Software Coding and Testing

Marwadi University

Computer Engineering Diploma

Unit 6:-
Software Coding and Testing Software Engineering
(09CE2402)

CODING

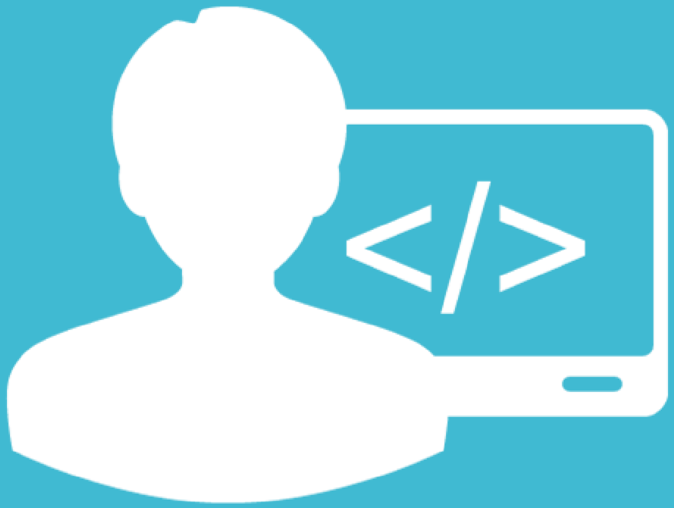
- The input to the coding phase is the design document.
- During coding phase:

  –modules identified in the design document are coded according to the module specifications.

Coding Standards

- Good software development organizations normally require their programmers

  - to adhere to some well-defined and standard style of coding called coding standards.



Coding Standards

- Most software development organizations formulate

  -their own coding standards that suit them most,

  -require their engineers to follow these standards strictly.

Coding Standards

- The purpose of requiring all engineers of an organization to adhere to a standard style of coding is the following:
  - it gives a uniform appearance to the codes written by different engineers,
  - it enhances code understanding,
  - encourages good programming practices.

Coding Standards

1. Limited use of globals
2. Standard headers for different modules
3. Naming conventions for local variables, global variables, constants and functions
4. Indentation
5. Error return values and exception handling conventions
6. Avoid using a coding style that is too difficult to understand
7. Code should be well documented
8. Length of functions should not be very large
9. Try not to use GOTO statement

Coding Standards

(1) Limited use of globals

These rules tell about which types of data that can be declared global and the data that can’t be.

CODING STANDARDS

(2) Standard headers for different modules

For better understanding and maintenance of the code, the header of different modules should follow some standard format and information.

The header format must contain below things that is being used in various companies:

1) Name of the module
2) Date of module creation
3) Author of the module
4) Modification history
5) Synopsis of the module about what the module does
6) Different functions supported in the module along with their input output parameters
7) Global variables accessed or modified by the module

Coding Standards

(2) Sample Header

```
/**
 * MyClass <br>
 *
 * This class is merely for illustrative purposes.
<br>
 *
 * Revision History:<br>
 * 1.1 – Added javadoc headers <br>
 * 1.0 - Original release<br>
 *
 * @author P.U. Jadeja
 * @version 1.1, 12/02/2018
 */
public class MyClass {
    . . .
}
```

Coding Standards

(3) Naming conventions for local variables, global variables, constants and functions

Some of the naming conventions are given below:

- Meaningful and understandable variables name helps anyone to understand the reason of using it.
- Local variables should be named using camel case lettering starting with small letter (e.g. localData) whereas Global variables names should start with a capital letter (e.g. GlobalData). Constant names should be formed using capital letters only (e.g. CONSDATA).
- It is better to avoid the use of digits in variable names.
- The names of the function should be written in camel case starting with small letters.
- The name of the function must describe the reason of using the function clearly and briefly.

Coding Standards

(4) Indentation

- Proper indentation is very important to increase the readability of the code.
- For making the code readable, programmers should use White spaces properly. Some of the spacing conventions are given below:

1) There must be a space after giving a comma between two function arguments.
2) Each nested block should be properly indented and spaced.
3) Proper Indentation should be there at the beginning and at the end of each block in the program.
4) All braces should start from a new line and the code following the end of braces also start from a new line.

Coding Standards

(5) Error return values and exception handling conventions

- All functions that encountering an error condition should either return a 0 or 1 for simplifying the debugging.

Coding Standards

(6) Avoid using a coding style that is too difficult to understand

- Code should be easily understandable.
- The complex code makes maintenance and debugging difficult and expensive.

Coding Standards

(7) Code should be well documented

- The code should be properly commented for understanding easily.
- Comments regarding the statements increase the understandability of the code.

Coding Standards

(8) Length of functions should not be very large

- Lengthy functions are very difficult to understand.
- That’s why functions should be small enough to carry out small work and lengthy functions should be broken into small ones for completing small tasks.

Coding Standards

(9) Try not to use GOTO statement

- GOTO statement makes the program unstructured, thus it reduces the understandability of the program and also debugging becomes difficult.

Coding Guidelines

Coding guidelines

The following are some representative coding guidelines

1. Coding guidelines increase the efficiency of the software and reduces the development time.
2. Coding guidelines help in detecting errors in the early phases, so it helps to reduce the extra cost incurred by the software project.
3. If coding guidelines are maintained properly, then the software code increases readability and understandability thus it reduces the complexity of the code.
4. It reduces the hidden cost for developing the software.

Software Faults

Many reasons

- Software systems with large number of states
- Complex formulas, activities, algorithms
- Customer is often unclear of needs
- Size of software
- Number of people involved

Types of Faults

Syntax	Wrong syntax; types Compiler
Computation/ Precision	Not enough accuracy
Documentation	Misleading documentation
Throughput/Performance	System performs below expectations
Recovery	System restarted from abnormal state
Hardware & related software	Compatibility issues

Software Quality

Software Quality remains an issue

Who is to blame?

Customers blame developers
Arguing that careless practices lead to low-quality software

Developers blame Customers & other stakeholders
Arguing that irrational delivery dates and continuous stream of changes force to deliver software before it has been fully validated

Who is Right? Both – and that’s the problem

Code Review

Code Review

- Code Review is carried out after the module is successfully compiled and all the syntax errors have been eliminated.
- Code Reviews are extremely cost-effective strategies for reduction in coding errors and to produce high quality code.

Two methods

1) Code Walk Through
2) Code Inspection

(1) Code Walk Through

- Code walk through is an informal code analysis technique.
- The main objectives of the walk through are to discover the algorithmic and logical errors in the code.
- A few members of the development team are given the code few days before the walk through meeting to read and understand code.

(1) Code Walk Through

- Each member selects some test cases and simulates execution of the code by hand
- The members note down their findings to discuss these in a walk through meeting where the coder of the module is present.

(2) Code Inspection

- The aim of Code Inspection is to discover some common types of errors caused due to improper programming.
- In other words, during Code Inspection the code is examined for the presence of certain kinds of errors.

(2) Code Inspection

- For instance, consider the classical error of writing a procedure that modifies a parameter while the calling routine calls that procedure with a constant actual parameter.
- It is more likely that such an error will be discovered by looking for these kinds of mistakes in the code.
- In addition, commitment to coding standards is also checked.

Few classical programming errors

- Use of uninitialized variables
- Jumps into loops
- Nonterminating loops
- Incompatible assignments
- Array indices out of bounds
- Improper storage allocation and deallocation
- Mismatches between actual and formal parameter in procedure calls
- Improper modification of loop variables

Software Documentation

Software Documentation

- When various kinds of software products are developed, various kinds of documents are also developed as part of any software engineering process e.g.
  - Users’ manual,
  - Software requirements specification (SRS) documents,
  - Design documents,
  - Test documents,
  - Installation manual

Software Documentation

- Different types of software documents can broadly be classified into the following:

1) Internal documentation
2) External documentation

(1) Internal Documentation

- It is the code perception features provided as part of the source code.
- It is provided through appropriate module headers and comments embedded in the source code.
- It is also provided through the useful variable names, module and function headers, code indentation, code structuring, use of enumerated types and constant identifiers, use of user-defined data types, etc.

(1) Internal Documentation

- Even when code is carefully commented, meaningful variable names are still more helpful in understanding a piece of code.
- Good organizations ensure good internal documentation by appropriately formulating their coding standards and guidelines.

(2) External Documentation

- It is provided through various types of supporting documents
  1) such as users' manual
  2) Software Requirements Specification (SRS) document
  3) Design Document
  4) Test Documents, etc.
- A systematic software development style ensures that all these documents are produced in an orderly fashion.

Difference Between Internal and External Documentation

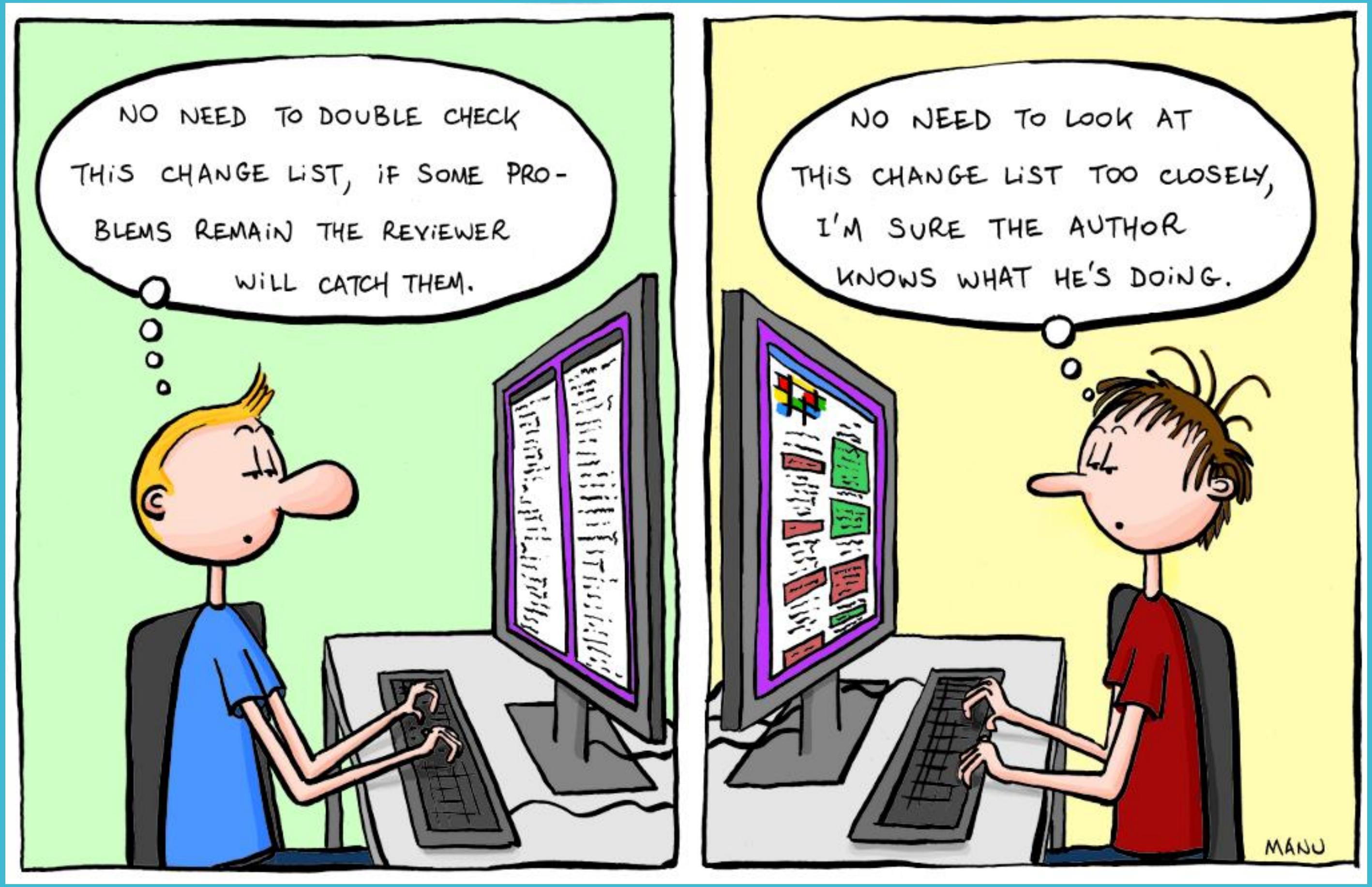
S.NO.	Internal documentation	External documentation
1.	Internal documentation is written in a program as comments.	External documentation is written in a place where people who need to use the software can read about how to use the software.
2.	Internal documentation would be comments and remarks made by the programmer in the form of line comments	External documentation would be things like flow charts, UML diagrams, requirements documents, design documents etc.
3.	Internal Documentation is created within the programming department and shows the design and implementation of the project (flow charts, UML diagrams, design documents, etc.).	External Documentation is created by the user and Programmer/System Analyst.

Software Testing

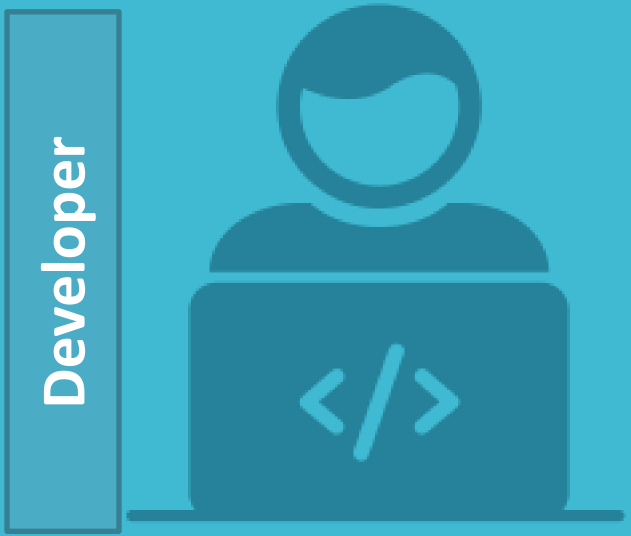
Software Testing

- **Testing** is the **process** of exercising a program with the specific **intent of finding errors** prior to delivery to the end user.

SOFTWARE TESTING CONT.


NO NEED TO DOUBLE CHECK THIS CHANGE LIST, IF SOME PRO-BLEMS REMAIN THE REVIEWER WILL CATCH THEM.
NO NEED TO LOOK AT THIS CHANGE LIST TOO CLOSELY, I'M SURE THE AUTHOR KNOWS WHAT HE'S DOING.
MANU

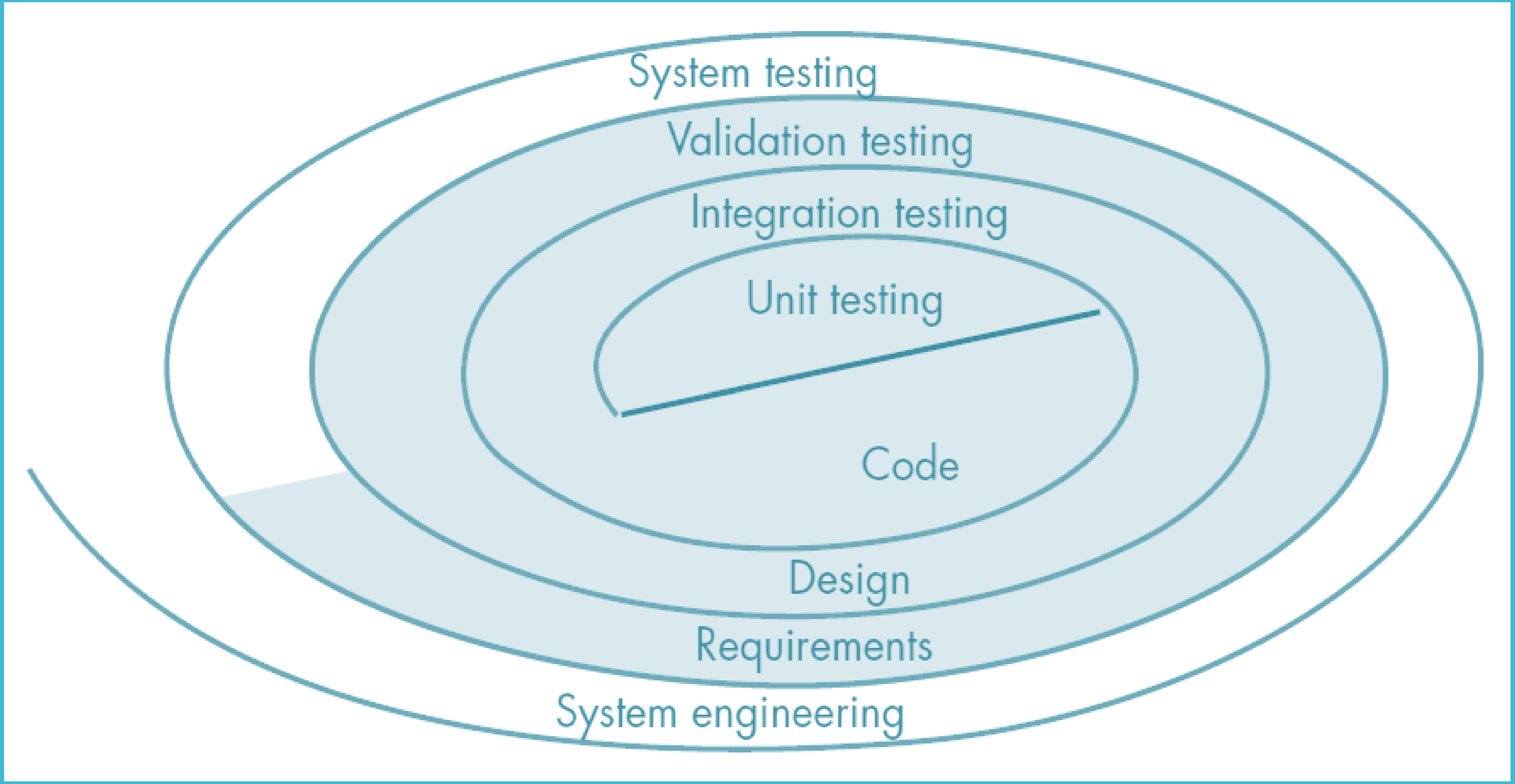
WHO TEST THE SOFTWARE



Understands the system but, will test "gently"
and, is driven by "delivery"

Must learn about the system, but, will attempt to break it
and, is driven by quality

Software Testing Strategy



Software Testing Strategy

Unit Testing

- It concentrate on each unit of the software as implemented in source code.
- It focuses on each component individual, ensuring that it functions properly as a unit.

Integration Testing

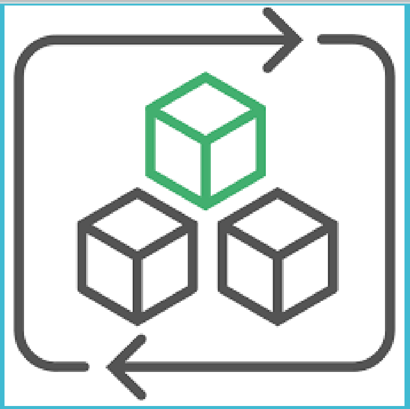
- It focus on design and construction of software architecture
- Integration testing is the process of testing the interface between two software units or modules

Software Testing Strategy

Validation Testing

- Software is **validated against requirements** established as a part of requirement modeling
- It give **assurance** that software meets all informational, functional and performance requirements

System Testing



- The **software** and **other software elements** are **tested as a whole**
- Software once validated, must be combined with other system elements e.g. hardware, people, database etc…
- It verifies that all elements mesh properly and that overall system function / performance is achieved.

VIEWS OF TEST OBJECTS

Black Box Testing

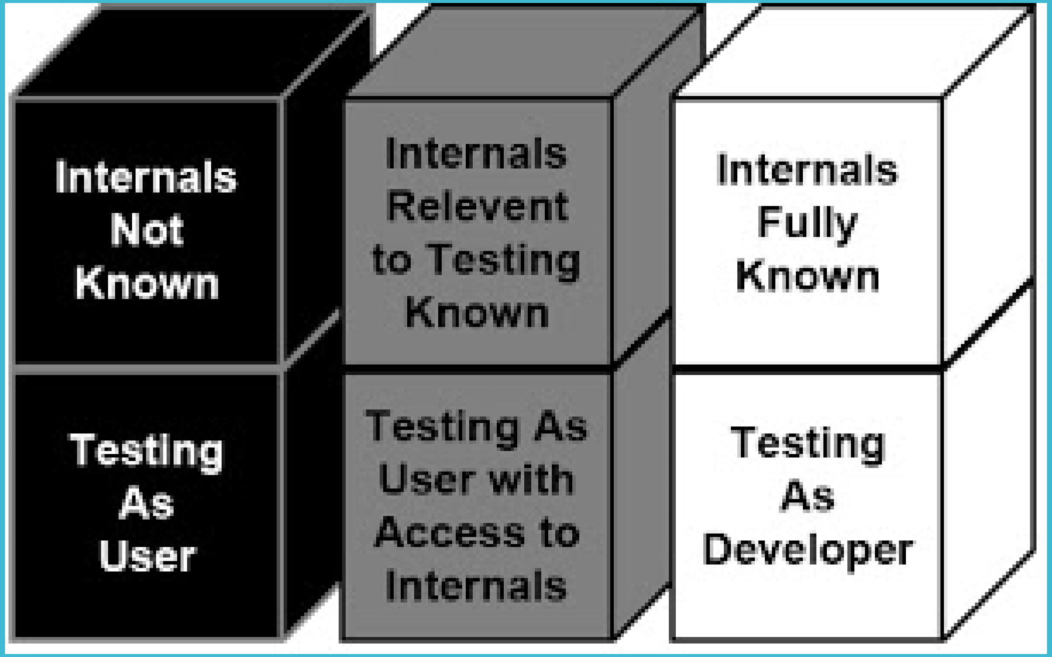
Close Box Testing
Testing based only on specification

Grey Box Testing

Partial knowledge of source code

White Box Testing

Open Box Testing
Testing based on actual source code


Internals Not Known
Testing As User
Internals Relevent to Testing Known
Testing As User with Access to Internals
Internals Fully Known
Testing As Developer

Black Box Testing

- Black Box Testing is testing technique having no knowledge of the internal functionality/structure of the system
- It also called Functional or Closed-Box Testing

Black Box Testing

- Black-box Testing focuses on testing the function of the program or application against its Specification
- Determines whether combinations of inputs and operations produce expected results
- When black box testing is applied to software engineering, the tester would only know the "legal" inputs and what the expected outputs should be, but not how the program actually arrives at those outputs

Black Box Testing

- Exhausting testing is not always possible when there is a large set of input combinations, because of budget and time constraint.
- The special techniques are needed which select test-cases smartly from the all combination of test-cases in such a way that all scenarios are covered.
- **Two techniques are used**
    - Equivalence Partitioning
    - Boundary Value Analysis (BVA)

Equivalence Partitioning

- It is a software test design technique that involves dividing input values into valid and invalid partitions and selecting representative values from each partition as test data.
- 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.

EQUIVALENCE PARTITIONING

Example - Equivalence Partitioning

- For example, a program which edits credit limits within a given range ($10,000 - $15,000) would have three equivalence classes
    - < $10,000 (invalid)
    - Between $10,000 and $15,000 (valid)
    - > $15,000 (invalid)

Boundary Value analysis

Boundary Value Analysis (BVA)

- It arises from the fact that most program fail at input boundaries
- This technique consists of developing test cases and data that focus on the input and output boundaries of a given function
- In Boundary Testing, Equivalence Class Partitioning plays a good role
- Boundary Testing comes after the Equivalence Class Partitioning

Boundary Value analysis

Boundary Value Analysis (BVA)

The basic idea in boundary value testing is to select input variable values at their:

Just below the minimum	**Minimum**	Just above the minimum
Just below the maximum	**Maximum**	Just above the maximum

Boundary Value analysis

Example – Boundary Value Analysis

- In the same credit limit example, boundary analysis would test.
- Low boundary -/+ one ($9,999 and $10,001)
- On the boundary ($10,000 and $15,000)
- Upper boundary -/+ one ($14,999 and $15,001)

Black Box Testing

❑ **Advantages**

- More effective on larger units of code
- Tester needs no knowledge of implementation, including specific programming languages
- Tester and programmer are independent of each other
- Tests are done from a user's point of view

Black Box Testing

- **Disadvantages**
  - Only a small number of possible inputs can actually be tested
  - Without clear and concise specifications, test cases are hard to design
  - There may be unnecessary repetition of test inputs if the tester is not informed of test cases the programmer has already tried
  - Cannot be directed toward specific segments of code which may be very complex (and therefore more error prone)

White Box Testing

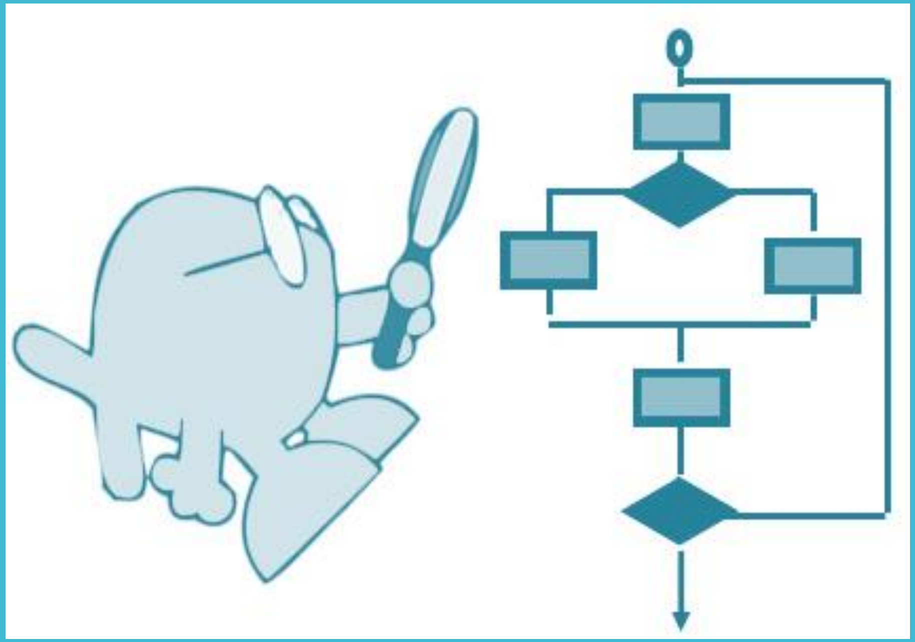
- Also known as structural testing
- White Box Testing is a software testing method in which the internal structure/design/implementation of the module being tested is known to the tester

White Box Testing

- Using white-box testing methods, you can derive test cases that
  - Guarantee that all independent paths within a module have been exercised at least once
  - Exercise all logical decisions on their true and false sides
  - Execute all loops at their boundaries
  - Exercise internal data structures to ensure their validity

WHITE BOX TESTING

Goal is to ensure that all statements and conditions have been executed at least once.



White Box Testing

- **Advantages**
  - Testing can be commenced at an earlier stage as one need not wait for the GUI to be available.
  - Testing is more thorough, with the possibility of covering most paths

White Box Testing

- **Disadvantages**
  - Since tests can be very complex, highly skilled resources are required, with thorough knowledge of programming and implementation
  - Test script maintenance can be a burden, if the implementation changes too frequently

THANK YOU!!!!