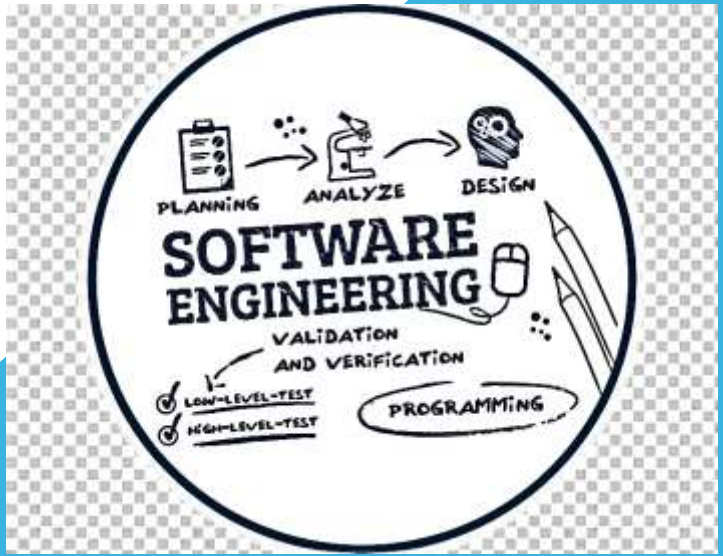
CHAPTER 4


PLANNING
ANALYZE
DESIGN
SOFTWARE ENGINEERING
VALIDATION AND VERIFICATION
LOW-LEVEL-TEST
HIGH-LEVEL-TEST
PROGRAMMING

CONTENTS PART-I

A strategic approach to software testing

Testing Strategies

Black Box and White Box Testing

Validation Testing

System Testing

Art of Debugging

SOFTWARE TESTING

Software Testing is a method to check whether the actual software product matches expected requirements and to ensure that software product is defect free.

It involves execution of software/system components using manual or automated tools to evaluate one or more properties of interest.

A STRATEGIC APPROACH TO TESTING

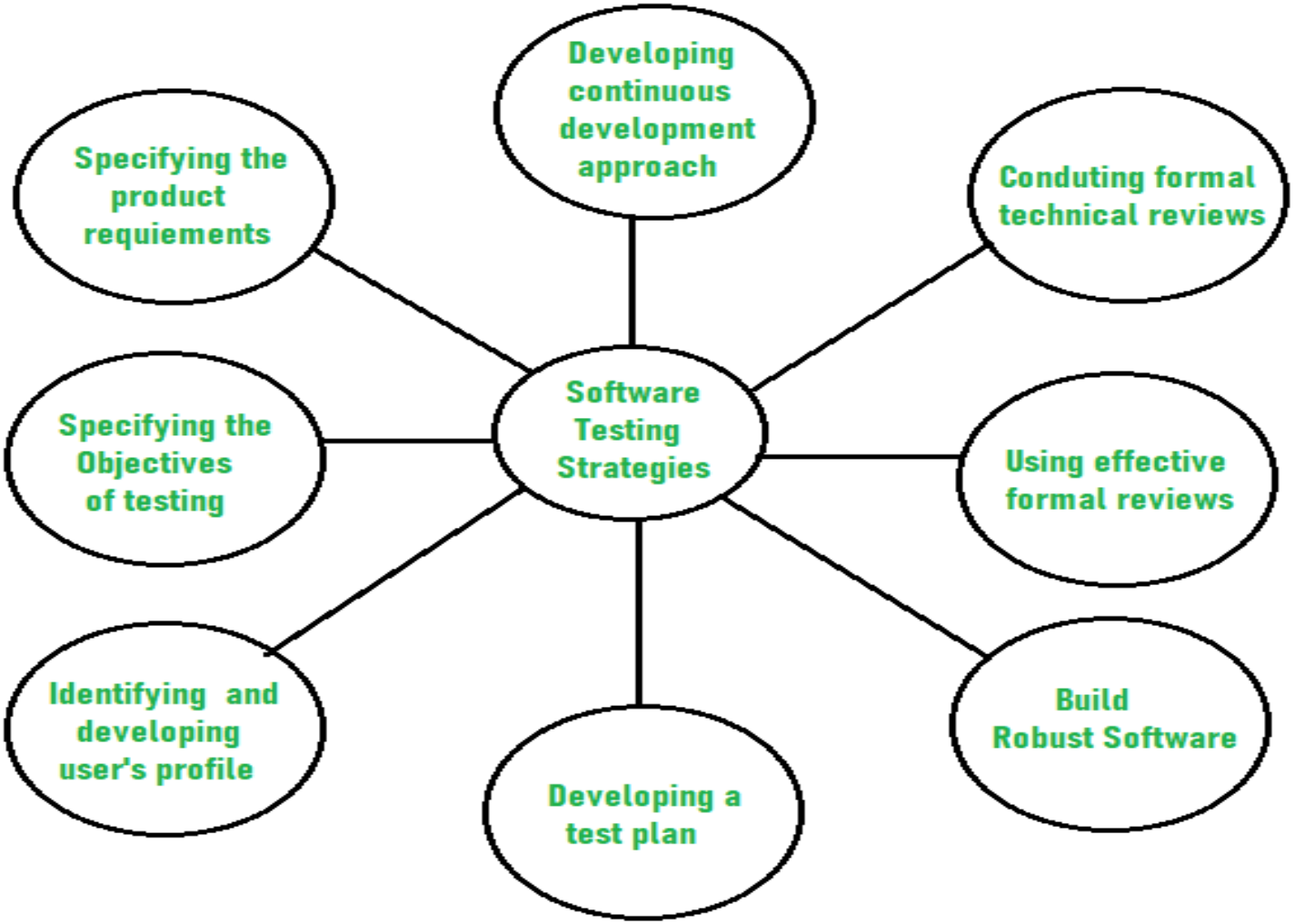
According to Glenn Myers, The objectives are :

The process of investigating and checking a program to find whether there is an error or not and does it fulfill the requirements or not is called testing.

When the number of errors found during the testing is high, it indicates that the testing was good and is a sign of good test case.

Finding an unknown error that's wasn't discovered yet is a sign of a successful and a good test case.

VARIOUS TESTING STRATEGIES



CHARACTERISTICS OF TESTING

The developer should conduct the successful technical reviews.

Testing starts with the component level and work from outside toward the integration.

Different testing techniques are suitable at different point in time.

Testing is organized by the developer of the software and by an independent test group.

Debugging and testing are different activities, then also the debugging should be accommodated in any strategy of testing.

SPIRAL TESTING STRATEGY

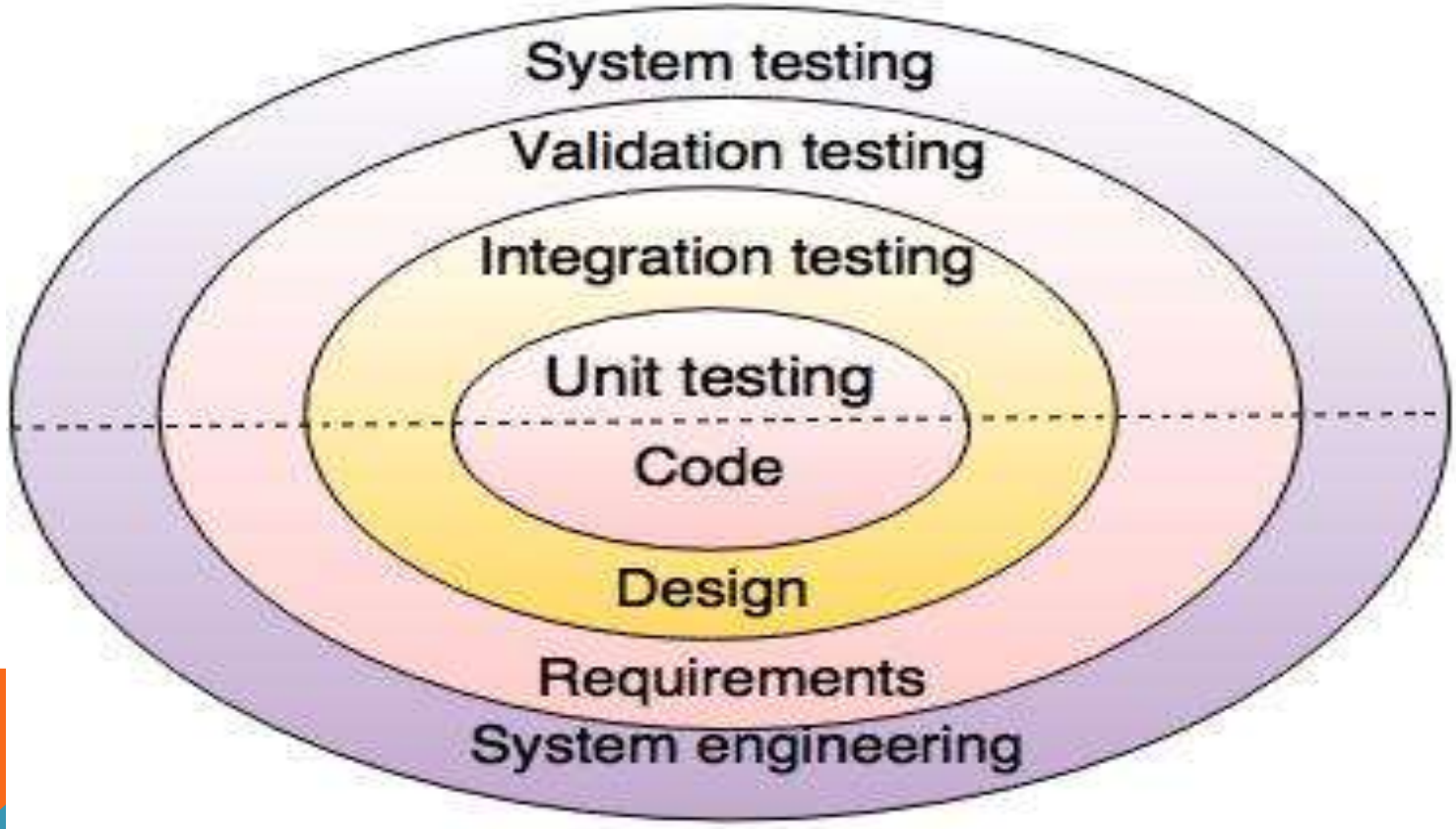
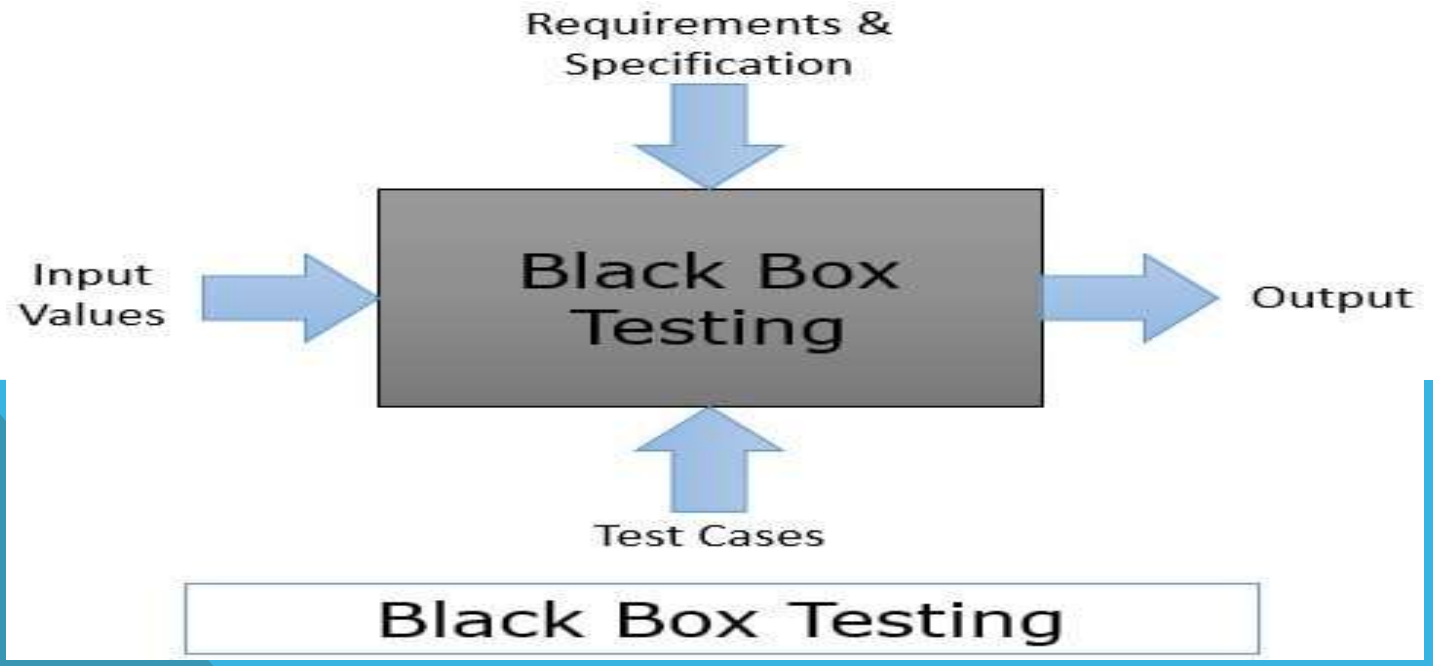


Fig. - Testing Strategy

BLACK BOX TESTING

A type of Software Testing where the functionality of software is not known.

The Testing is done without internal knowledge of products.



SYNTAX DRIVEN TESTING

1. Syntax Driven : This type of testing is applied to systems that can be syntactically represented by some language.

Example: Compilers, Language that can be represented by context free grammar

Each grammar rule is used once.

EQUIVALENCE PARTITIONING

The idea is to partition the input domain of the system into a number of equivalence classes such that each member of class works in a similar way, i.e.,

If a test case in one class results in some error, other members of class would also result into same error.

Has Two Steps: Identification of equivalence classes and generating test cases. For each class.

BOUNDARY VALUE ANALYSIS

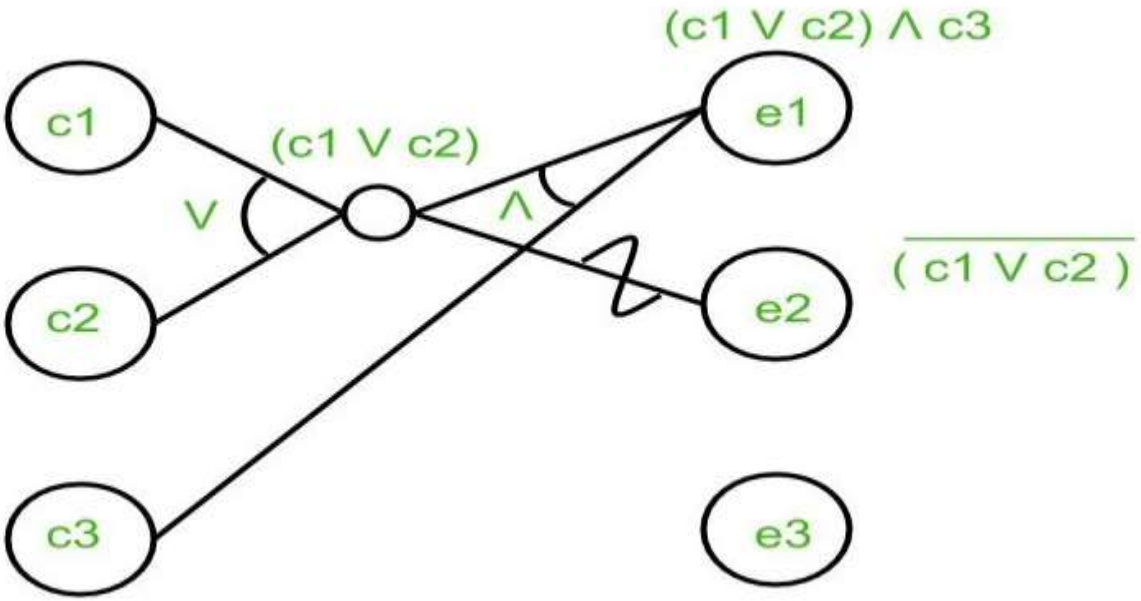
Boundaries are very good places for errors to occur. Hence if test cases are designed for boundary values of input domain then the efficiency of testing improves and probability of finding errors also increase.

For example – If valid range is 10 to 100 then test for 10,100 also apart from valid and invalid inputs.

CAUSE EFFECT GRAPHING

This technique establishes relationship between logical input called causes with corresponding actions called effect.

The causes and effects are represented using Boolean graphs.



REQUIREMENT BASED TESTING

This Testing Includes Validating the requirements given in the Software Requirement Specification Document for a particular software system.

Testing must be carried out in a timely manner.

Testing process should add value to the software life cycle, hence it needs to be effective.

Testing must provide the overall status of the project, hence it should be manageable.

COMPATIBILITY TESTING

The test case result not only depend on product but also infrastructure for delivering functionality. When the infrastructure parameters are changed it is still expected to work properly.

Some of them include processor, architecture, back-end components and OS.

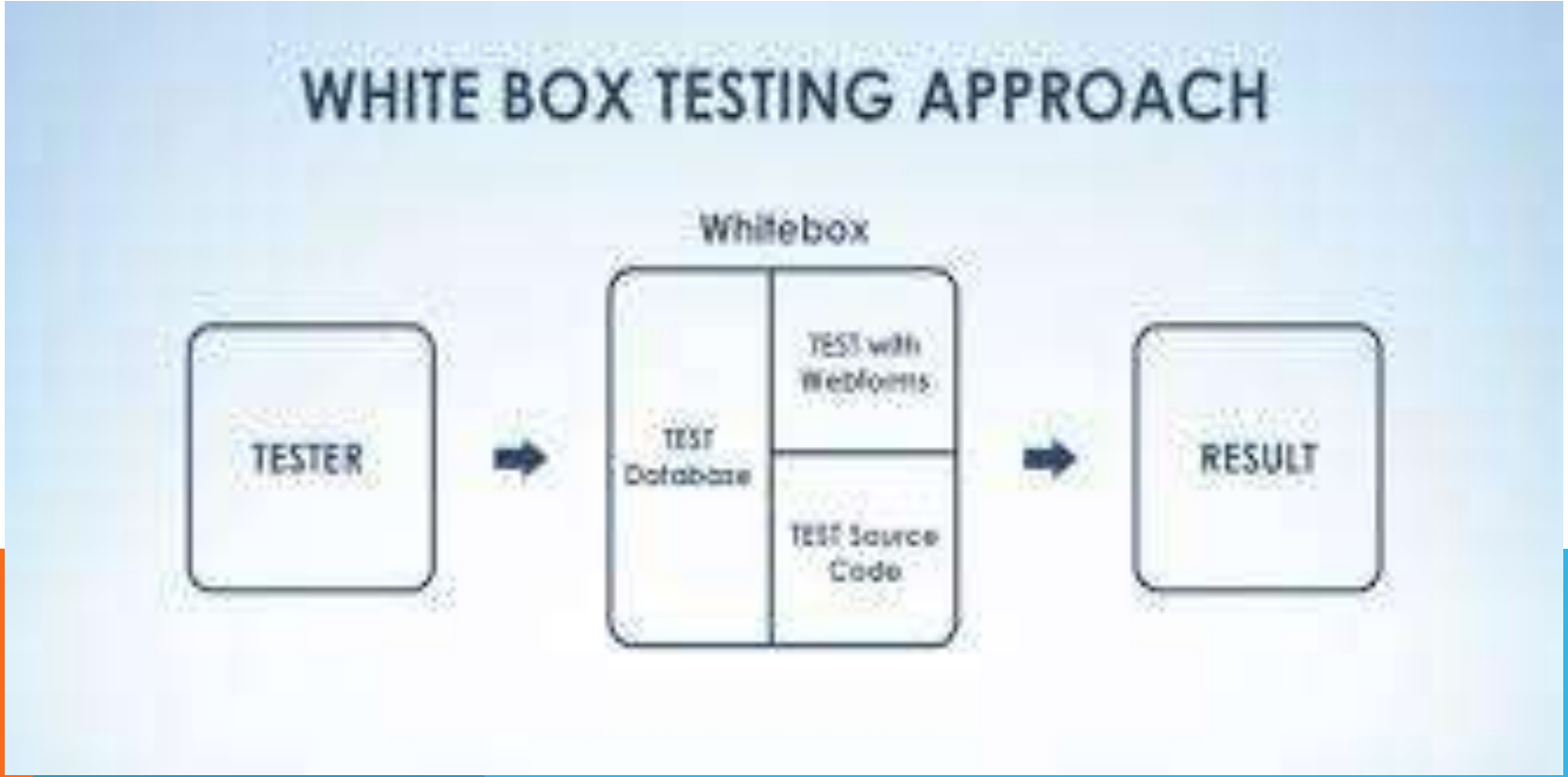
WHITE BOX TESTING

White box testing techniques analyze the internal structures the used data structures, internal design, code structure and the working of the software rather than just the functionality as in black box testing.

It is also called glass box testing or clear box testing or structural testing.

The Process Steps Include: Input, Processing , Proper Test Planning and Output.

WHITE BOX TESTING



WHITE BOX TESTING TECHNIQUES

Statement Coverage: In this technique, the aim is to traverse all statement at least once. Hence, each line of code is tested. In case of a flowchart, every node must be traversed at least once

Branch Coverage In this technique, test cases are designed so that each branch from all decision points are traversed at least once.

Condition Coverage: In this technique, all individual conditions must be covered.

WHITE BOX TESTING TECHNIQUES

Multiple Condition Coverage: In this technique, all the possible combinations of the possible outcomes of conditions are tested at least once.

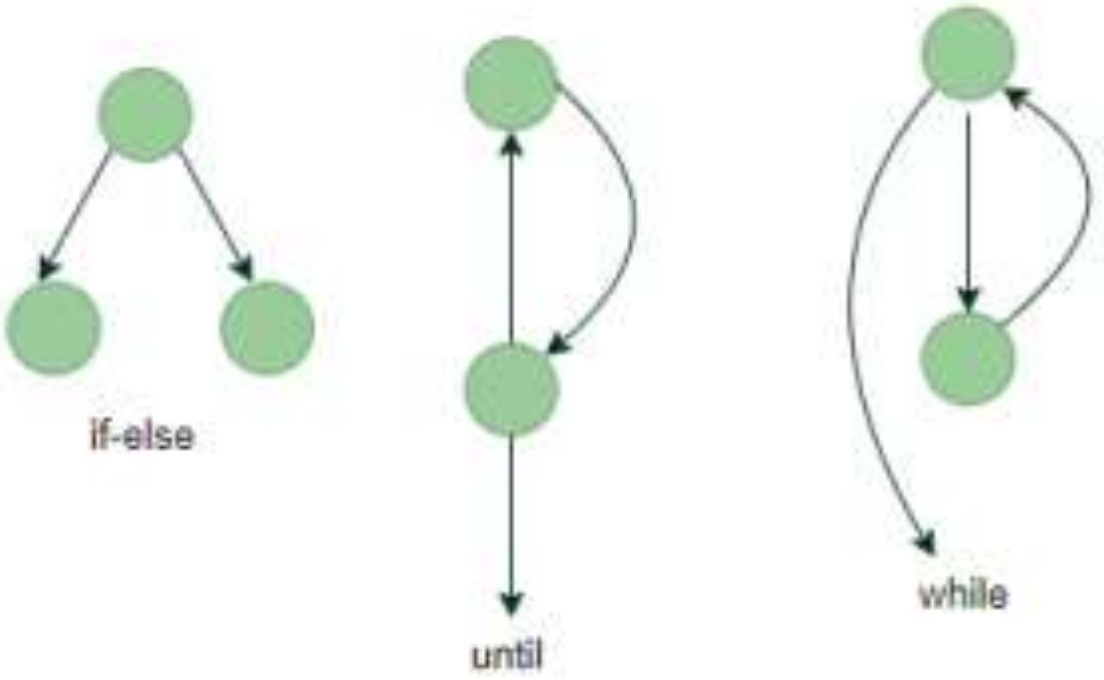
Basis Path Testing: In this technique, control flow graphs are made from code or flowchart and then Cyclomatic complexity is calculated which defines the number of independent paths so that the minimal number of test cases can be designed for each independent path.

WHITE BOX TESTING TECHNIQUES

Flow graph notation: It is a directed graph consisting of nodes and edges. Each node represents a sequence of statements, or a decision point.

A predicate node is the one that represents a decision point that contains a condition after which the graph splits.

Regions are bounded by nodes and edges.



ADVANTAGES OF WHITE BOX TESTING

White box testing is very thorough as the entire code and structures are tested.

It results in the optimization of code removing error and helps in removing extra lines of code.

It can start at an earlier stage as it doesn't require any interface as in case of black box testing.

Easy to automate.

DISADVANTAGES OF WHITE BOX TESTING

Main disadvantage is that it is very expensive.

Redesign of code and rewriting code needs test cases to be written again.

Testers are required to have in-depth knowledge of the code and programming language as opposed to black box testing.

Missing functionalities cannot be detected as the code that exists is tested.

Very complex and at times not realistic.

DIFFERENCES BETWEEN WHITE BOX AND BLACK BOX

Black Box Testing	White Box Testing
1. Black box testing techniques are also called functional testing techniques.	1. White box testing techniques are also called structural testing techniques.
2Black Box Testing is a software testing method in which the internal structure/ design/ implementation of the item being tested is NOT known to the tester	2. White Box Testing is a software testing method in which the internal structure/ design/ implementation of the item being tested is known to the tester.
3. It is mainly applicable to higher levels of testing such as Acceptance Testing and System Testing	3. Mainly applicable to lower levels of testing such as Unit Testing and Integration Testing
4. Black box testing is generally done by Software Testers	4. White box testing is generally done by Software Developers
5. Programming knowledge is not required	5. Programming knowledge is required
6. Implementation knowledge is not required.	6. Implementation knowledge is required

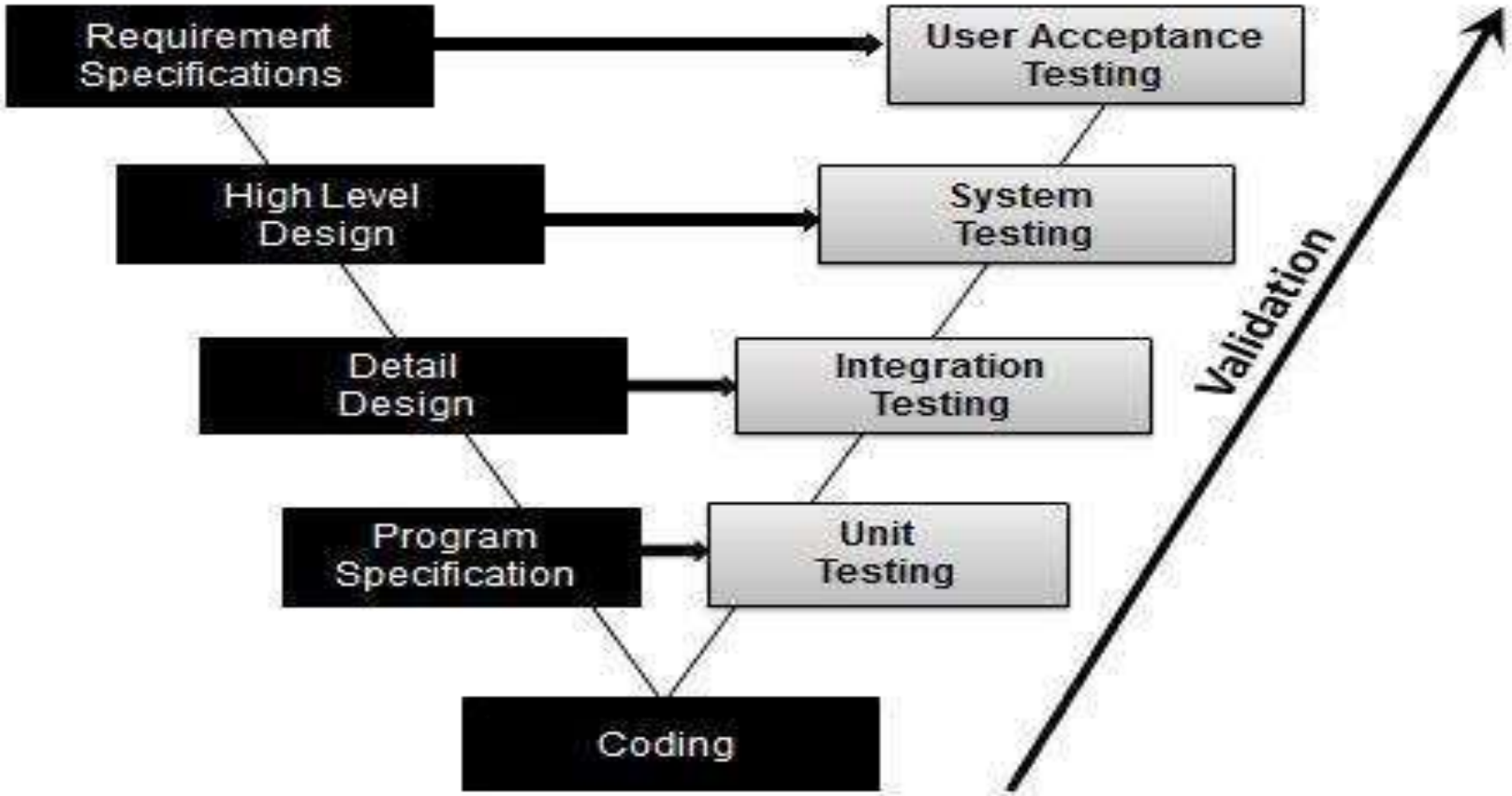
VALIDATION TESTING

The process of evaluating software during the development process or at the end of the development process to determine whether it satisfies specified business requirements.

It answers to the question, Are we building the right product?

Validation Testing ensures that the product actually meets the client's needs.

VALIDATION TESTING V-MODEL



VALIDATION TESTING TYPES

Unit Testing

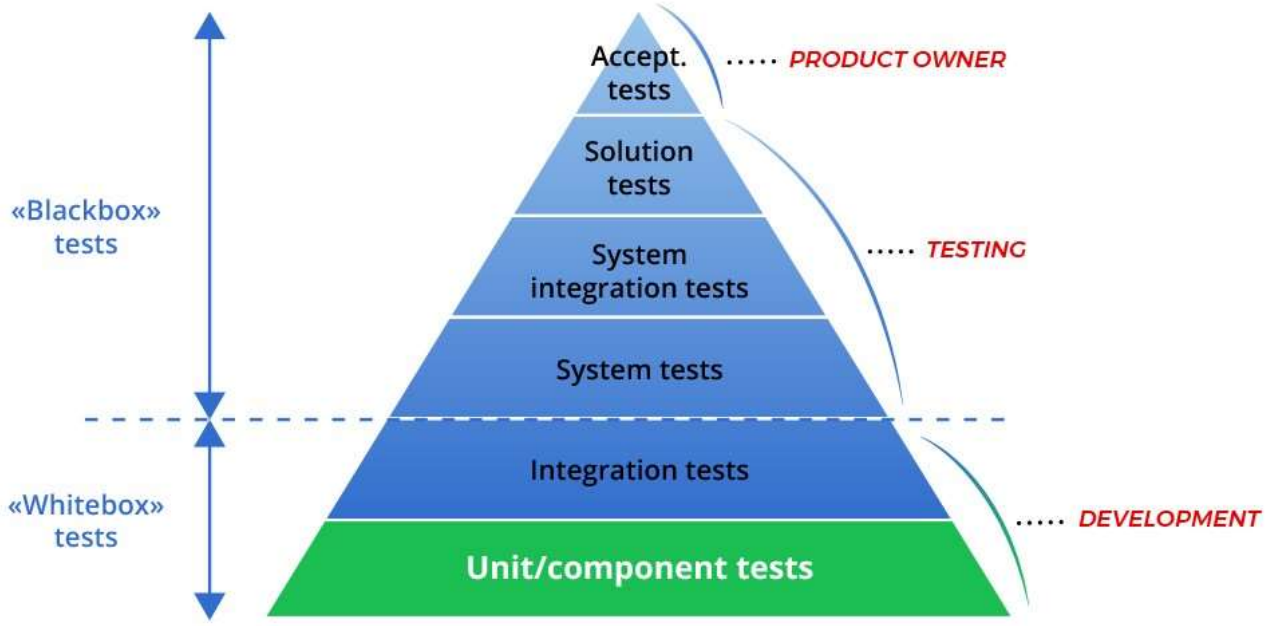
Integration Testing

System Testing

User Accepting Testing

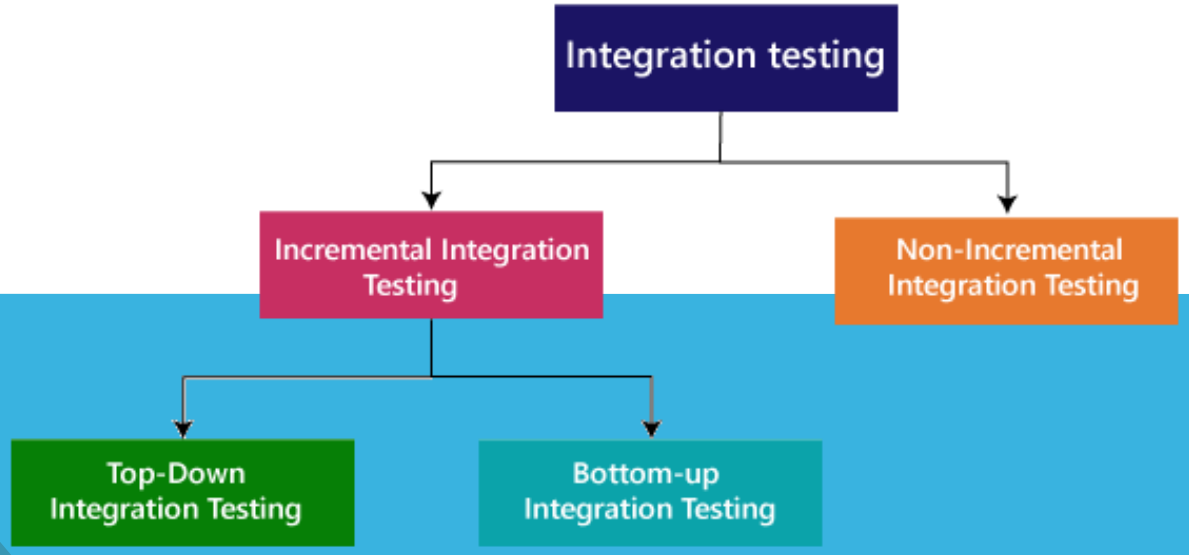
UNIT TESTING

It is an important type of validation testing. The point of the unit testing is to search for bugs in the product segment. Simultaneously, it additionally confirms crafted by modules and articles which can be tried independently.



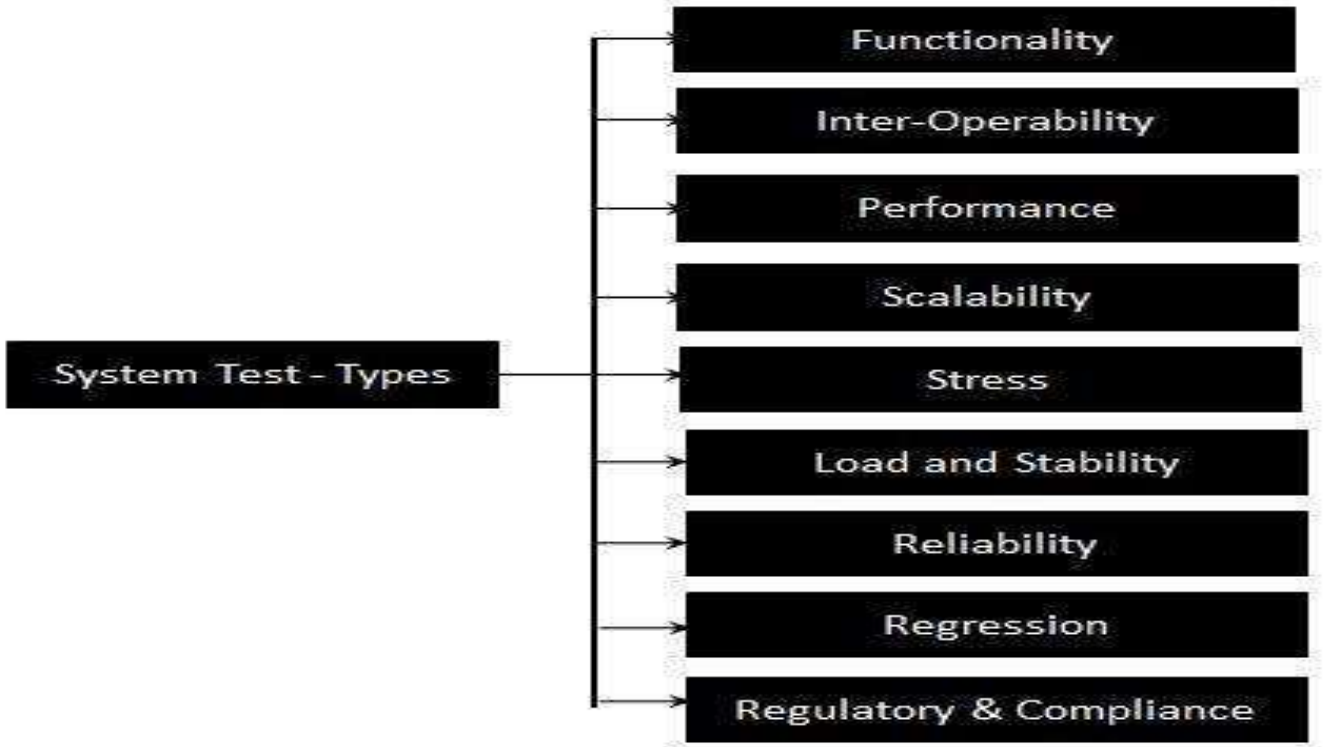
INTEGRATION TESTING

This is a significant piece of the validation model wherein the interaction between, where the association between the various interfaces of the pertaining component is tried.



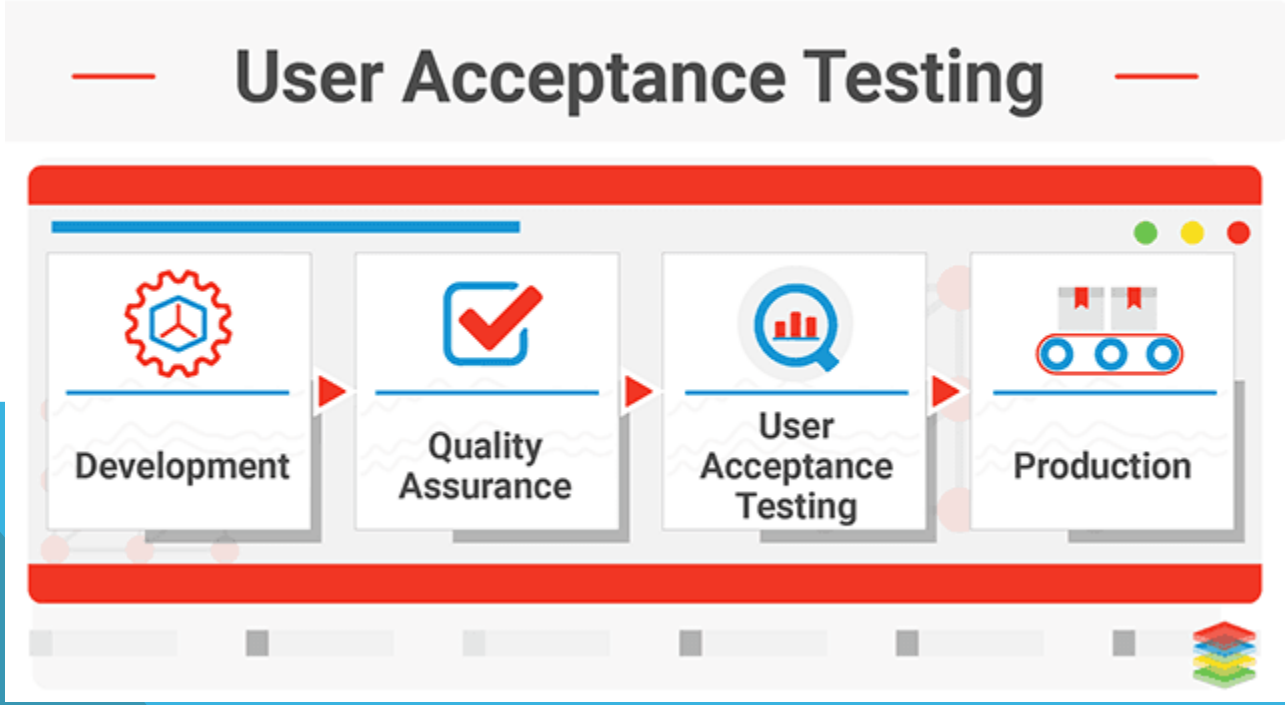
SYSTEM TESTING

System testing is done when the whole programming framework is prepared. The principal worry of framework testing is to confirm the framework against the predefined necessities.



USER ACCEPTANCE TESTING

During this testing, the tester actually needs to think like the customer and test the product concerning client needs, prerequisites, business forms and decide if the product can be given over to the customer or not.



DEBUGGING

In the context of software engineering, debugging is the process of fixing a bug in the software. In other words, it refers to identifying, analyzing and removing errors.

This activity begins after the software fails to execute properly and concludes by solving the problem and successfully testing the software

. It is considered to be an extremely complex and tedious task because errors need to be resolved at all stages of debugging.

DEBUGGING PROCESS

Problem identification and report preparation.

Assigning the report to software engineer to the defect to verify that it is genuine.

Defect Analysis using modeling, documentations, finding and testing candidate flaws, etc.

Defect Resolution by making required changes to the system.

Validation of corrections.

DEBUGGING TOOLS

Debugging tool is a computer program that is used to test and debug other programs. A lot of public domain software like gdb and dbx are available for debugging.

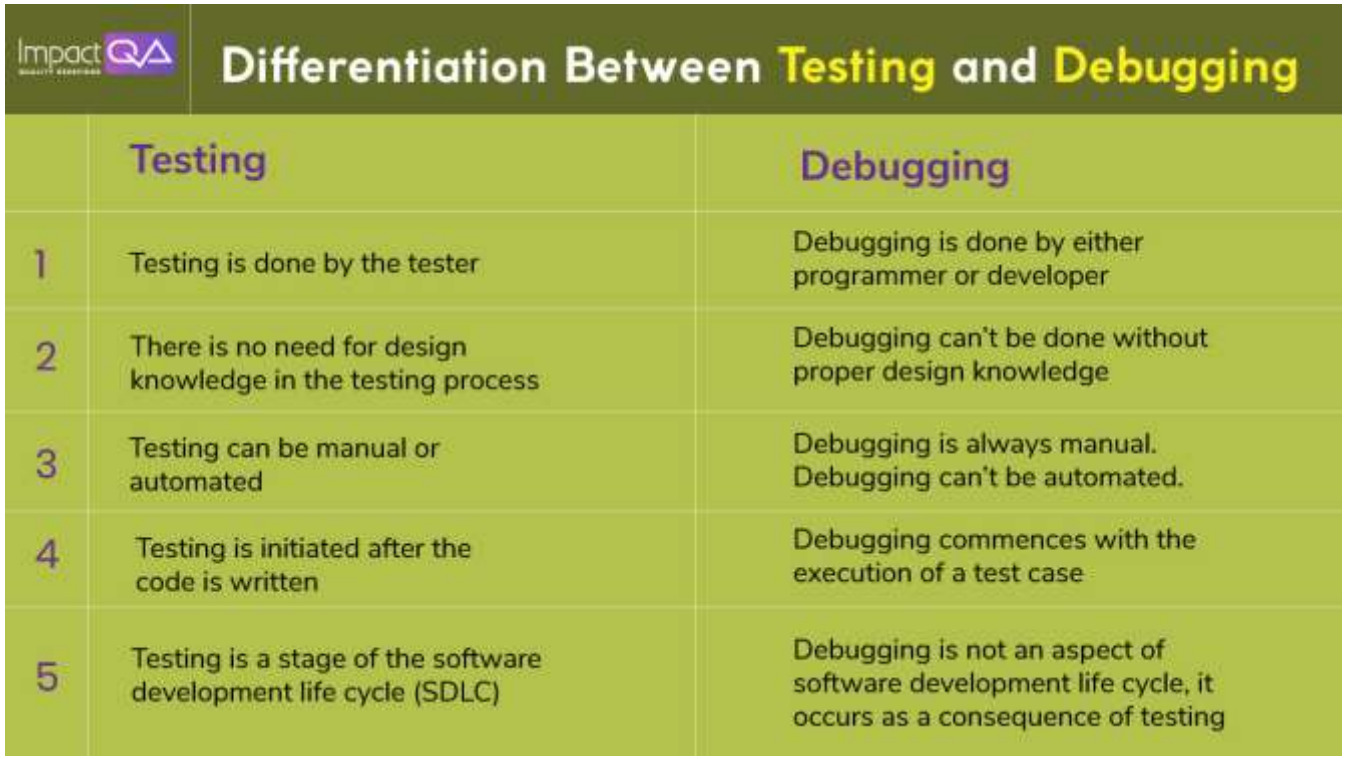
Some of the widely used debuggers are:

Radare2

Win Dbg

Valgrind

DEBUGGING VS TESTING



	Testing	Debugging
1	Testing is done by the tester	Debugging is done by either programmer or developer
2	There is no need for design knowledge in the testing process	Debugging can't be done without proper design knowledge
3	Testing can be manual or automated	Debugging is always manual. Debugging can't be automated.
4	Testing is initiated after the code is written	Debugging commences with the execution of a test case
5	Testing is a stage of the software development life cycle (SDLC)	Debugging is not an aspect of software development life cycle, it occurs as a consequence of testing

SOFTWARE ENGINEERING

PART-II

CONTENTS

Software Quality

Metrics for Analysis Model

Metrics for Design Model

Metrics for Source Code

Metrics for Testing

Metrics for Maintenance.

SOFTWARE QUALITY

Software quality is defined as a field of study and practice that describes the desirable attributes of software products. There are two main approaches to software quality: defect management and quality attributes.

Software Quality refers to both functional quality and structural quality.

SOFTWARE FUNCTIONAL QUALITY

It reflects how well it satisfies a given design, based on the functional requirements or specifications.

SFQ is pertaining to conformance to the functional requirements.

The SFQ is measured by the level of end user satisfaction.

SOFTWARE STRUCTURAL QUALITY

It deals with the handling of non-functional requirements that support the delivery of the functional requirements, such as robustness or maintainability, and the degree to which the software was produced correctly

Attributes are : Code testability, Maintainability, understandability, efficiency , security.

SOFTWARE QUALITY ASSURANCE

Is simply a way to assure quality in the software. It is the set of activities which ensure processes, procedures as well as standards suitable for the project and implemented correctly.

Software Quality Assurance is a process which works parallel to development of a software

Software Quality Assurance is a kind of an Umbrella activity that is applied throughout the software process.

SOFTWARE QUALITY CONTROL

Software Quality Control (SQC) is a set of activities to ensure the quality in software products.

These activities focus on determining the defects in the actual products produced.

It involves product-focused action.

Software Quality Control is commonly referred to as Testing.

SOFTWARE QUALITY CHALLENGE

In the software industry, the developers will never declare that the software is free of defects, unlike other industrial product manufacturers usually do.

The Key Reasons are :

- → Product Complexity
- → Product Visibility
- → Product Development and Production Process.

METRICS FOR ANALYSIS MODEL

Technical work in software engineering begins with the creation of the analysis model. It is at this stage that requirements are derived and that a foundation for design is established.

Therefore, technical metrics that provide insight into the quality of the analysis model are desirable.

These Metrics are used to analyze the analysis model with the objective of increased coding, integration and testing effort.

Ex: Function Point(FP) and Lines of Code(LOC)

METRICS FOR DESIGN MODEL

The success of a software project depends largely on the quality and effectiveness of the software design.

Hence, it is important to develop software metrics from which meaningful indicators can be derived.

Various design metrics such as architectural design metrics, component-level design metrics, user-interface design metrics, and metrics for object-oriented design are used to indicate the complexity, quality, and so on of the software design.

METRICS FOR SOURCE CODE

Halstead proposed the first analytic laws for Computer science by using a set of primitive measures, which can be derived once the design phase is complete and code is generated. These measures are listed below.

n_1 = number of distinct operators in a program

n_2 = number of distinct operands in a program

N_1 = total number of operators

N_2= total number of operands.

The Halstead Equation denotes the Coding metric for software quality.

$N = n_1 \log_2 n_1 + n_2 \log_2 n_2$. [Program Length]

$V = N \log_2 (n_1+n_2)$. [Program Volume]

METRICS FOR TESTING

Majority of the metrics used for testing focus on testing process rather than the technical characteristics of test. Generally, testers use metrics for analysis, design, and coding to guide them in design and execution of test cases.

Halstead measures can be used to derive metrics for testing effort. By using program volume (V) and program level (PL),Halstead effort (e)can be calculated by the following equations.

$e = V / PL$

Percentage of testing effort (z) = $e(z)/\sum e(i)$

METRICS FOR MAINTENANCE

For the maintenance activities, metrics have been designed explicitly. IEEE have proposed Software Maturity Index (SMI), which provides indications relating to the stability of software product. For calculating SMI, following parameters are considered.

Number of modules in current release (M_T)

Number of modules that have been changed in the current release (F_e)

Number of modules that have been added in the current release (F_a)

Number of modules that have been deleted from the current release (F_d)

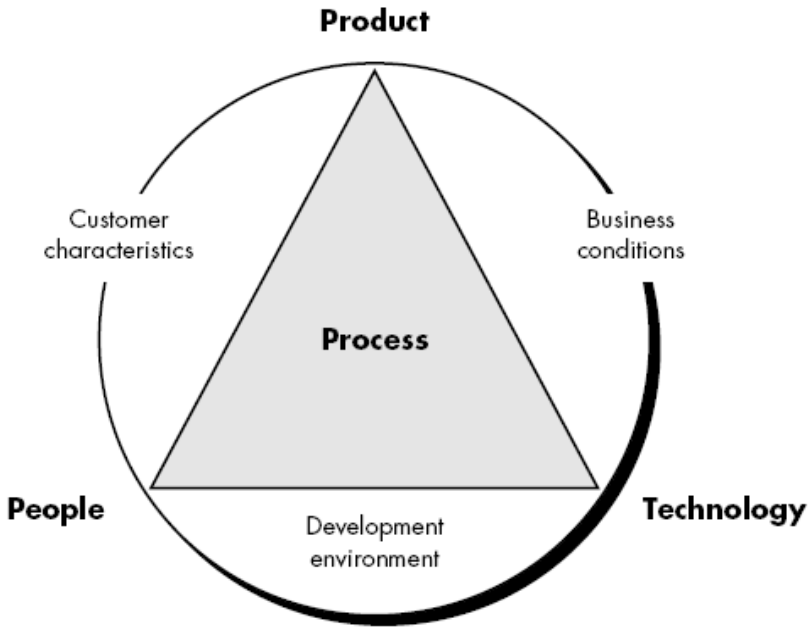
Once all the parameters are known, SMI can be calculated by using the following equation.

$SMI = [M_T – (F_a + F_e + F_d)]/M_T$.

PROCESS METRICS

To improve any process, it is necessary to measure its specified attributes, develop a set of meaningful metrics based on these attributes, and then use these metrics to obtain indicators in order to derive a strategy for process improvement.

Using software process metrics, software engineers are able to assess the efficiency of the software process that is performed using the process as a framework



PRODUCT METRICS

Product metrics are software product measures at any stage of their development, from requirements to established systems. Product metrics are related to software features only.

Metrics are of 2 types :

- Dynamic metrics that are collected by measurements made from a program in execution.
- Static metrics that are collected by measurements made from system representations such as design, programs, or documentation.

DYNAMIC PRODUCT METRICS

Dynamic metrics are usually quite closely related to software quality attributes. It is relatively easy to measure the execution time required for particular tasks and to estimate the time required to start the system. These are directly related to the efficiency of the system failures and the type of failure can be logged and directly related to the reliability of the software.

STATIC PRODUCT METRICS

Static metrics have an indirect relationship with quality attributes. A large number of these matrices have been proposed to try to derive and validate the relationship between the complexity, understandability, and maintainability.