|  |
| --- |
| **Objectives:**   * **To introduce basic types of testing concepts.** * **To develop an appreciation in terms of different levels of testing approaches and their inter-relations.** * **To sensitize students as to how testing techniques vary with the levels of system development** |

**4.1 Basic Types of Testing**

The following figure represents the basic types of testing:

Black Box Testing

White Box Testing

(Logic of the Program)

Functional

Structural

Static

Dynamic

**Testing**

(Functionality of the program)

**Fig 4.1 Testing Classifications**

**4.1.1 Static Testing**

Static testing refers to testing something that’s not executing - examining and reviewing it.

**4.1.2 Dynamic Testing**

Dynamic testing is what you would normally think of testing - executing and using the software. Techniques used are determined by the type of testing that must be conducted

* Functional Testing
* Structural Testing

**a) Functional Testing**

Structure of the program is not considered. Test cases are decided based on the requirements or specification of the program or module. Hence it is often called as “Black Box Testing”.

**b) Structural Testing**

Structural testing is concerned with testing the implementation of the program. The focus is made on the internal structure of the program. The intent of structural testing is not to exercise all the different input or output condition but to exercise the different programming structures and data structures in the program. It applies for the testing technique called as “white box”.

**4.2 Software Testing Fundamentals**

Testing performed in most organizations, is a process designed to compensate for an ineffective software development process. Testing is no longer an adjunct to the system development life cycle, but rather a key part of it. There is no real “shortcut” to a sound testing philosophy, but the highest payback comes from detecting problems early in the system development life cycle in order to avoid designing and coding the system incorrectly and then later correcting those defects. The objective of life cycle testing is to spend more effort testing right from the start of the project, and less time debugging the coding and testing phase of development.

**4.3 What is Primary role of Software Testing?**

Determine whether the system meets specification (Producer view)

Determine whether the system meets business and user needs (Customer view)

**4.4 What is Defects?**

The purpose of testing is to find defects. A defect is a variance from a desire product attribute. Two categories of defects are:

* Variance from product specifications
* Variance from customer/user expectation

**4.4.1 Variance from Product Specifications**

The product built varies from the product specified. For example, the specifications may say that A is to be added to B to produce C. if the algorithm in the built product varies from that specification, it is considered to be defective.

**4.4.2 Variance from Customer/User Expectation**

This variance is something that the user wanted, that is not in the built product, but also was not specified to be included in the built product. The missing piece may be a specification or requirement, or the method by which the requirement was implemented may be unsatisfactory.

Defect generally fall into one of the following 3 categories:

* **Wrong:** The specifications have been implemented incorrectly. This defect is a variance from customer/user specification.
* **Missing:** The specified or wanted requirement is not in the built produect. This can be a variance from specification, an indication that the specification was not implemented, or a requirement of the customer identified during or after the product was built.
* **Extra:** A requirement incorporated into the product that was not specified. This is always a variance from specifications, but may the user of the product desire an attribute. However, it is considered a defect.

**4.4.3 Testing Policy Vs Quality Policy**

**a) Testing Policy**

A Testing Policy is management’s definition of testing a department. A testing policy involves the following four criteria.

* **Definition of Testing:** A clear, brief and unambiguous definition of Testing.
* **Testing System:** The method through which testing will be achieved and enforced.
* **Evaluation:** How testing team will measure and evaluate testing.
* **Standards:** The standards against which testing will be measured.

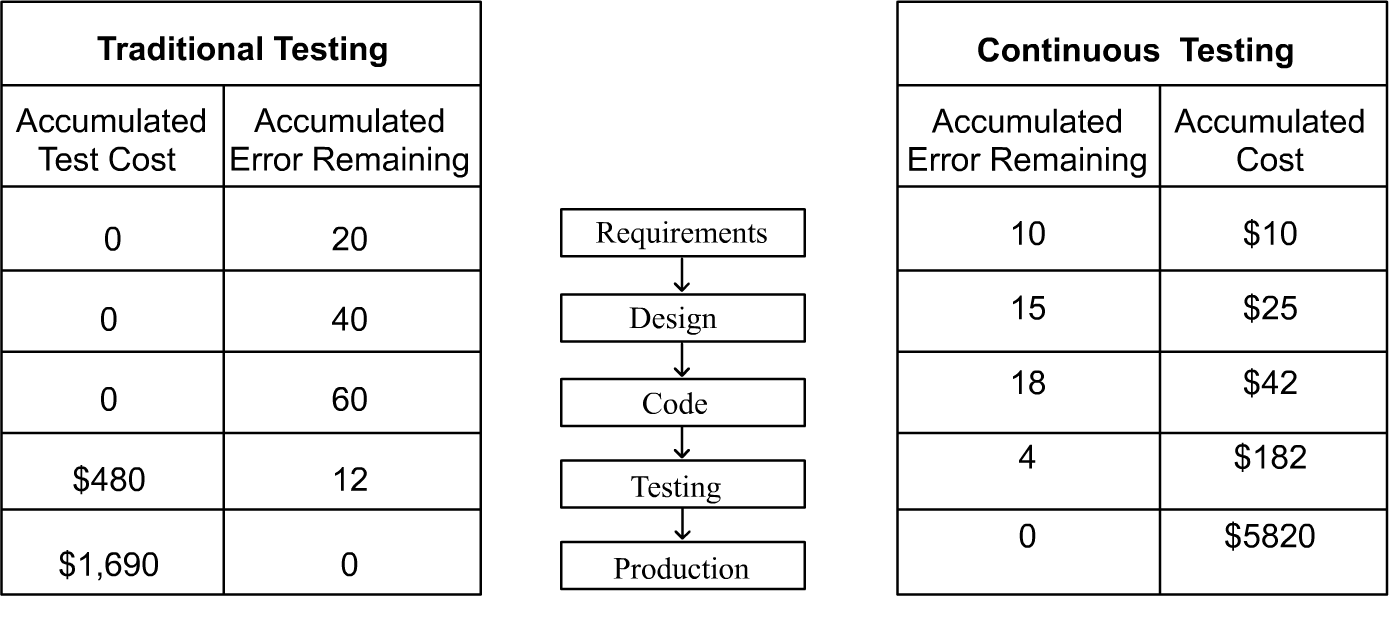
**b) Quality Policy**

A Quality Policy is again a management definition of providing customer satisfaction for the first time and every time. Understanding the definitions of excellence and quality is important because it is starting point of any management team contemplating the implementation of a quality policy.

**4.5 Testing Economics and Testing Cost**

**4.5.1 Cost of Quality**

Cost of quality is a term that is used to quantify the total cost of failure, appraisal, and prevention costs associated with the production of software.



**4.6 Testing Levels**

* Unit Testing
* Integration Testing
* System Testing
* Acceptance Testing

**4.6.1 Unit Testing**

Unit Testing is a testing in which the individual unit of the software is tested in isolation from other parts of a program.

**4.6.2 Integration Testing**

Integration Testing refers to the testing in which Software units of an application are combined and tested for a communication interfaces between them. There are three types of integration testing:

* Big Bang Testing
* Bottom Up Testing
* Top Down Testing

**a) Big Bang Testing**

A type of Integration Testing in which software components of an application are combined all at once into a overall system. According to this approach, every module is first unit tested in isolated from every module. After that, each module combined all at once and tested.

**Fig 4.1 CAM Model**

**b) Bottom Up Testing**

In bottom up integration testing, all the modules are added or combined from lower level hierarchy to higher level hierarchy i.e. the lower model is tested in isolation first, then the next set of higher level modules are tested with the previously tested lower modules.

Drivers are replaced one at a time, Depth first

Worker modules are grouped into builds & integrated

A

F

B

G

E

C

D

A cluster

**Fig 4.2 Integration Testing: Bottom-up**

**c) Top Down Integration Testing**

In Top down integration testing, all the modules are added or combined from higher level hierarchy to lower level hierarchy i.e. the higher model is tested in isolation first, then the next set of lower level models are tested with the previously tested higher models.

Stubs are replaced one at a time “Depth first”

C

D

E

A

B

F

G

Top module is tested with stub

As new modules are integrated some subset of the tests re run

**Fig 4.3 Integration Testing: Top-down**

**4.6.3 System Testing**

Testing conducted on a complete, integrated system’s compliance with its specified requirements.

**4.6.4 Acceptance Testing**

The testing conducted by client to evaluate the system compliance as per the business requirements.

**4.7 Testing Techniques**

**4.7.1 White Box Testing Techniques**

White Box Testing examines the basic program structure and it derives the test data from the program logic; ensuring that all statements and conditions have been executed at least ones.

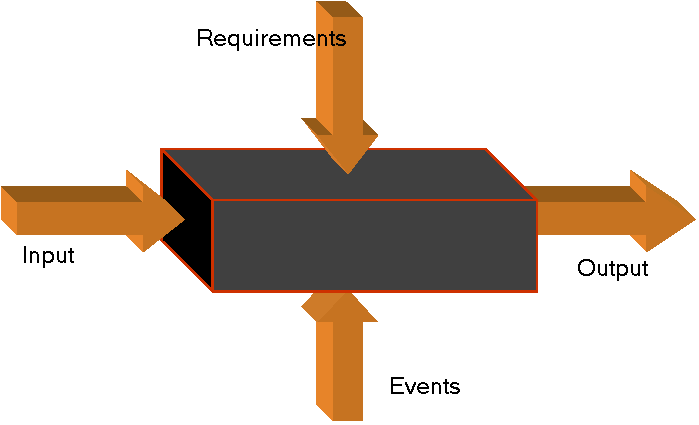
White box tests verify that the software design is valid and also whether it was built according to the specified design.

* **Statement Coverage:** Execute all statements atleast once.
* **Decision coverage:** Execute each decision direction atleast once.
* **Conditional Coverage:** Execute each decision with all possible outcomes atleast once.
* **Decision/condition Coverage:** Execute all possible combinations of condition outcomes in each decision. Treat all iterations as two-way conditions exercising the loop zero times and once.
* **Multiple Condition coverage:** Invokes each point of entry atleast once.

**4.7.2 Black Box Testing Technique**

Black Box Testing methods focus on the functional requirements of the software. This type of testing method attempts to find incorrect or missing functions, error in data structures or external database access, interface errors, performance errors, and initialization and termination errors.

Black Box Testing is conducted on integrated, functional components whose design integrity has been verified through completion of traceable white box tests. Black box testing traces the requirements focusing on system externals. It validates that the software meets the requirements irrespective of the paths of execution taken to meet each requirement.

****

**Fig 4.2 CAM Model**

Black box testing focuses testing the function of the program or application against its specifications. Specially, this technique determines whether combinations of inputs and operations produce expected results.

When creating black box test cases, the input data used is critical. Three successful techniques for managing the amount of input data required include:

* **Equivalence Partitioning:** An equivalence class is a subset of data that is representative of a larger class. Equivalence partitioning is a technique for testing equivalence classes rather than undertaking exhaustive testing of each value of the larger class. For example, a program which edits credit limits within a given range ($10,000 - $15,000) would have three equivalence classes:
  + Less than $10,000(Invalid)
  + Between $10,000 and $15,000(Valid)
  + Grater than $15,000(Invalid)



**Fig 4.3 CAM Model**

* **Boundary Value Analysis:** A technique that consists of developing test cases and data that focus on the input and output boundaries of a given function. In same credit limit example, boundary analysis would test:
  + Low boundary plus or minus one($9,999 and $10,001)
  + On the boundary ($10,000 and $15,000)
  + Upper boundary plus or minus one ($14,999 and $15,001)
* **Error Guessing:** Based on the theory that test cases can be developed based upon the intuition and experience of the test engineer. For example, in an example where one of the inputs is the date, a test engineer may try February 29, 200 or 9.9.99.

**4.7.3 Incremental Testing**

Incremental testing is a disciplined method of testing the interfaces between unit-tested programs as well as between system components. It involves adding unit-tested programs to a given module or component one by one, and testing each result and combination. There are two types of incremental testing:

* **Top-down** begins testing from the top of the module hierarchy and work down to the bottom using interim stubs to stimulate lower interfacing modules or programs. Modules are added in descending hierarchical order.
* **Bottom-up** begins testing from the bottom of the hierarchy and works up to the top. Modules are added in ascending hierarchical order. Bottom-up testing requires the development of driver modules which provide the test input, call the module or program being tested, and display test output.

There are procedures and constrains associated with each of these methods, although bottom-up testing is often thought to be easier to use. Drivers are often easier to create than stubs, and can serve multiple purposes. Output is also often easier to examine in bottom-up testing, as the output always comes from the module directly above the module under test.

* + 1. **Thread Testing**

This test technique, which is often used during early integration testing, demonstrates key functional capabilities by testing a string of units that accomplish a specific function in the application. Thread testing and incremental testing are usually utilized together. For example, units can undergo incremental testing until enough units are integrated and a single business function can be performed, threading through the integrated components.

The table below illustrates how the various techniques can be used throughout the standard test levels.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Levels/Techniques** | **White Box Testing** | **Black Box Testing** | **Incremental testing** | **Thread Testing** |
| Unit Testing | X |  |  |  |
| Integration Testing | X | X | X | X |
| System Testing |  | X |  |  |
| Acceptance Testing |  | X |  |  |

* 1. **V Model**

**Unit**

**Integration**

**System**

**UAT**

**LLD**

**HLD**

**SRS**

**Business Scenario**

**Fig 4.4 V Model**

**4.9 Summary**

* Static testing refers to testing something that’s not executing - examining and reviewing it.
* Dynamic testing is what you would normally think of testing - executing and using the software.
* In functional testing, test cases are decided based on the requirements or specification of the program or module
* Structural testing is concerned with testing the implementation of the program.
* Cost of quality is a term that is used to quantify the total cost of failure, appraisal, and prevention costs associated with the production of software.
* Integration testing refers to the testing in which Software units of an application are combined and tested for a communication interfaces between them. There are three types of integration testing:
* Big Bang Testing
* Bottom Up Testing
* Top Down Testing
* Thread testing represents the key functional capabilities by testing a string of units that accomplish a specific function in the application

**Review Questions**

1. What is the difference between testing policy and quality policy?

2. Explain the types of testing levels.

3. What is incremental testing?

4. Describe black box testing technique.

5. What is defect and what are the two views of defects?