to identify problems as close to their point of origin as possible. Inspections improve reliability, availability, and maintainability of [software product](https://www.geeksforgeeks.org/software-engineering-software-product/).

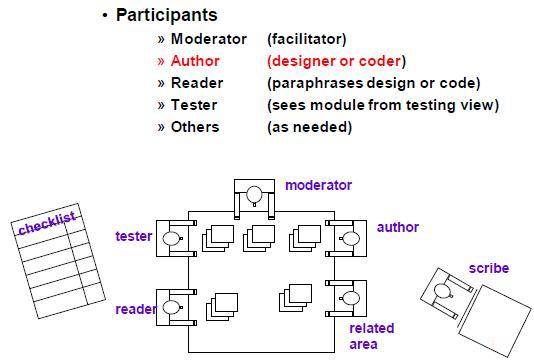
Anything readable that is produced during the [software development](https://www.geeksforgeeks.org/software-development-life-cycle-sdlc/) can be inspected. Inspections can be combined with structured, systematic testing to provide a powerful tool for creating defect-free programs.

Inspection activity follows a specified process and participants play well-defined roles.  
An inspection team consists of three to eight members who plays roles of moderator, author, reader, recorder and inspector.

**For example,** designer can acts as inspector during code inspections while a quality assurance representative can act as standard enforcer.

**Stages in the inspections process :**

* **Planning :** Inspection is planned by moderator.
* **Overview meeting :** Author describes background of work product.
* **Preparation :** Each inspector examines work product to identify possible defects.
* **Inspection meeting :** During this meeting, reader reads through work product, part by part and inspectors points out the defects for every part.
* **Rework :** Author makes changes to work product according to action plans from the inspection meeting.
* **Follow-up :** Changes made by author are checked to make sure that everything is correct.
* A Software inspection is a formal evaluation technique in which software requirements, design, or code is examined in detail by a person or group other than the author to detect faults, violations of development standards, and other problems.”
* Detect, identify, and describe defects
* Collect metrics data
* Verify “fitness for use” in subsequent efforts
* Providecontrols that determines the next step
* Essential requirements
* Definition of development process and exit criteria
* Description of the inspection process
* Correct execution of the process
* Common inspections:
* High-level design inspection
* Low-level design inspection
* Code inspection



**Figure : Inspection Team**

* **Inspection Moderator’s Duties:**

•Prior to inspection

* Be trained
* Determine if entry criteria met
* Work with author to establish team membership
* Preview material for conformance to standards
* Establish inspection time, place
* Ensure materials are distributed

•During inspection

* Ensure attendance & preparation; or postpone
* Facilitate meeting
* Log defects
* Require re-inspection of major defects

•After inspection

* Review results with author
* Provide manager with estimate of rework
* Write inspection summary and distribute
* **Inspection Participants: Reader**

•Prior to inspection

* Review material

•During inspection

* Guide inspection team through material
* Paraphrase or verbalize the review material
* Present material with clarity and understanding
* Note any items difficult to understand
* Be prepared to tie back to specification or design

•After inspection

* Assist in verifying follow-up items are completed.
* **Inspection Participants: Tester**
* Determines if inspected item can be verified
* Ensures that code is compatible within the system
* Understands verification practices
* Enforces them
* **Inspection Process:**

•Planning

* Ensure materials meet inspection criteria
* Arrange availability of participants
* Arrange suitable time and place

•Overview

* Educate participants on what is to be inspected.
* Assign inspection roles

•Preparation

* Participants learn material and prepare to fulfill assigned roles

•Execution

* Introductions and verification of preparedness
* Review software and record defects
* Review defect list
* Make exit decision
* Accept “as is” or with minor rework.
* Accept after moderator verifies rework
* Re-inspect reworked software element

•Rework

* Author addresses all defects

•Follow-up

* Verify fixes are effective without secondary defects Inspection.

## Characteristics of Inspection :

* Inspection is usually led by a trained moderator, who is not the author. Moderator's role is to do a peer examination of a document
* Inspection is most formal and driven by checklists and rules.
* This review process makes use of entry and exit criteria.
* It is essential to have a pre-meeting preparation.
* Inspection report is prepared and shared with the author for appropriate actions.
* Post Inspection, a formal follow-up process is used to ensure a timely and a prompt corrective action.
* Aim of Inspection is NOT only to identify defects but also to bring in for process improvement.
* **3.2. Static Techniques – Static Analysis**

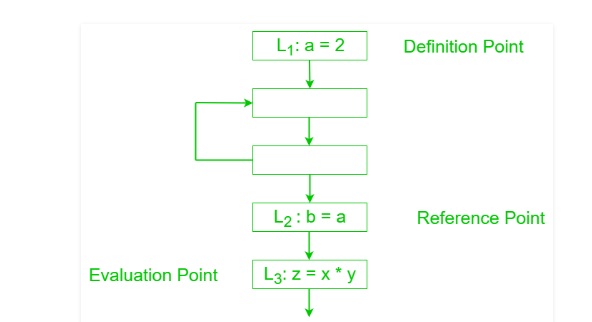
**3.2.1 Data Flow Analysis :-**

Data-flow analysis is a technique for gathering information about the possible set of values calculated at various points in a [computer program](https://en.wikipedia.org/wiki/Computer_program). It attempts to obtain particular information at each point in a procedure. Usually, it is enough to obtain this information at the boundaries of [basic blocks](https://en.wikipedia.org/wiki/Basic_block), since from that it is easy to compute the information at points in the basic block.

It is the analysis of flow of data in control flow graph, i.e., the analysis that determines the information regarding the definition and use of data in program. With the help of this analysis, optimization can be done. In general, its process in which values are computed using data flow analysis. The data flow property represents information that can be used for optimization.

Basic Terminologies –

* **Definition Point**: a point in a program containing some definition.
* **Reference Point**: a point in a program containing a reference to a data item.
* **Evaluation Point**: a point in a program containing evaluation of expression.



Data flow testing is a family of test strategies based on selecting paths through the program's control flow in order to explore sequences of events related to the status of variables or data objects. Dataflow Testing focuses on the points at which variables receive values and the points at which these values are used.

## Advantages of Data Flow Testing:

Data Flow testing helps us to pinpoint any of the following issues:

* A variable that is declared but never used within the program.
* A variable that is used but never declared.
* A variable that is defined multiple times before it is used.
* De-allocating a variable before it is used.
* To efficiently optimize the code compiler collects all the information about the program and distribute this information to each block of the flow graph. This process is known as data-flow graph analysis.
* Certain optimization can only be achieved by examining the entire program. It can't be achieve by examining just a portion of the program.
* For this kind of optimization user defined chaining is one particular problem.
* Here using the value of the variable, we try to find out that which definition of a variable is applicable in a statement.
* Data-flow analysis can be used to increase program understanding and to develop test cases based on data flow within the program.
* The data-flow testing technique is based on investigating the ways values are associated with variables and the ways that these associations affect the execution of the program.
* Data-flow analysis focuses on occurrences of variables, following paths from the definition (or initialization) of a variable to its uses.
* The variable values may be used for computing values for defining other variables or used as predicate variables to decide whether a predicate is true for traversing a specific execution path.
* A data-flow analysis for an entire program involving all variables and traversing all usage paths requires immense computational resources; however, this technique can be applied for select variables.
* The simplest approach is to validate the usage of select sets of variables by executing a path that starts with definition and ends at uses of the definition.
* The path and the usage of the data can help in identifying suspicious code blocks and in developing test cases to validate the runtime behavior of the software.
* For example, for a chosen data definition-to-use path, with well-crafted test data, testing can uncover time-of-check-to-time-of-use (TOCTTOU) flaws.
* Data flow testing is the name given to a family of test strategies based on selecting paths through the program's control flow in order to explore sequences of events related to the status of data objects.
* For example, pick enough paths to assure that every data object has been initialized prior to use or that all defined objects have been used for something.
* **3.2.2 Control-Flow Analysis (CFA) :-**

It is a [static-code-analysis](https://en.wikipedia.org/wiki/Static_code_analysis) technique for determining the [control flow](https://en.wikipedia.org/wiki/Control_flow) of a program. The control flow is expressed as a [control-flow graph](https://en.wikipedia.org/wiki/Control-flow_graph) (CFG).

Control flow testing is a testing technique that comes under white box testing. The aim of this technique is to determine the execution order of statements or instructions of the program through a control structure. The control structure of a program is used to develop a test case for the program. In this technique, a particular part of a large program is selected by the tester to set the testing path. It is mostly used in unit testing. Test cases represented by the control graph of the program.

Control Flow Graph is formed from the node, edge, decision node, junction node to specify all possible execution path.

Notations used for Control Flow Graph

1. Node
2. Edge
3. Decision Node
4. Junction node

**Node**

Nodes in the control flow graph are used to create a path of procedures. Basically, it represents the sequence of procedures which procedure is next to come so, the tester can determine the sequence of occurrence of procedures.

We can see below in example the first node represent the start procedure and the next procedure is to assign the value of n after assigning the value there is decision node to decide next node of procedure as per the value of n if it is 18 or more than 18 so Eligible procedure will execute otherwise if it is less than 18 Not Eligible procedure executes. The next node is the junction node, and the last node is stop node to stop the procedure.

## Edge

Edge in control flow graph is used to link the direction of nodes.

We can see below in example all arrows are used to link the nodes in an appropriate direction.

## Decision node

Decision node in the control flow graph is used to decide next node of procedure as per the value.

We can see below in example decision node decide next node of procedure as per the value of n if it is 18 or more than 18 so Eligible procedure will execute otherwise if it is less than 18, Not Eligible procedure executes.

## Junction node

Junction node in control flow graph is the point where at least three links meet.

### Example

1. **public** **class** VoteEligiblityAge{
3. **public** **static** **void** main(String []args){
4. **int** n=45;
5. **if**(n>=18)
6. {
7. System.out.println("You are eligible for voting");
8. }  **else**
9. {
10. System.out.println("You are not eligible for voting");
11. }
12. }
13. }

**Diagram - control flow graph**

Control Flow Testing in white box testing Link

The above example shows eligibility criteria of age for voting where if age is 18 or more than 18 so print message "You are eligible for voting" if it is less than 18 then print "You are not eligible for voting."

Program for this scenario is written above, and the control flow graph is designed for the testing purpose.

In the control flow graph, start, age, eligible, not eligible and stop are the nodes, n>=18 is a decision node to decide which part (if or else) will execute as per the given value. Connectivity of the eligible node and not eligible node is there on the stop node.

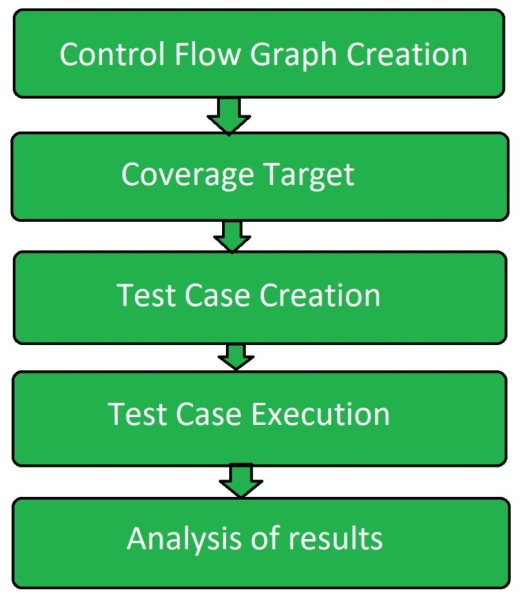
Test cases are designed through the flow graph of the programs to determine the execution path is correct or not. All nodes, junction, edges, and decision are the essential parts to design test cases.

***Control Flow Graph is a graphical representation of control flow or computation that is done during the execution of the program.*** Control flow graphs are mostly used in static analysis as well as compiler applications, as they can accurately represent the flow inside of a program unit. Control flow graph was originally developed by Frances E. Allen.

**Control flow testing** is a type of [software testing](https://www.geeksforgeeks.org/software-testing-basics/) that uses program’s control flow as a model. Control flow testing is a structural testing strategy. This testing technique comes under white box testing. For the type of control flow testing, all the structure, design, code and implementation of the software should be known to the testing team.

This type of testing method is often used by developers to test their own code and own implementation as the design, code and the implementation is better known to the developers. This testing method is implemented with the intention to test the logic of the code so that the user requirements can be fulfilled. Its main application is to relate the small programs and segments of the larger programs.

**Control Flow Testing Process:**  
Following are the steps involved into the process of control flow testing:



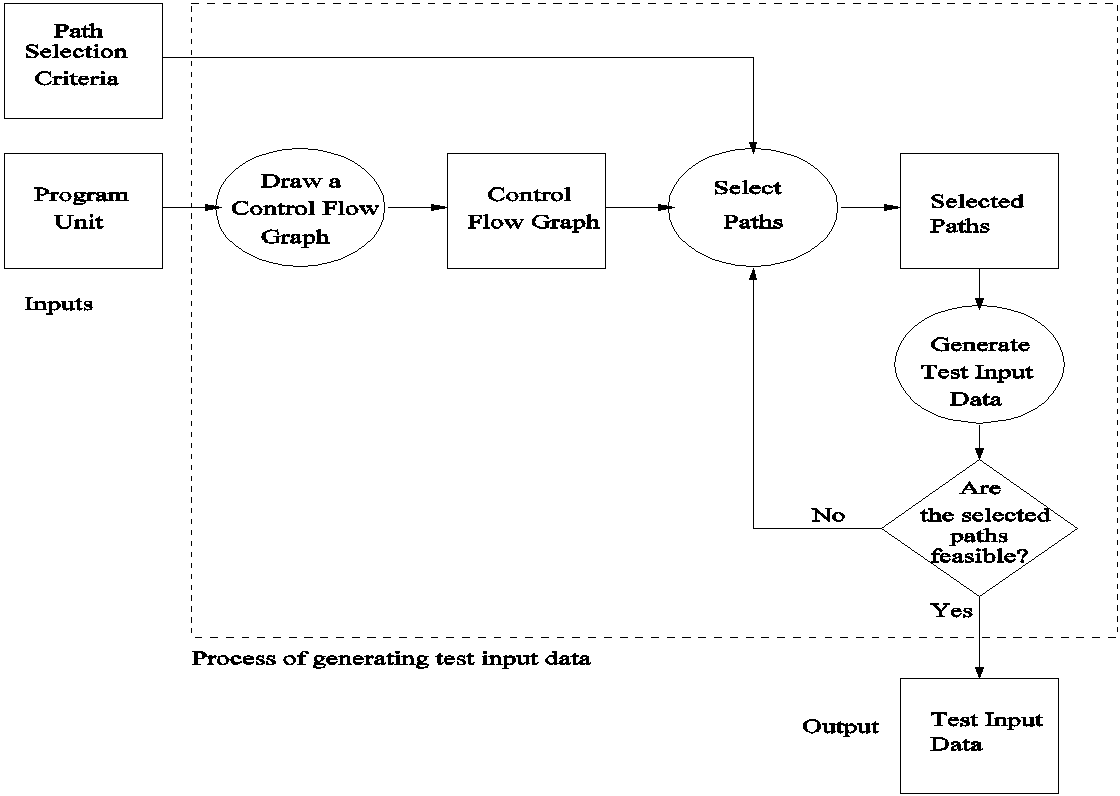
* **Control Flow Graph Creation:**  
  From the given source code a control flow graph is created either manually or by using the software.
* **Coverage Target:**  
  A coverage target is defined over the control flow graph that includes nodes, edges, paths, branches etc.
* **Test Case Creation:**  
  Test cases are created using control flow graphs to cover the defined coverage target.
* **Test Case Execution:**  
  After the creation of test cases over coverage target, further test cases are executed.
* **Analysis:**  
  Analyze the result and find out whether the program is error free or has some defects.

**Advantages of Control flow testing:**

* It detects almost half of the defects that are determined during the unit testing.
* It also determines almost one-third of the defects of the whole program.
* It can be performed manually or automated as the control flow graph that is used can be made by hand or by using software also.

**Disadvantages of Control flow testing:**

* It is difficult to find missing paths if program and the model are done by same person.
* Unlikely to find spurious (False or Bogus ) features.
* This stage identifies and highlights loops with multiple exit or entry points and unreachable code. Unreachable code is code that is surrounded by unconditional goto statements or that is in a branch of a conditional statement where the guarding condition can never be true.
* Control flow analysis is a static code analysis technique for determining the control flow of a program.
* The control flow is expressed as a control flow graph (CFG).
* The term "control flow analysis" was introduced independently by Neil D. Jones and Olin Shivers.
* For both functional programming languages and object-oriented programming languages, the term CFA, and elaborations such as k-CFA, refer to specific algorithms that compute control flow.
* For many imperative programming languages, the control flow of a program is explicit in a program's source code.
* As a result, control-flow analysis implicitly usually refers to a static analysis technique for determining the receiver(s) of function or method calls in computer programs written in a higher-order programming language.
* For example, in a programming language with higher-order functions like Scheme, the target of a function call may not be explicit: in the isolated expression
* **Two kinds of basic program statements:**
* Assignment statements (Ex. x = 2\*y; )
* Conditional statements (Ex. if(), for(), while(), …)
* **Control flow**
* Successive execution of program statements is viewed as flow of control.
* Conditional statements alter the default flow.
* **Program path**
* A program path is a sequence of statements from entry to exit.
* There can be a large number of paths in a program.
* There is an (input, expected output) pair for each path.
* Executing a path requires invoking the program unit with the right test input.
* Paths are chosen by using the concepts of path selection criteria.
* Tools: Automatically generate test inputs from program paths.



**Fig . The process of generating test input data for control flow testing**

* **3.2.3 Static Analysis by Tools (Automated Static Analysis) :-**

Static analysis tools refer to a wide array of tools that examine source code, executable, or even documentation, to find problems before they happen; without actually running the code. These tools vary greatly in scope and purpose, ranging from compiler-level checks for logical errors, to code styling enforcement, to cloud-based suites of tools that cover everything from documentation formatting to code complexity analysis.

Put simply, you could think of static analysis tools as anything that helps you maintain a healthy code base without having to actually run that code.

Static analysis tools can help us reduce and avoid these issues all-together by making it easier to find & fix these issues before they have significant impact on our projects.

This type of analysis addresses weaknesses in source code that might lead to vulnerabilities. Of course, this may also be achieved through manual code reviews.

**Static code Analysis Tools :-**

* 1. **AdaControl :-**

A tool to control occurrences of various entities or programming patterns in Ada code, used for checking coding standards, enforcement of safety related rules, and support for various manual inspections. Features automatic fixing of violations.

* 1. **Apache Yetus :-**

A collection of build and release tools. Included is the 'precommit' module that is used to execute full and partial/patch CI builds that provides static analysis of code via other tools as part of a configurable report. Built-in support may be extended with plug-ins.It is supported by Java & Python

* 1. **Astree :-**

Finds all potential runtime errors and data races by abstract interpretation, can prove their absence, and can prove functional assertions; tailored towards safety-critical C code

* 1. **Code Peer**

An advanced static analysis tool that detects potential run-time logic errors in Ada programs.

* 1. **Code Scene**:-

Behavioral analysis of code. Helps identify, prioritize, and manage technical debt. Measures organizational aspects of developer teams. Automated pull request integrations. It is supported by various programming languages as C,C++ ,C#, Java