# ARUNAI ENGINEERING COLLEGE

(Affiliated to Anna University)
Velu Nagar,Thiruvannamalai-606 603
www.arunai.org

## DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

## BACHELOR OF ENGINEERING

## THIRD YEAR

### SIXTH SEMESTER

# IT8076 - SOFTWARE TESTING

# IT8076-SOFTWARE TESTING

## UNIT I INTRODUCTION

Testing as an Engineering Activity – Testing as a Process – Testing Maturity Model- Testing axioms – Basic definitions – Software Testing Principles – The Tester's Role in a Software Development Organization – Origins of Defects – Cost of defects – Defect Classes – The Defect Repository and Test Design –Defect Examples- Developer/Tester Support of Developing a Defect Repository.

## UNIT II TEST CASE DESIGN STRATEGIES

Test case Design Strategies – Using Black Box Approach to Test Case Design – Boundary Value Analysis – Equivalence Class Partitioning – State based testing – Cause-effect graphing – Compatibility testing – user documentation testing – domain testing – Random Testing – Requirements based testing – Using White Box Approach to Test design – Test Adequacy Criteria – static testing vs. structural testing – code functional testing – Coverage and Control Flow Graphs – Covering Code Logic – Paths – code complexity testing – Additional White box testing approaches- Evaluating Test Adequacy Criteria.

## UNIT III LEVELS OF TESTING

The need for Levels of Testing – Unit Test – Unit Test Planning – Designing the Unit Tests – The Test Harness – Running the Unit tests and Recording results – Integration tests – Designing Integration Tests – Integration Test Planning – Scenario testing – Defect bash elimination System Testing – Acceptance testing – Performance testing – Regression Testing – Internationalization testing – Ad-hoc testing – Alpha, Beta Tests – Testing OO systems – Usability and Accessibility testing – Configuration testing –Compatibility testing – Testing the documentation – Website testing.

## UNIT IV TEST MANAGEMENT

People and organizational issues in testing – Organization structures for testing teams – testing services – Test Planning – Test Plan Components – Test Plan Attachments – Locating Test Items – test management – test process – Reporting Test Results – Introducing the test specialist – Skills needed by a test specialist – Building a Testing Group- The Structure of Testing Group.The Technical Training Program

## UNIT V TEST AUTOMATION

Software test automation – skills needed for automation – scope of automation – design and architecture for automation – requirements for a test tool – challenges in automation – Test metrics and measurements – project, progress and productivity metrics

# IT8076
# SOFTWARE TESTING

# UNIT-1 INTRODUCTION

# 2 MARKS & 16 MARKS
# WITH ANSWERS

# UNIT - I

## Part - A

1 Define the objective of software testing. [May/june-16]
Mention the objectives of software testing [Nov/Dec-16]

* The major objectives of software testing are as follows.

1) To prevent defects
2) Finding defects which may get created by the Programmer while developing the software
3) Providing Information about the level of Quality.
4) To make sure that the end result meets the business and User requirements

2. Differentiate Error, Defect and failure. [May/june-16]
(Nov/Dec 19)

Error:

* An Error is mistake or misconception or misunderstanding on the part of a software developer

Define Defects with an Example.
Defects (Faults): [Nov/Dec-2016] [Nov/Dec-2014]

* It is introduced into the software as the result of an error. Eg coins to dollars Conversion problem

* It is an anomaly in the software that the System behave incorrectly and not according to its specification

Failure:

* Unsuccessful software produce failures.

* A failure is the inability of a software or Component to perform its required functions within specified performance requirements.

---

3. Mention the role of test engineer in software development organization. [APR/MAY -17]

Tester important roles are

1. To reveal defects
2. To find weak points
3. To know the inconsistent behavior of system
4. To find the situations at which Software does not work.
5. Try to produce High quality Software
6. Try to satisfy User Requirements and Needs
7. Responsible for Low Defect software
8. Inform the Errors/Defects to Developers.

---

Define Software Quality. [May/june-16] (Nov/Dec19,18)

* Quality relates to the degree to which a system, system component or process meets speified requirements.

* Quality relates to the degree to which a system, system component or process meets customer or user needs. or expectations.

---

What is test case? Give Example [May/june-16]

* A test case in a practical sense is a test related item which contains the following information.

↳ A set of test inputs - These are data items received from an external source by the code under test. The external source can be hardware, software or human.

↳ Execution conditions - These are conditions required for running the test Eg Configuration of a hardware device

↳ Expected outputs - These are the specified results to be produced by the code under test

---

Define Test oracle and Test Bed [APR/MAY-15] April, may18

Test oracle:-

"A test oracle is a document or piece of software that allows testers to determine whether test has been passed or failed"

Test Bed :- Nov, dec17

A test bed is an environment that contains all the hardware and software needed to test a software component or a software system.

7. Mention the Quality attributes of software [Apr/may-1
Nov, dec17

The Quality attributes of software are

1. Correctness
2. Reliability
3. Usability
4. Integrity
5. Portability
6. Maintainability
7 Interoperability

8 what are the sources of Defects [Apr/May-17]
(Nov/Dec19,18)

- * Education
- * communication
- * oversight
- * Transcription
- * process

9. Mention any two role of process in Software Quality [NOV/DEC-14]

The need for software products of high quality has pressured those in the profession to identify

i) To identify and quantify quality factors such as usability, testability, maintainability and reliability and

ii) To identify Engineering practices that support the Productionof quality products having these favorable attributes.

1. State and explain in detail the various software testing principles [May/June-16] [Apr/may-17]

Elaborate on the principles of software testing and Summarize the tester role in software development organization [Nov/Dec-14] [Nov/Dec-16]

(Nov/Dec 19,18,17 Apr/may 18)

Software Testing Principles:

* Testing Principles are important to test specialists/ engineers because they provide the foundation for developing testing knowledge and acquiring testing skills.

* It is a foundation for

a) developing testing knowledge

b) Acquiring testing Skills

* A principle can be defined as

1. A general or fundamental, law, doctrine or assumption

2. A rule or code of conduct

3. Ther law or facts of nature

* In Software domain, Principles may also refer to rules or codes of conduct relating to professionals who design, develop, test and maintain Software systems

Principles 1:

The testing is the process of exercising a Software component using a selected set of test cases, with the intent of

i) revealing defects and

ii) Evaluating quality

* The principle supports testing as an execution-based activity to detect defects

* It also supports the separation of testing from debugging, since the intent of the latter to locate defects and repair the software.

* In the case of the latter, the tester executes the software using test cases to evaluate properties such as reliability, usability, maintainability and level of Performance

* Test results are used to compare the actual properties of the Software to those specified in the requirement document as a quality goal.

principle 2:

* when the test objective is to detect defects then a good test case is one that has a high Probability of revealing a yet-undetected errors.

* It supports careful test design and provides a criterion with which to evaluate test case design and the effectiveness of the testing effort when the objective is to detect defects.

* The goal of the test is to prove/disprove the hypothesis (ie) determine if the specific defect is present/absent

Principle 3:

* Test results should be inspected meticulously.

* Testers need to carefully inspect and interpret tests results several erroneous and costly scenarios may occur if case is not taken.

Eg: A failure may be overlooked and the test may be granted a "pass" status when in reality the software has failed the test.

* A failure may be suspected when in reality non exists.

* The outcome of quality test may be misunderstood, resulting in unnecessary network, or oversight of a critical problem.

Principle 4:

A test case must contain the expected output or result.

* Expected outputs allow the tester to determine
  i) whether a defect has been revealed
  ii) Pass/fail Status for the test

* It is very important to have a correct Statement of the output so that needless time is not spent due to misconceptions about the outcome of a test

Principle 5:

Test cases should be developed for both valid and Invalid input conditions.

* A tester must not assume the Software Under test will always be provided with valid inputs

* Inputs may be incorrect for several reasons

Eg software users may have misunderstandings or lack information about the nature of the inputs

Principle 6:

The Probability of the existence of additional defects in a Software Component is proportional to the number of defects already detected in the Component.

* This principle is that the higher number of defects already detected in a component, the most likely it is to have additional defects when it undergoes further testing.

Eg Two components A and B and tester have found 20 defects in A and 3 defects in B, then the probability of the existence of additional defects in A is higher than B.

* This empirical observation may be due to several causes.

Principle 7:

Testing should be carried out by a group that is independent of the development group.

Tester must realize that

i) Developers have a great deal of pride of their work

ii) on a practical level it may be difficult for them to conceptualize where defects could be found.

Testing is not successful for

1. Misunderstanding of requirements

2. Misunderstanding of specifications relating to software

3. Hard to known the faults

4. Difficulty in locating the defects

Its solved by Independent testing Group (ITG) People.

1. These people one involved in an organization to find the defects

2. Its implemented as a completely separate functional entity in an organization.

3. The SQA group may hold the testers as their members

4. Testers should not play "gotcha" games with developers

5. There should be cooperation in the group, then only a software with highest quality will be produced

Principle 8:

Tests must be repeatable and reusable

* Testing repeatedly is termed as "Regression test"

* Repeat the tests after the defect is repaired

* Repeatable to test, avoid the duplicate test conditions

* It keeps exact information/conditions of the testing.

Principle 9:

## Testing should be planned

* The test plan describes the objective of testing.

* Its used to verify that whether enough time and resources are allocated for the testing tasks or not.

* Testing is easily monitored and managed through test plan.

* Test planning is accommodated with project Planning

* It helps to maintain mutual relationship between Project manager and test manager.

* After the Software is developed by the developer in the correct date, testercan test the software on particular test, otherwise tester cannot test at the time

* The test risk has to be evaluated.

Principle 10:

Testing Activities Should be integrated into the software Life cycle.

* It is no. longer feasible to postpone testing activities until after the Code has been written.

* In addition to test Planning, Some other types of testing activities such as usability testing can also be

carried out early in the life cycle by using prototypes

* These activities can continue on until the software is delivered to the users.

Principle 11:

Testing is a creative and challenging task.

Difficulties and challenges for the tester include the following

1. Tester needs to have comprehensive knowledge of the software engineering discipline.

2. A tester need to have knowledge from both experienc and education as to

i) How software is specified ii) Designed iii) Develope

3 A tester needs to be able to manage many details

4. Tester must know about test fault and its root cast

5. Tester must have hypothesis knowledge

6. Tester must understand what has to be tested?

7. Before testing, Test cases must be prepared.

8. Test procedures should be designed & recorded

9. Tester must Analyze the test results

10. Tester must have a knowledge about testing tool

11. Tester must be educated & trained in particular area

12 Tester keep Good relationship wit clients, users, designers, developer, requirements engineer

## Tester role in a Software Development organization

Tester important roles are (nov/Dec 19,17) (apr/may 18)

1. To reveal defects
2. To find weak points
3. To know the inconsistent behavior of system
4. To find the situations at which software does not work
5. To get more programming experience, it helps to
   - * Understanding software/system
   - * How code is developed
   - * Possibilities of errors.
   - * when error it may be occured
   - * which situation error it may occured
6. Try to produce High Quality software
7. Try to Satisfy User Requirements and Needs
8. Tester combined work with Test Manager and Project Manager it helps to
   - * To Prepare test plans
   - * To maintain organizational Testing Standards, Policies, goals, procedures

9. Responsible for Low Defect Software

10. TO minimize the Costs for support

11. TO Deliver the Software as per the customer Needs, also Reliable, Usable

12. Inform the Errors/ Defects to Developers

13. Tester needs

↳ Communication Skills

↳ Team working Skills

↳ Decision making Skills

↳ Scripting knowledge/ coding Skills

↳ working experience

what are the typical origins of defects? Explain the major classes of defects in the software artefacts - [Apr/May-17] [May/june-16] (nov/Dec 19) (apr/may 18)

Describe the defect classes in detail with example (nov/Dec 18) [Apr/may-15]

Origins of Defects:

* It is a variation between actual software requirement specification (SRS) and final executed build (ie exe file)

Types of defects

1. Defects from product specifications

* The product developed varies in the product specifications (ie SRS/ BRS/ CRS)

2. variance from customer/user expectations.

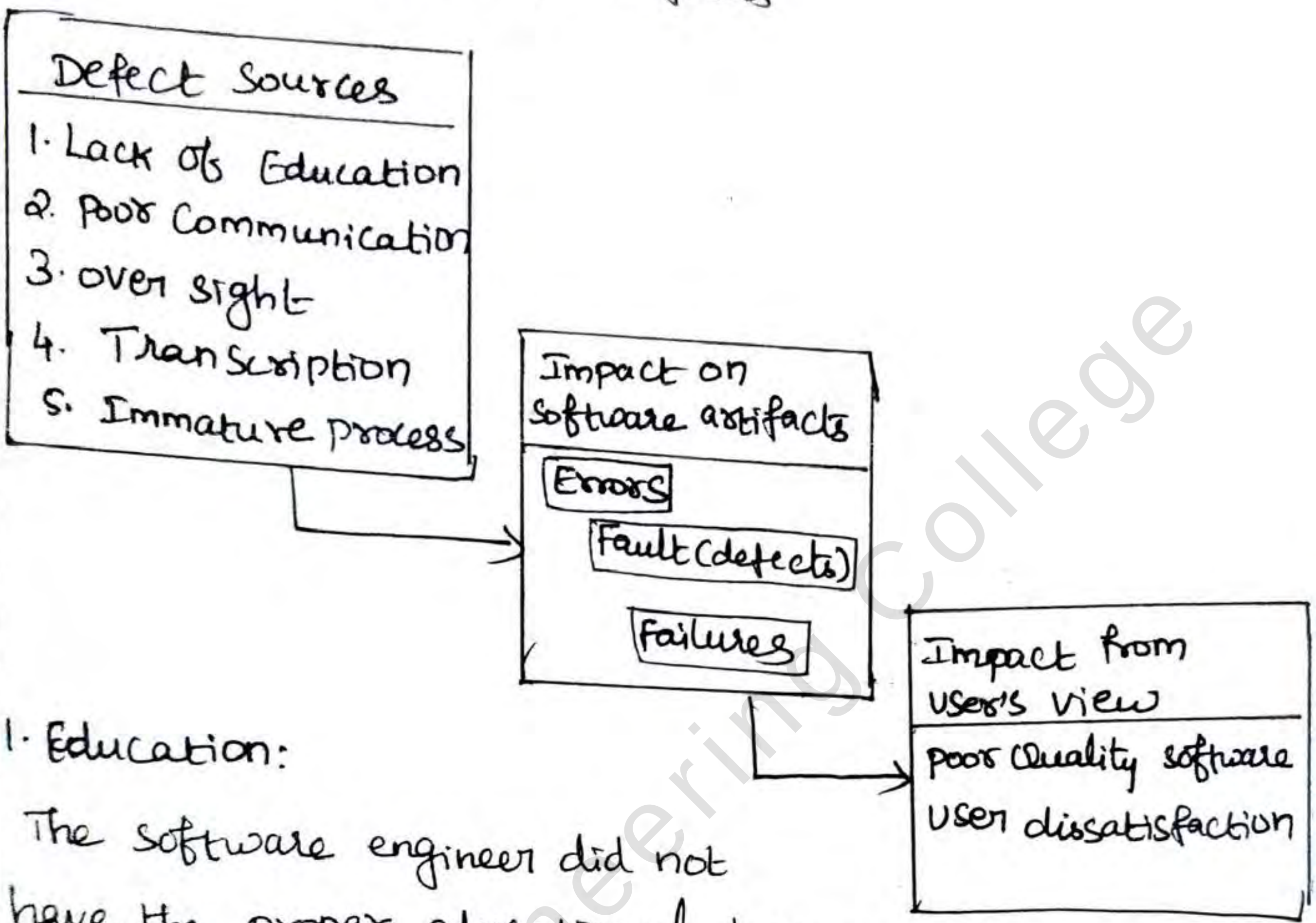
* Variance means difference

* This variance is something that the user wanted is not in the built product

* Defects have harmful affects on software user and software engineers work very hard to produce high-quality software with a low number of defects

Defects Sources:

Origins of defects



1. Education:

The software engineer did not have the proper educational background to prepare the software artifact

2. Communication:

* The Software engineer must communicate with group members properly. Eg Misunderstanding Stupidly

3. over sight:

* The software engineer omitted to do something Eg Initialization statement omitted

4. Transcription:

* The software engineer knows what to do, but makes a mistake in doing it

5. process:

* The process used by the software engineer misdirected his/her actions

# Defect classes:

* Defects classified in many ways.

* A single classification scheme is necessary to adapt and apply for all projects

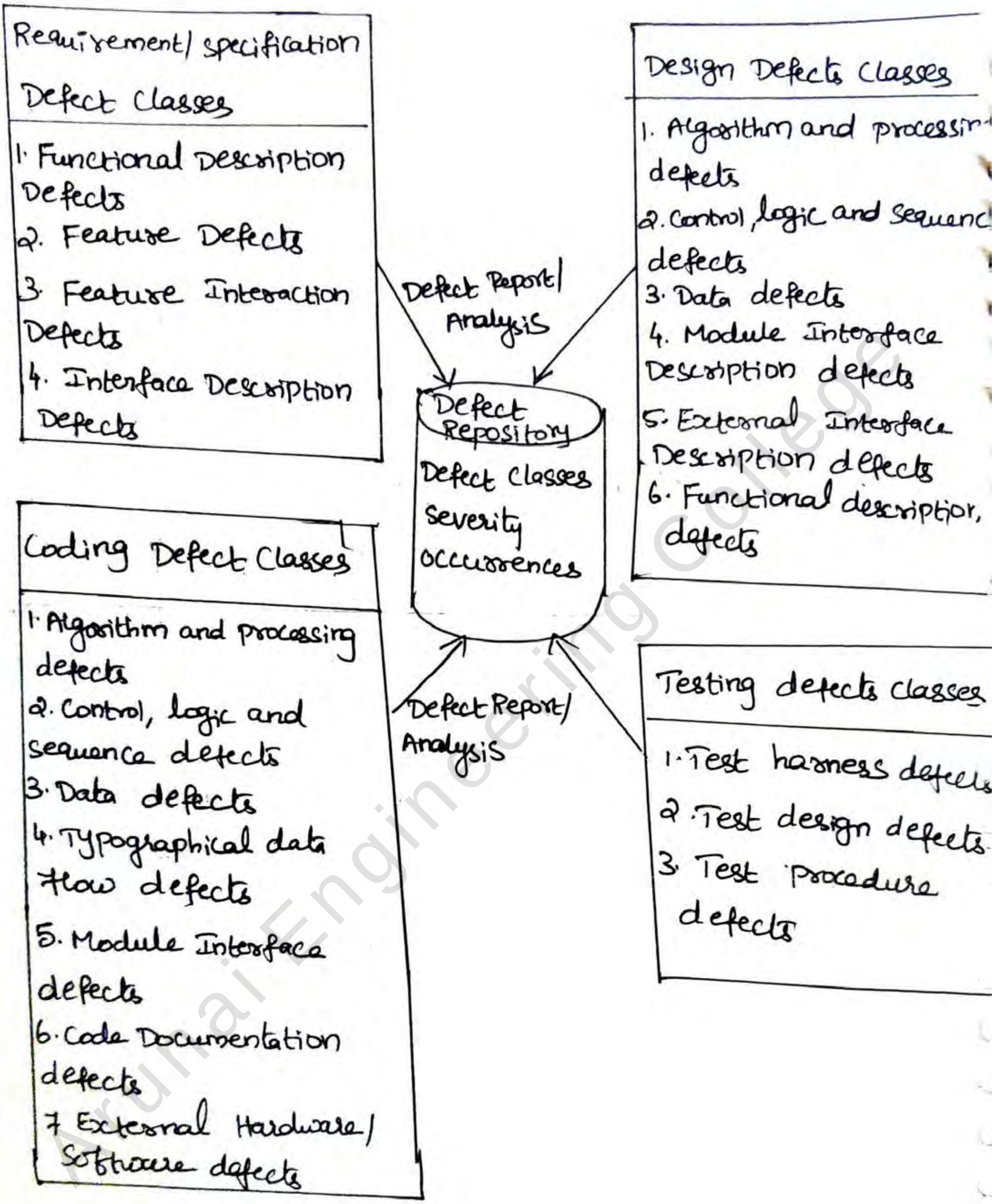
* Some defects fit into more than one classes or category.

* Developers, testers and SQA staff should try to be as consistent as possible when recording defect data

* The defect types and frequency of occurrence should be used to guide test planning and test design

* The defects are classified into 4 types, They are

1. Requirement and specification defects
2. Design Defects
3. Coding defects
4. Testing defects


Requirement/ specification Defect Classes
1. Functional Description Defects
2. Feature Defects
3. Feature Interaction Defects
4. Interface Description Defects
Design Defects Classes
1. Algorithm and processin defects
2. Control, logic and sequenc defects
3. Data defects
4. Module Interface Description defects
5. External Interface Description defects
6. Functional description defects
Defect Report/ Analysis
Defect Repository
Defect Classes
severity
occurrences
Coding Defect Classes
1. Algorithm and processing defects
2. Control, logic and sequence defects
3. Data defects
4. Typographical data flow defects
5. Module Interface defects
6. Code Documentation defects
7 External Hardware/ Software defects
Defect Report/ Analysis
Testing defects classes
1. Test harness defects
2. Test design defects.
3. Test procedure defects

Defect classes and the defect Repository

1. Requirement and Specification defects

* Defects injected in early phases is very difficult to remove in latter phase

* Requirement documents are written in natural language So it creates a chance for Ambiguous, contradictory, Unclear, redundant, Imprecise requirements

* Some of the Specification | requirments defects are

1. Functional Description Defects:

The overall description of what the system does, and how it should behave is

* Incorrect * Ambiguous * Incomplete

2. Feature Defects:

* Feature may be described as distinguish characteristics of a software component or system

* It describes Missing, incorrect, Incomplete, superfluous

3. Feature Interaction Defects:

* It describes, how the incorrect description interact in the feature.

* How the features interacts with another features

4. Interface Description Defects:

* It describes, how the target software is interfaced with

1. External software 2. Hardware 4. Users.

2. Design Defects:

* It occur when System Components, interaction between System components, interaction between the components and outside software/hardware and Interaction with Users.

* Defects may also occur in Algorithm design, control, Logic representation, Data elements, Module interface descriptions and external hardware/software/User Interface descriptions.

Design defects are

1. Algorithm and processing Defects:

* These occur when the processing steps in the algorithm as described by pseudocode are incorrect

2. Control, logic and Sequence Defects:

* Incorrectly developed pseudo code create control and logic defects

↳ Control defects - Poor logic flow in code

↳ Logic defects - Logic operator applied mistakenly

↳ Sequence defects - Conditions are not properly checked in the pseudocode

Eg Incorrect branching condition

3. Data Defects

* Due to poor data structure design, it create data defects

Eg Incorrect allocation of Memory, Lacking of field in a record

4. Module Interface Description Defects:

* Its derived from incorrect or inconsistent Parameter types, an in correct number of parameter, Incorrect ordering of Parameter

5. Functional Description Defects:

* It included incorrect, missing and/or unclear Design element

Eg Poor Explanation of function

6. External Interface Description Defects.

* These defects are derived from incorrect design descriptions for the interface with

↳ COTS components ↳ Data base
↳ External software system ↳ Hardware devices

3. Coding Defects:

* when executing the code, if any errors occurred it is Called "coding defects".

* It may be occurs on,

↳ Misunderstanding of programming languages and its construction

↳ Miscommunication with designes.

Codeing Defects are

1. Algorithmic and processing Defects

* It occurs on Unchecked overflow/underflow conditions, Data Conversion, Missing Parenthesis, Precision loss, In-correct order of Parenthesis

2. Control, logic and Sequence Defects

* It created by,

- In correct expression of case statements
- Incorrect iteration of loops
- Missing path and conditions

3. Typographical Defects

* These defects also called "Syntax error"

Eg Incorrect spelling of a variable name, that are usually detected by a compiler, Self review & peer reviews

4. Initialization Defects:

* It occurs on, when initialization statements are omitted or in-correct

* It may occur because of

1) Misunderstanding or Lack of Communication between programmers
2) Misunderstanding of programming environment
3) Carelessness

5. Data flow defects

* Poor operational Sequences create data flow defects

6. Data Defects:

* Due to poor data Structure implementation, it Create data defects

Eg Incorrect accessing of files

Module Interface Defects

* These defects are derived from

↳ Inconsistent Parameter types

↳ Incorreet number of parameters

↳ Improper Design ↳ Incorrect seauence of calls

8. Code Documentation Defects:

* When the code documentation does not reflect what the program actually does or is incomplete or ambiguous this is called a code documentation defect

9. External Hardware, software Interfaces defects:

* These defects a arised from System calls, Database linking, Memory usage, Resources usage, interrupts and Exception handling

4. Testing Defects:

*Test Plans, test cases, test harnesses and test Procedures can also contain defects.

*Testing defects are classified into two types

1. Test Harness Defects:

* It is also called the test harness or scaffolding code

* code is reusable code. Code is used again when the version of software is released.

* Code must be correctly, Designed, Implemented Tested.

*Otherwise, defects occurs on unit and Integration levels. so, code should be maintain and Support easily checking by the Software.

2. Test Case Design and Test Procedure Defects

*Test case design arise some valid defects incorrect test cases, incomplete test cases, missing test cases, inappropriate Test cases and test Procedures

* These defects are best detected in test plan review.

Explain the developer and tester support for the development of a defect repository. [May/june-16] (nov/Dec 18)

Developer and tester support for Development of defect Repository:

* A requirement for repository development Should be a part of testing and/or debugging Policy statements.

* Forms and templates will need to be designed to collect the data.

Eg The test incident reports and defect fix reports, Test reports

* Defect and its relevant information are stored in defect repository

* Defect repository development is a essential part of testing and debugging.

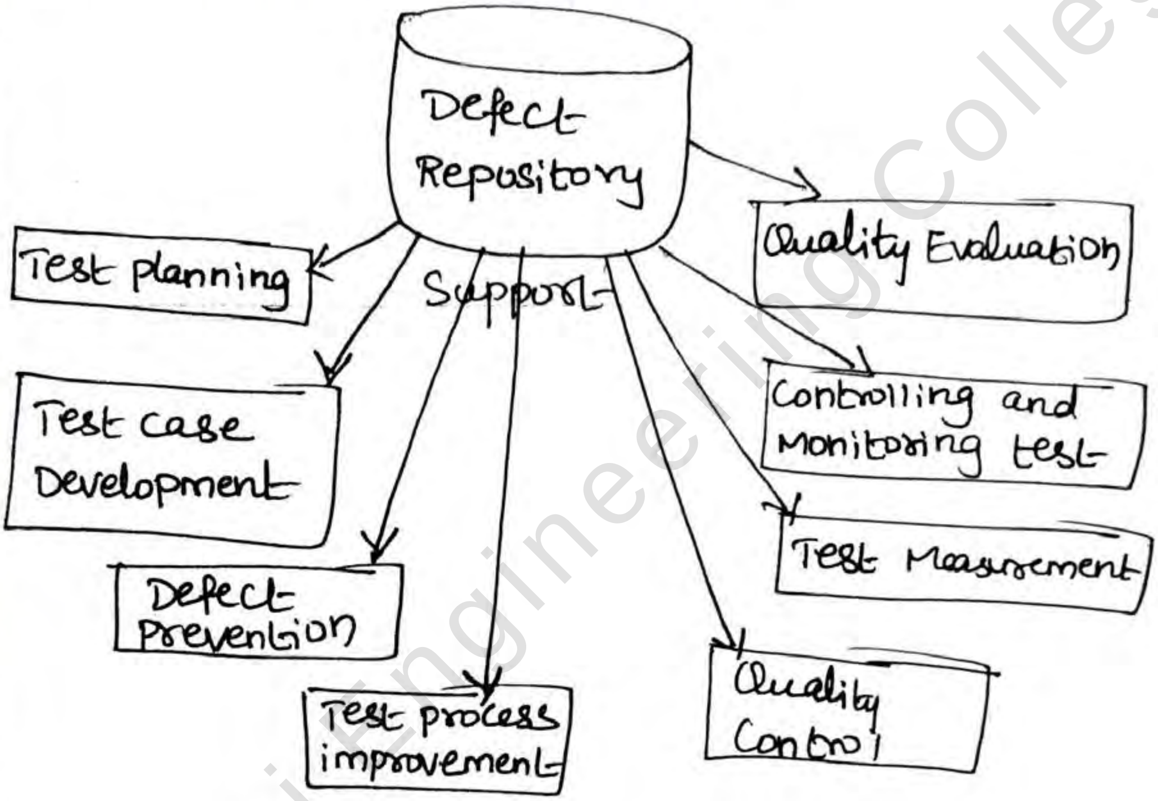
* Reports are to be recorded each defect and frequency of the occurrences for each defect type after testing.

* Every defects to be monitoring for each ongoing project

* when the tester/developer does changes in the Process, the distribution of defects will also be changed.

The defect data is used for test planning. It helps.

1. To choose testing techniques
2. To design test cases (if required)
3. To allocate the number of resources (if needed)
4. To estimate testing schedule
5. To estimate testing cost



TMM Maturity goals

1. Controlling and monitoring test
2. Quality evaluation and control
3. Test measurement
4. Test process improvement
5. Defect prevention
6. Test planning
7 Test case development

4. what approach would you use to solve the concepts of defects with the coin problem? [May/june-16] [Apr/may-17]

Defect Examples: The coin problem

Example: coins to dollars conversion Problem.

1. specification for program calculate-coin-values

Requirements:

* This program or system finds total dollars and cents value for a set of coins

*User input may be pennies, nickles, dimes, quarters etc.

* The program outputs must be the total dollars and cent values of the coins to the user.

* Number of coins is an integer (input)

* Number of dollars is an integer (output)

* Number of cents is also integer (output)

Specification Defects:

1. Functional Description Defects:

* The functional description defects arise because the function description is ambiguous and incomplete.

* It does not state

i. Input & output ii) Number of coins iii) Number of cents and dollars iv) value is zero or greater

5) Pre and post conditions

Eg: No. of coins>=0 (not to be negative)

Eg: No. of dollars >=0

vi) Accept invalid values or not

vii) pre and post conditions are to be solved by black box testing using

a) Boundary value Analysis

b) Negative value testing

c) Null and database testing

2. Interface Description Defects:

* It is relate to

i) Poor education/ unexperience people

ii) Ambiguous nature

iii) Incomplete nature of specification

Design Description for program calculate-coin-values

```
Program   Calc-Coin Value
          no-of-coins is integer
          total-coin-val- is integer
          no-of-cents is integer
          no-of-cents is integer
      coin-val is array of 6 integers
     coin-val initalized to: 1, 5, 10, 25, 25, 100 begin
        initialize total-coin-val to zero
initialize loop-Count to one
while loop-Count is less than 6 begin
  output "enter coins-count" read (no-of-coins)
  total-coin-val = total-coin-val + no-of-coin *
          coin-val [loop-count]
  increment loop-count
end
no-of-dollars= total-coin-val/100
no-of-cents = total-coin-val/100* no.of dollars
output (no-of-dollars, no-of-cents)
end.
```

Design Defects:

1. Control, logic and sequencing Defects

* The defect in this subclass arises from an incorrect "while" loop Condition.

2. Algorithmic and processing defects:

* These defects arise from

1. Lack of error Checks for incorrect and/or invalid inputs

2. Lack of path. Ex. error inputs

3 Rejection of error conditions checks like division by zero

3. Data Defects:

* The wrong values to be entered

* Coin-val read 1, 5, 10, 25, 50, 100

4. External Interface Description Defects:

* This defects arising from the absence of input messages or prompts that introduce the program to the user and the request input.

5) Control, logic and sequence Defects

* These include the loop variable increment step which is out of the scope of the loop.

* It Create logic defects

6. Algorithmic and processing Defects

* The division operator will create a problem.

7 Data flow defects

* The variable total-coin-val is not initialized its used before its defined

8 Data defects:

* Array coin-val is holding error when it is initialized. These errors are carried from designing to coding

9. External Hardware, Software Interface Defects:

* External function "Scanf" is correctly written

* "&" Symbol not included in "scanf" statement

10. code Documentation Defects:

* The Entire code is represented as a document which is incomplete and ambiguous

* The coding defects are solved by white box Testing, logical, Loop & branch testing, Control testing.

# IT8076-SOFTWARE TESTING

# UNIT-2
# TEST CASE DESIGN STRATEGIES

# 2 MARKS & 16 MARKS
# WITH ANSWERS

# Unit - II

## Part - A

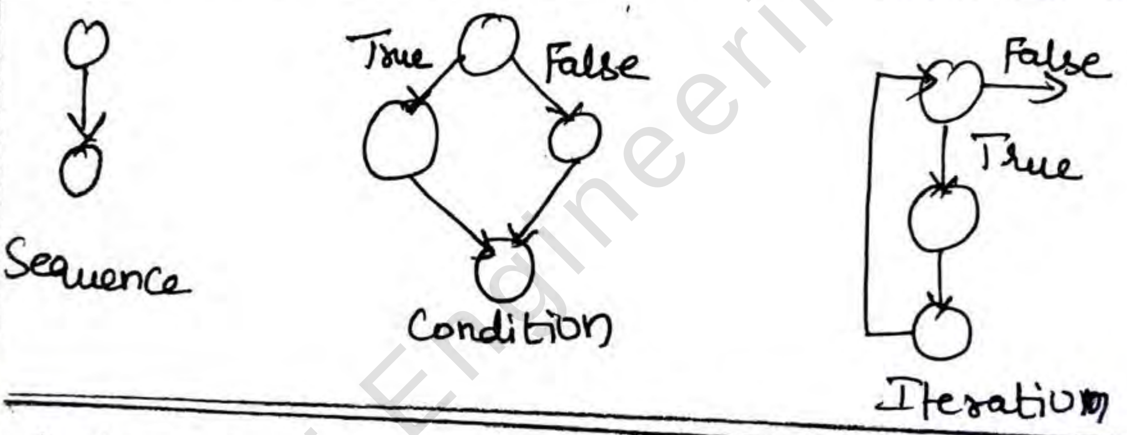
1. What are the basic primes for all Structured program? [May/june-16]

↳ sequential (Eg; computation, assignment statements)

↳ Condition (Eg; if/then/else statements)

↳ Iteration (Eg; while, for loop

Graphical representation of these three primes are below



Sequence

Condition

Iteration

2. what are the errors uncovered by black box testing? [may/june-16]

The errors uncovered by black box testing are

* Incorrect or missing functions
* Interface errors
* Errors in data structures
* Performance errors
* Initialization or termination error

3. Give a note on the procedure to compute Cyclomatic Complexity. [NOV/DEC-16]

1. Draw the flow graph from given set of Program Statements

2. The complexity value is usually calculated from control flow graph (G) by the formula

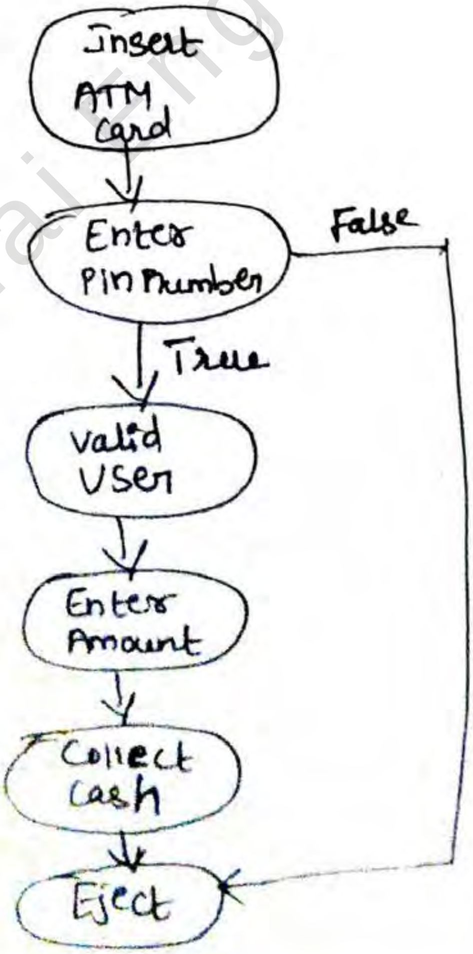
$$V(G) = E - N + 2$$

where

E - Number of edges in the Control flow graph

N - Number of nodes

4. Sketch the control flow graph for an ATM withdrawal system [NOV/DEC-16]



5. State the difference between white-box testing and (nov/Dec Dec19,18,17) black-box testing. [May|June-16] [Apr|May-17]

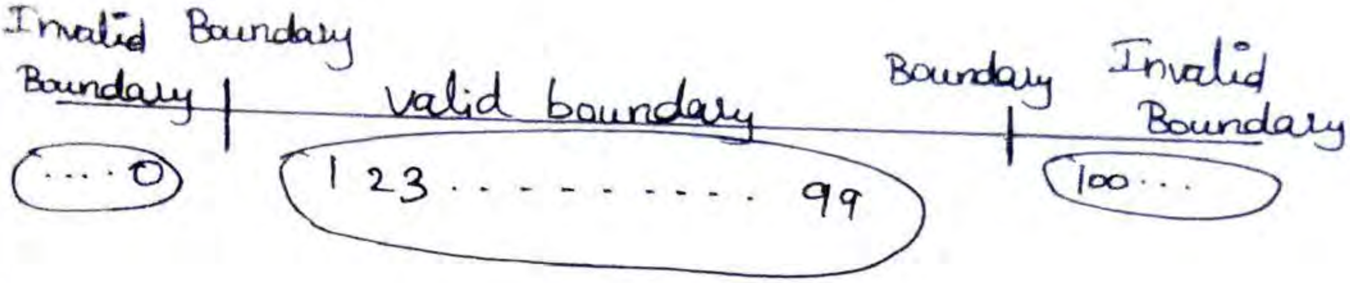
| White-box testing | Black-box Testing |
|---|---|
| 1. White-box testing is also called clear or glass box testing | 1. Black-box testing sometimes called "functional or specification testing", "closed box testing", "opaque testing", "Behavioral testing. |
| 2. White box approach is usually applied small size piece of software | 2. Black box approach is usually applied large size piece of software |
| 3. The white box approach focuses on the inner structure of the software to be tested | 3. Black box testing the tester is no knowledge of its inner structure The tester only has knowledge of what it does (focus only input & output) |

6. What is boundary value analysis? Eg [May|June-16]

* Boundary value analysis (BVA) method is useful for arriving at tests that are effective in catching defects that happen at boundaries.

* Boundary value analysis believes and extends the concept that the density of defect is more towards the boundaries.

Eg Consider a printer that has an input option of the number of copies to be made from 1 to 99 -



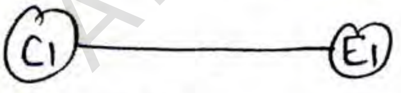
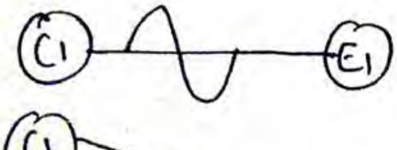
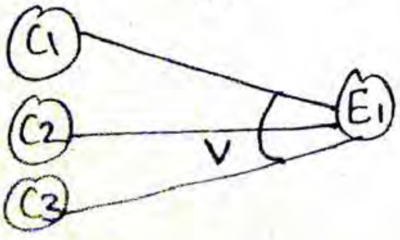
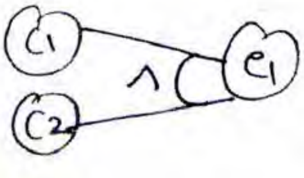
Boundary value Analysis

7 Define Test Adequacy criteria [Apr/may-15]

* A software test adequacy criterion predicate that defines "what properties of a program must be exercised to constitute a thorough test" (ie) one who sucessful execution implies no errors in a tested program.

8 Draw the notations used in cause Effect Graph [Apr/may-15]

* Cause Effect Graph is a visual representation of a logical relationship among the inputs and outputs which can be expressed as a boolean Expression

| Notation | Meaning |
|---|---|
|  | Identify |
|  | Not |
|  | or [OR] |
|  | AND |

9. What do you mean by code complexity testing? [Nov | Dec-14]

* Cyclomatic complexity is a source code complexity measurement that is being correlated to a number of coding errors.

* It is calculated by developing a control flow graph of the code that measures the number of linearly independent paths through a program module.

* These complexity measure helps to derive test cases. By using this test cases the basic set of execution paths are used.

10. How static testing is differing from structural testing [Nov|Dec-14]

| Static Testing | Structural Testing |
|---|---|
| 1) The product is tested by humans using just the source code and not the executables or binaries. | 1) These tests are actually run by the computer on the built product |
| 2) Static Testing done by humans in many ways. There are<br>i) Desk checking of the code<br>ii) Code walkthrough<br>iii) Code review<br>iv) Code Inspection | 2) Structural Testing types<br>i. Unit / code function Testing<br>ii) Code coverage Testing<br>iii) Code complexity Testing. |

| Static Testing | Structural Testing |
|---|---|
| 3. Static Testing accesses Requirements document, Design document, User manuals, Static Testing tool | 3. It focus on code, code Structure, internal design and how they are coded? |
| 4. It does not need computer as the testing of program is done without executing the program<br>Ex: Review, walkthrough, Inspection | 4. It must need computer as the testing of program is done,<br>Ex: Code based, functional based |

Explain the Significance of Control flow graph and cyclomatic complexity in white box testing with a Pseudo code for Sum of positive numbers. Also mention the independent paths with test cases

[may/june-16] [Apr/may -17]

Control Flow Graphs:

* "The logic elements most commonly considered for coverage are based on the flow of control in a unit of code".

Eg: 1) program statements

2) Decisions / branches

3) conditions

4) combinations of decisions and conditions

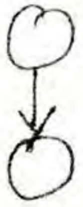
5) paths

* These logical elements are rooted in the concept of a program prime.

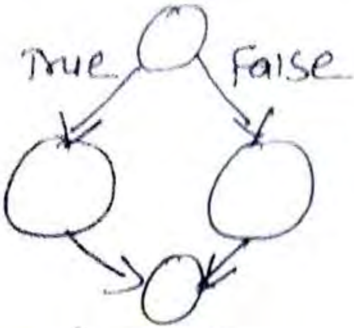
* A program prime is an automatic programming unit.

* All Structural programs can be built from three basic primes

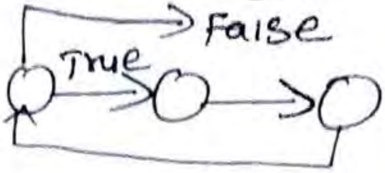
1) Sequence



2) Decision / condition



3) Iteration



* Using the concept of a Prime and the ability to use combinations of primes to develop Structural code, a flow diagram for the Software Unit under test can be developed.

* The flow graph can be used by the tester to evaluate the code with respect to its testability as well as to develop white box test cases.

* A Control flow graph describes in which the different instructions of a program gets executed.

* Control flow graph also says how the control flow through the Program.

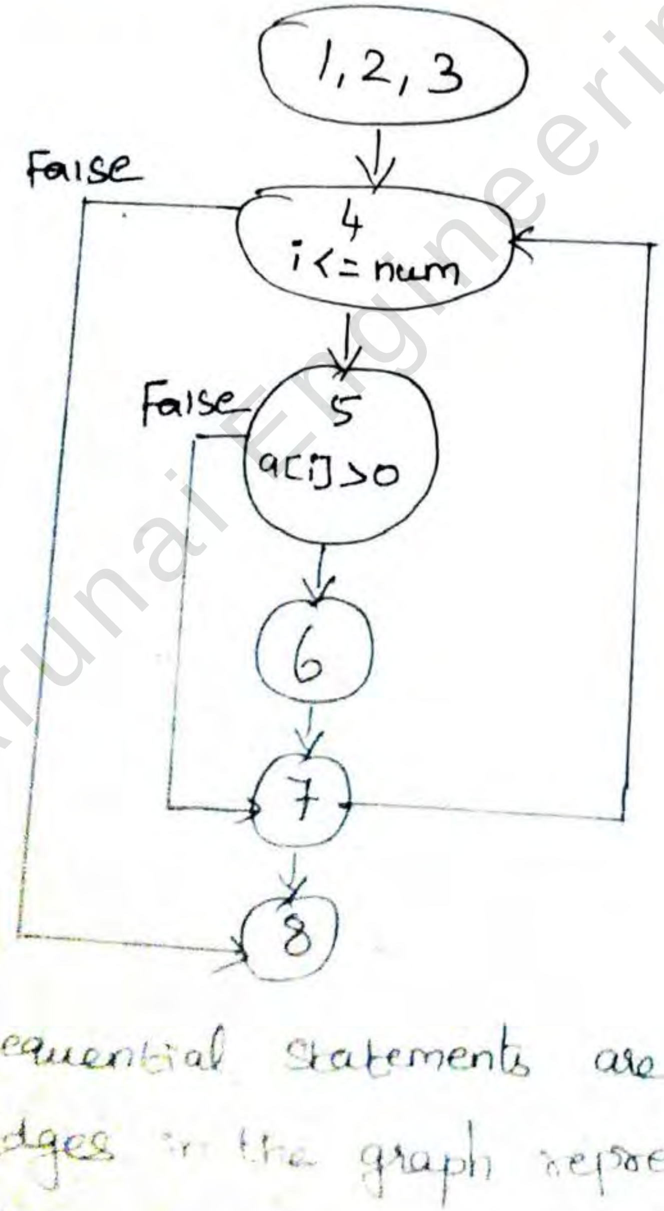
* Code Sample with branch and loop

```
POS_SUM (a, num, sum)
 Sum =0
 int i=1
  while(i<=num)
     if(a[i]>0)
```

```
        Sum = Sum + a[i]
    endif
        i = i + 1
    end while
end POS_SUM
```

* In the flow graph the nodes represent sequential statements, as well as decision and looping predicates.

A control flow graph representation for the code



* Sequential statements are combined as a block

* Edges in the graph represent transfer of control

* An edge from one node to another node exists if the execution of the statement representing the first code can result in transfer of control to the other code.

* Each circle is said to be flow graph node

* The direction of the transfer depends on the outcome of the condition in the predicate (true or false)

* Arrow on the flow graph represent edges or links

* Commercial tools are used to generate control flow graphs from code and in some cases from Pseudo code.

* It support to generate control flow graphs for complex code.

* It helps to design white box test cases.

Cyclomatic complexity

* The cyclomatic complexity is a software altribute and it is very useful to a tester.

* The complexity value is usually calculated from the Control flow graph(G) by the formula

$$\boxed{V(G) = E-N+2} \quad \text{or } V(G) = E-N+P$$

where

E- Number of edges in the control flow graph
N- Number of nodes in the flow graph
P - Number of nodes that have exit points

The cyclomatic complexity value is useful to the tester in several ways.

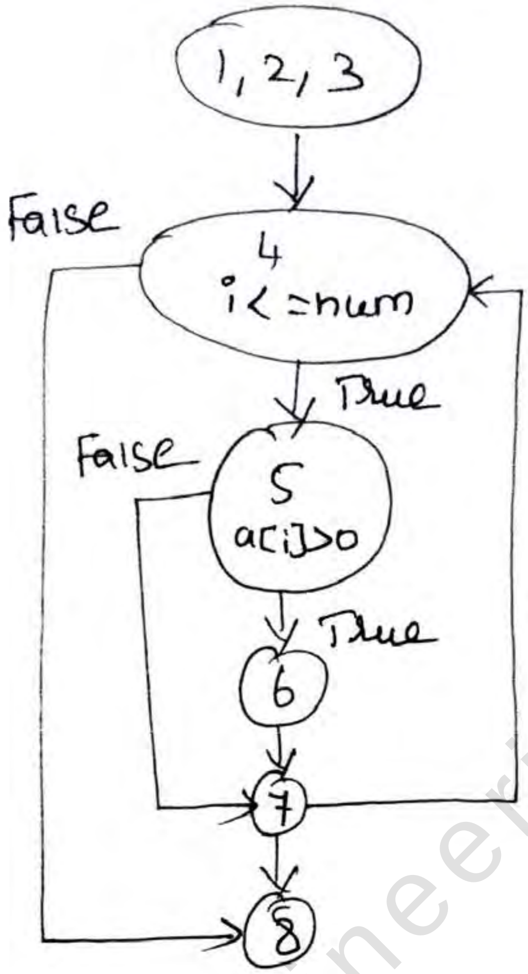
1) It provide an approximation of the number of test cases needed for branch coverage in a module of structural code.

2) The tester can use the value of V(G) along with past project data to approximate the testing time and resources required to test a software module.

3) Cyclomatic complexity value & control flow graph introduces another tool, it is called "path".

Path:

Control flow graph



E= 7, N=6

cyclomatic complexity

V(G) = 7-6+2

V(G) = 3

Path:

* A path is a sequence of control flow nodes usually beginning from the entry node of a graph through to the exit node.

* Paths are denoted by

1 - 2 - 3 - 4 - 8

"-" dashes represents edges between two nodes.

* Cyclomatic complexity is a measure of the number of "independent" paths in the graph.

Independent path:

* Deriving a set of independent paths Using a flow graph can support a tester in identifying the control flow features in the code and in setting coverage goals

* "The independent paths are defined as any new path through the graph that introduces a new edge that has not be traversed before the path is defined".

Set of independent paths for flow graph

i) 1 - 2 - 3 - 4 - 8

ii) 1 - 2 - 3 - 4 - 5 - 6 - 7 - 8

iii) 1 - 2 - 3 - 4 - 5 - 7 - 4 - 8

Basic set:

* A set of independent paths for a graph is called a basis set.

* The number of independent paths in a basis set is equal to the cyclomatic complexity of the graph.

* Identifying the independent paths provides useful support for achieving decision coverage goals.

Complete path coverage:

* Every path in a module must be exercised by the test set at least once.

* Even in a small and simple unit of code there may be many paths between the entry and exist node.

* Every loop multiplies the number of paths based on number of possible iteration of the loop.

* Complete path coverage for even a single module may not be practical & for large and complex modules it is not feasible.

* Some paths in a program may be unachievable (ie) they cannot be executed no matter what combinations of input datas are used.

* So, complete coverage for path cannot be obtained It is impossible to work

1. The basis set is a special set of paths
2. It does not represent all the paths in the module.
3. This is used as a tool to help the tester in the process of getting decision coverage.

P.M Demonstrate the various black box test cases Using Equivalence Class partitioning and boundary Value analysis to test a module for an ATM (nov/Dec 19, 18) [Nov/Dec-16] [May/june-16]

Illustrate with an Example the following black box testing techniques (apr/may 18) [Apr/May-17]

i) Equivalence class partitioning

ii) Boundary value Analysis

i) Equivalence Class Partitioning: (ECP)

* Equivalence Class partitioning is a software testing technique that evolves identifying a small set of representative input values that produce as many different output conditions as possible.

* The set of input values that generate one Single expected output is called a partition.

* when the behavior of the software is the same for a set of values then the set is termed as an equivalence Class (or) equivalence partition

* one sample from the partition is enough for testing as the result of picking up some more values from the set will be the same and will not yield any additional defects.

* All the values produce equal and same output they are termed as equivalence partition.

Testing by this techniques involves.

1) Identifying all partitions for the complete set of input, output values for a product.

2) Picking up one member value from each partition for testing to maximize complete coverage.

Advantages:

1) It gain good coverage with a small number of test cases

2) Redundancy of tests is minimized by not repeating the same tests for multiple values in the same partition

The equivalence partition table consists of

1. Equivalence partition definition
2. Type of input
3. Representative data for that partition
4. Expected Results

Each row is taken as a single test case and is executed.

The step to prepare an Ecp table are as follows.

1. Choose criteria for doing the Ecp (range, list of value etc)

2. Identify the valid Ecp based on the above criteria (number of ranges allowed values)

3. select a sample data from the Partition

4. write expected result based on the requirement given.

5. Identify special values if any, and include them in the table.

6. Check to have expected results for all the cases prepared.

7. If expected result is not clear for any particular test case, mark appropriately and escalate for corrective actions.

Example:

Life Insurance Premium rates:

* A life Insurance Company has base premium of RS.50 for all ages. Based on age group, an additional monthly premium has to be paid that is listed

| Age group | Additional Premium |
|---|---|
| Under 35 | RS.2 |
| 35-39 | RS.3 |
| 60+ | RS.6 |

Equivalence classes for the life Insurance premium

| S.NO | Equivalence Partitions | Type of Input | Test data | Expected results |
|---|---|---|---|---|
| 1. | Age below 35 | valid | 26, 12 | Monthly premium = Rs 50+2<br>= RS.52 |
| 2 | Age 35-39 | valid | 37 | Monthly premium = RS.50+3<br>= RS.53 |
| 3. | Age above 60 | valid | 65, 90 | Monthly premium = RS.50+6<br>= RS.56 |
| 4 | Negative age | Invalid | -23 | warning Message - Invalid input |
| 5 | Age as 0 | Invalid | 0 | warning Message - Invalid input |

## ii) Boundary value Analysis (BVA)

* Boundary value Analysis (BVA) is used to find the errors at boundaries of input domain rather than finding those errors in the Center of input

* Test both valid boundaries and Invalid boundaries

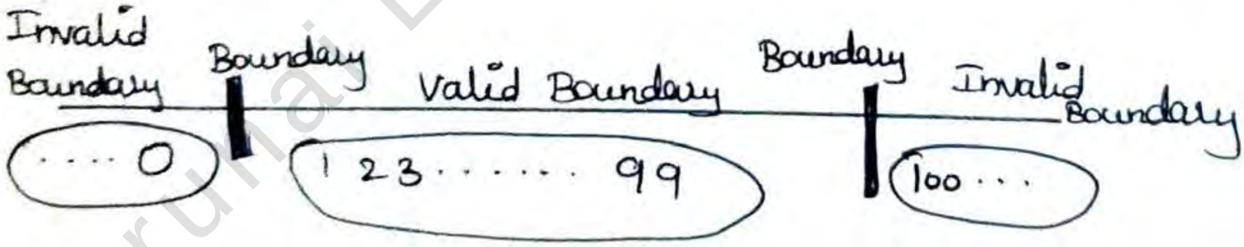
* It is based on boundaries between partitions

* It is a part of functionality testing but test engineers are giving special treatment to input domains of objects

* It is also called "INPUT DOMAIN TESTING".

Example:

* consider a printer that has an input option of the number of copies to be made from 1 to 99.



In the Example valid / Invalid boundary values are

Min = 1    Max = 99

So the solution is

| | | | | | |
|---|---|---|---|---|---|
| min | →1 | Pass | Max+1 | →100 | fail |
| Min-1 | →0 | Fail | Max-1 | →98 | Pass |
| Min+1 | →2 | Pass | | | |
| Max | →99 | Pass | | | |

* Boundary value analysis is Chosen to detect the error.

* Most of the error/defect occurs on boundary, not in the Center of the input domain

* It allows the Selection of set of test cases and those test cases exercises the boundary values.

* It designs test cases at the edge of input domain,

* It also concentrates on the output domain

Guidelines for BVA:

1. BVA is to select input variables values for minimum, above minimum, normal value, below maximum, maximum

2. Failures occurs rarely as the result of the Simultaneous occurrence of 2 or more faults

3. Variables are program dependent, language dependent, bounded discrete, unbounded discrete & logical variables.

4. Programs written in non-strongly typed language are more appropriate candidates for BVA

5. Boundary in equalities of 'n' input variables define a n-dimensional input space

3) 16M With Examples Explain the following black box techniques to testing. [Apr/may - 17]

i) Requirements based testing

ii) Positive and Negative testing

iii) State based testing

iv) User documentation and compatibility.

## i) Requirements Based Testing:

* Requirements-based testing is a testing approach in which test cases, conditions and data are derived from requirements.

* It includes functional tests and also non functional attributes such as Performance, reliability or usability.

It is a 12 step process

1. validate requirements against objectives
2. Apply scenarios against requirements
3. Perform initial ambiguity review
4. Perform domain Expert review
5. Create Cause - effect graph
6. Logical Consistency check by RBT
7. Review of test cases by Specification writers
8. Review of test Cases by Users

9. Review of test Cases by developers

10. walk test cases through design

11. walk test cases through code

12. Execute test Cases against code

Requirements types:

1. Explicit requirements - It is stated and documented as part of the requirements specification

2. Implied or Implicit requirements - This is not documented but assumed to be incorporated in the system.

* The precondition for requirements testing is a detailed review of the requirements specification. It checks

↳ Consistency, Correctness, Completeness, Testability Clarity of requirements etc.

* All explicit requirements and implied requirements are collected and documented as "Test Requirement Specification" (TRS)

* Eg sample requirements specification for lock and key system

Sample requirements specification for lock & key system

| S.NO | Requirement Identifier | Description | Priority (High, Medium, low) |
|---|---|---|---|
| 1. | BR-01 | Inserting the key numbered 123-456 and turning it clockwise should facilitate locking | H |
| 2. | BR.02 | Inserting the key numbered 123-456 and turning it anticlockwise should facilitate unlocking | H |
| 3. | BR-04 | No other object can be used to lock | M |
| 4 | BR-05 | No other object can be used to unlock | M |
| 5 | BR-07 | The lock & key must be made of metal & must weigh approximately 150 grams | L |
| 6 | BR-08 | Lock and unlock directions should be changeable for usability of left-handers | L |

* Requirements are traced by Requirements Traceability Matrix (RTM). An RTM traces all the requirements from their genesis through design, development and testing.

* The "test conditions" Column lists the different ways of testing the requirements. These conditions can be grouped together to form a single test case.

* The "test case ID's" Column can be used to complete the mapping between test cases & the requirements

* RTM helps in identifying the relationship between the requirements and test cases.

1) one-to-one: For each requirement there is one test case
2) one to many: For each requirements there are many test case
3) Many to one: A set of requirements can be tested by one test cases
4) Many to many: Many requirements can be tested by many test cases
5) One to none: The requirements can have no test case

* The "phase of testing" column is used multiple Phases of testing - Unit, Component, integration and System testing

Sample requirements Traceability Matrix

| SNO | Requirement Identifier | Description | Priority (H, M, L) | Test Conditions | Test Case IDS | Phase Of Te |
|---|---|---|---|---|---|---|
| 1. | BR-01 | Inserting the key numbered 123-456 and turning it clockwise should facilitate locking | H | Use key 123-456 | Lock 001 | Unit, Component |
| 2 | BR-04 | No other object can be used to lock | M | use key 789-001<br>use hairpin<br>use screw drivers | Lock -005,<br>Lock -006,<br>Lock -007 | Integration |
| 3. | BR-07 | Lock and key must be made of metal and must weigh approximately 150 grams | L | Use weighing | Lock -012 | System |

Role of RTM:

1) The RTM enables testers to prioritize the test cases execution to find the defects

2) Test conditions can be grouped to create test Cases or can be represented as unique test cases.

3) To find the adequate test cases/ high priority requirements

Metrics that can be collected or inferred from RTM matrix are

1) Requirements addressed prioritywise

2) Number of test cases requirement wise

3) Total number of test cases prepared.

After the test cases are executed, the test results can be used to collect metrics such as

↳ Total number of test cases passed

↳ Total number of test cases failed

↳ Total number of defects in requirements

↳ Number of requirements completed

↳ Number of requirements Pending.

ii) Positive and Negative Testing:

* The purpose of positive Testing is to prove that the product works as per specification and expectations.

* A product delivering an error when it is expected to given an error is also a part of positive testing

Example of positive Test cases

| Requirements NO | Input 1 | Input 2 | Current State | Expected outcome |
|---|---|---|---|---|
| BR-01 | key 123-456 | Turn clockwise | Unlocked | Locked |
| BR-01 | key 123-456 | Turn clockwise | Locked | No change |
| BR-02 | key 123-456 | Turn anticlockwise | unlocked | No change |
| BR-03 | key 123-456 | Turn anticlockwise | Locked | Un lock |
| BR-04 | Hairpin | Turn clockwise | Locked | No change |

* Negative testing is done to show that the Product does not fail when an unexpected input is given.

*The Purpose of negative testing is to try and break the system. Negative testing covers scenarios for which the product is not designed and coded.

* Important for the tester to know the negative situations that may occur at the end-user level so that the application can be tested and made fool proof

Negative Test cases

| S.No | Input1 | Input2 | Current State | Expected output |
|---|---|---|---|---|
| 1. | Some other lock's key | Turn Clockwise | Lock | Lock |
| 2 | some other lock's key | Turn clockwise | unlock | Unlock |
| 3. | Thin Piece of wire | Turn anticlockwise | Unlock | Unlock |
| 4 | Hit with a stone | | Lock | Lock |

* The difference between positive testing and negative testing is in the Coverge.

* For positive testing if all documented requirements and test conditions are covered, then coverage can be considered to be 100 percent.

* For negative testing requires a high degree of creativity among the testers to cover as many "unknowns" as possible to avoid failure at a customer site

### iii) State based Testing (or) Graph Based Testing:

* It is black box testing techniques, in which output's are triggered by changes to the input conditions or changes to 'State' of the system.

* In other words, tests are designed to execute valid and invalid state transitions.

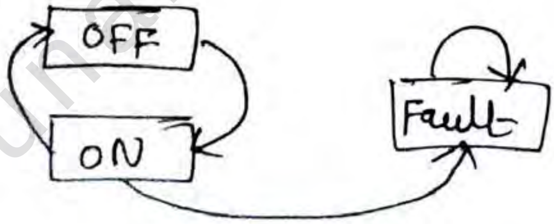
* State Based Testing Uses:

↳ when we have sequence of events that occur and associated conditions that apply to those events

↳ when the proper handling of a particular event depends on the events and conditions that have occurred in the past.

↳ It is used for real time systems with various states and transitions involved.

Eg: A system transition is represented



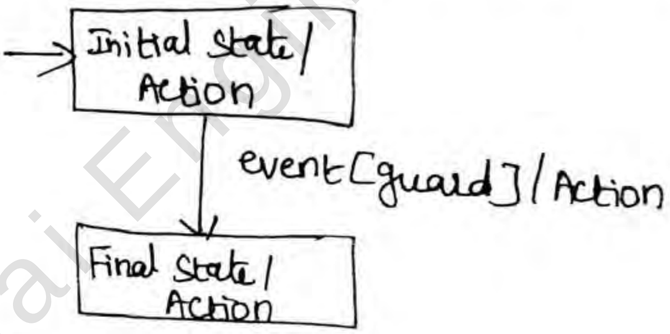
State & Transition Scenario

| Tests | Test 1 | Test 2 | Test 3 |
|---|---|---|---|
| Start state | off | on | on |
| Input | Switch ON | Switch off | switch off |
| output | Light ON | Light off | Fault |
| Finish state | ON | OFF | ON |

State machine:

* Implementation independent Specification (model) of the dynamic behavior of the system.

1. State - Abstract Situation in the life cycle of a System entity
2. Event - A Particular input (A message or method call)
3. Action - The result, output or operation that follows an event
4. Transition - An allowable two state sequence (ie) a change of state ("firing") caused by event
5. Guard - Predicate Expression associated with an event Stating a Boolean restriction for a transition to fire.



State Machine

State Transition Diagram:

* It is graphical representation of a state machine
* It is represented by a state graph having a finite number of states & a finite number of transitions between states
* It is also called "Graph based Testing"
* It is useful for both procedural & object oriented implementation

* State or graph based testing is very useful language processor (Eg compiler) testing, workflow modeling, Dataflow modelity etc.

Example 1:

Application in which a number is validated

Rules:

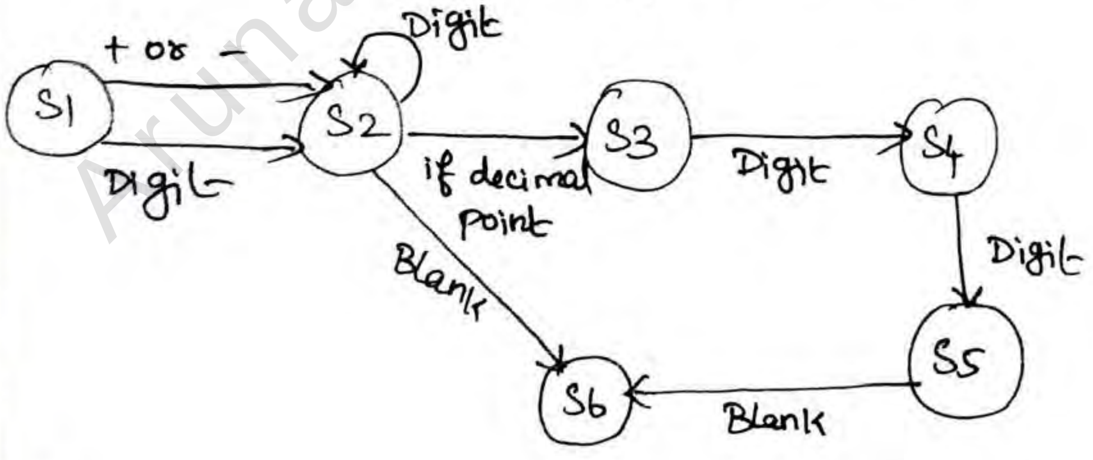
1) A number can start with an optional sign

2) The optional sign can be followed by any number of digits

3) The digits can be optionally followed by a decimal point, represented by period.

4) If there is a decimal point that there should be two digits after the decimal

5) Any number- whether or not, it has a decimal point should be terminated by the blank.



State Transition Diagram

## State Transition Table

| Input | Current state | Next state |
|---|---|---|
| Digit | 1 | 2 |
| + | 1 | 2 |
| - | 1 | 2 |
| Digit | 2 | 2 |
| Blank | 2 | 6 |
| Decimal Point | 2 | 3 |
| Digit | 3 | 4 |
| Digit | 4 | 5 |
| Blank | 5 | 6 |

* The state transition table can be used to derive test cases to test valid and invalid numbers.

* valid test cases can be generated by

1) Start from the start state (State 1)
2) Choose a path that leads to the next state
3) If invalid input is encountered in a given state, generate an error condition test case.
4) Repeat the process till you reach the final state.

iv) User documentation and Compatibility:

* User documentation covers all the manuals, user guides, installation guides. set up guides, Read me files, software release notes, online help

* User documentation testing should have two objectives

1) TO check if what is stated in the document is available in the product

2) TO check if what is there in the product is explained correctly in the document

* when a product is upgraded, the corresponding product documentation should also get updated as necessary to reflect any changes that may affect a user.

* TO concentrates on the specification given in the document is perfectly matching with product behavior or not

Benefits of User documentation Testing

1) Removes Uncertainties

2) offer good training materials to freshers.

3) Good marketing Strategy

4) Better customer Satisfication

5) Easily ensure the problem during review.

## Compatibility Testing:

* It is also called "portability Testing".

* It is a non functional testing conducted on the application to evaluate the application's compatibility with in different environments.

* During this test, test engineers validates that whether our application build run on customer expected platform or not?

Two types of compatibility testing:

1. Forward compatibility
2. Backward compatibility

## Compatibility Testing technique:

1) Horizontal combination:

* In this technique, all the value of parameters to execute test cases are combined into row in the compatibility matrix

* Machines are set to every row and the set of product characteristics are tested.

2) Intelligent Sampling:

* In this technique, combinations of infrastructure parameters, set of features are combines and tested.

* Intelligent Sample are generated based on the data collected on the set of dependencies of the product with Parameters.

* If the product result are less dependent on a set of Parameters then they are taken out from the collection of intelligent samples.

*Backward Compatibility Testing:

* The testing that ensures the current version of the product continues to work with the older versions of the same product is called backward compatibility testing.

* The Product parameters required for the backward compatibility matrix and are tested.

*Forward compatibility testing:

* There are some provisions for the product to work with later versions of the product and other infrastructure components keeping future requirements in mind.

Briefly Explain the weyuker's eleven axioms that allow testers to evaluate test adequacy Criteria [APR/May-17] [May-Jule-16]

Axioms:

It is a rule it helps to tester in order to

1. Recognize adequacy criteria
2. Properties of selected test data adequacy Criteria
3. It help to choose a suitable criterion
4. Stimulate thought for the development of new criteria.

Axioms are based on the following set of assumptions

1. Programs are written in a structured programming language.
2. Programs are single entry/exit
3. All input statements appear at the beginning of the program.
4. All output statements appear at the end of the program

Weyuker axioms/Properties:

1. Applicability property

* For every program there exists an adequate test set

2. No exhaustive applicability property

For a program P and a test set T, P is adequately tested by the test set T, and T is not an exhaustive test set

3. Monotonicity property:

If a test set T is adequate for program P and if T is equal to or a subset of T' then T' is adequate for program P.

4. In adequate empty set

An empty test set is not an adequate test for any program.

5. Anti extensionality property:

These are programs P and Q such that P is equivalent to Q and T is adequate for P, but T is not adequate for Q

6. General multiple Change property:

These are programs P and Q that have the same shape and there is a test set T such that T is adequate for P, but is not adequate for Q.

7. Anti decomposition property:

There is a program P and a component Q such that T is adequate for P, T' is a set of vectors of values that variables can assume on entrance to Q for some t in T, and T' is not adequate for Q.

8. AntiComposition property:

* These are programs P and Q, and test set T such that T is adequate for P, and the set of vectors of values that variables can assume on entrance to Q for inputs in T is adequate for Q, but T is not adequate for P.

9. Renaming Property:

* If P is a renaming of Q, then T is adequa for P only if T is adequate for Q.

* A program P is a renaming of Q if P is identical to Q expect for the fact that all instanc of an identifier, let us say "a" in Q have been replaced in P by an identifier.

* Let us say b where "b" does not occur in Q, or if there is a set of such renamed identifiers.

10. Complexity Property:

* For every n, there is a program P such that P is adequately tested by a size n test set, but not by any size n-1 test set

11. Statement Coverage Property

* If the test set T is adequate for P, then T causes every executable statement of P to be executed.

5)
a Explain the various white box techniques with Suitable test cases

b Discuss in detail about code Coverage testing

[Nov/Dec-16]

white box testing:

* white box testing is a way of testing the external functionality of the code by Examining and testing the program code that realizes the external functionality

* This is also known as clear box, or glass box or open box testing.

* white box testing takes into account the program code, code Structure and internal design flow.

* A number of defects come about because of incorrect translation of requirements and design into program code

* Some other defects are created by programming errors and Programming language unconventional behaviour.

## Testing Techniques:

1. Statement coverage- This technique is aim to exercising all programming Statement with minimal test.

2. Path Coverage- Every statement in program is correctly partipate in the program or not.

3. Program technique Coverage - It checks CPU cycles during execution, less number of memory cycles etc

4. Condition coverage

5. Loop coverage

6. Function coverage.

* It is a coding level testing strategy.

* It ensures all the statements & conditions have been executed at least once

* It verify that the software design is valid and also whether it was build according to the specified design or not.

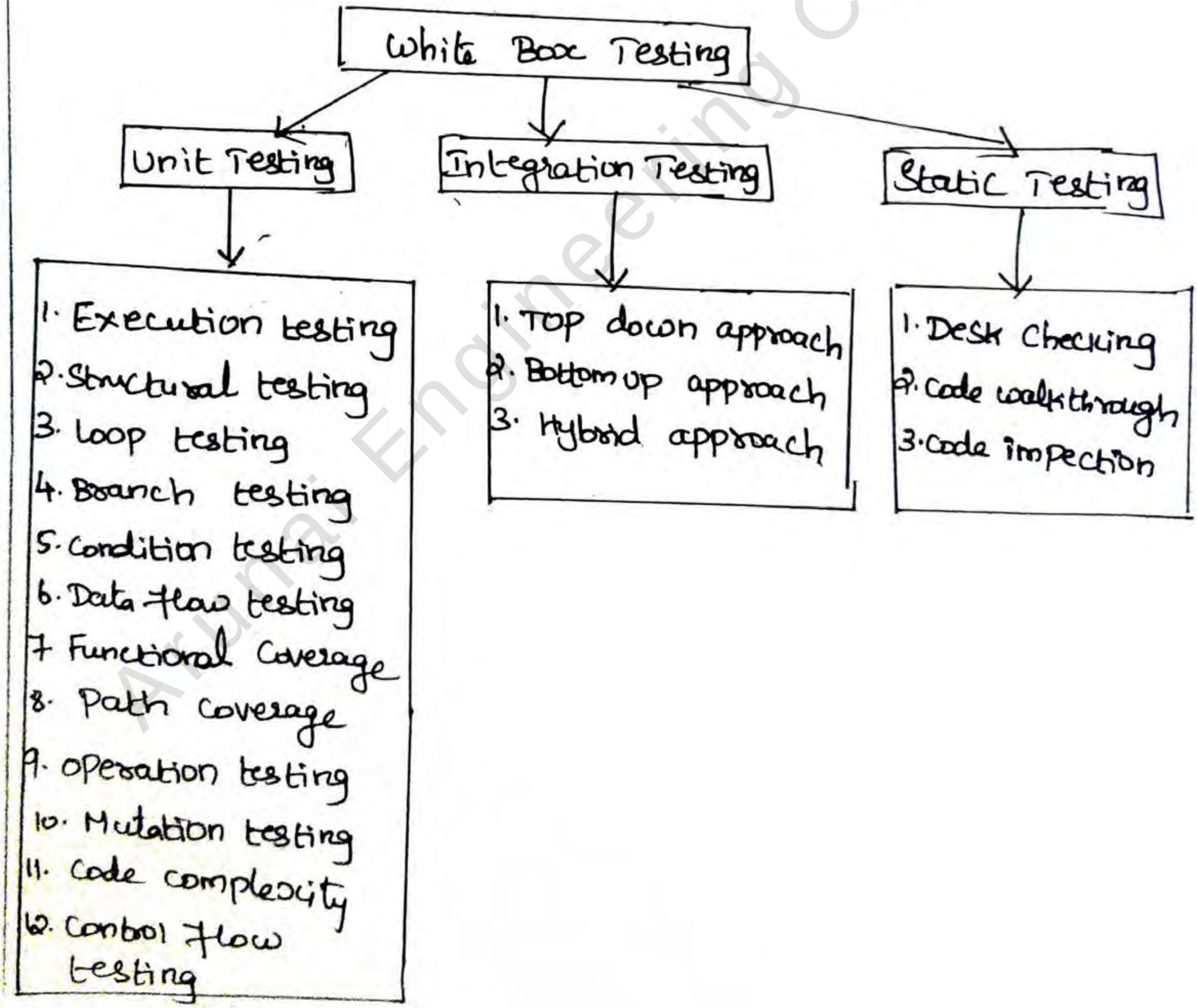
* Software developers verifies interms of internal logic with the help of test case design.

* The tester's aim is to find it all the logical + data elements in the software unit are functionating properly or not

* It helps to test small components/ modules

* This strategy is implemented for design + code base control

## White Box Testing Technique



b) Code Coverage Testing: (nov/Dec 17)

* Logic based (white booc based) test design and use of test data adequancy (criteria coverage) concepts provide 2 major payoffs for the tester

1. Quantative coverage goal propose
2. commercial tool support is readily available to facilitate the tester's work.

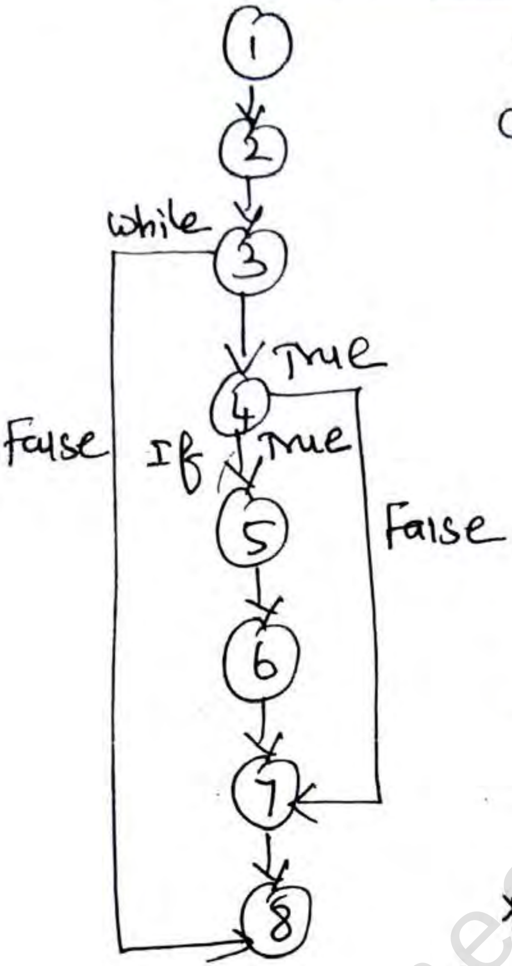
* when the goal is to satisfy the statement adequacy criterion, then a set of test cases can be developed.

* If the module gets executed, all of the statement in the module are executed at least once.

Code Sample with branch & loop

```
Sum (a, num, sum)
 Sum =0
  int i=1
   while (i<=num)
     if (a[i]>0)
        Sum= sum+a[i]
     end if
        i= i+1
     end while
   end sum
```

## Control flow graph



* A tester to prepare test cases which checks the nodes from 1 to 8 in the flow graph.

* when test reaches this goal, the statement adequate criterion is satisfied by the test data and also other logic structures associated with the adequate criterion.

* Test cases should be designed to get 100% coverage.

* Every decision element in the code executes with all possible outcomes atleast once.

* Complete decision coverage goal is the stronger coverage goal than statement coverage

* If the statement coverage is so weak that is not considered to be very useful for revealing defects

* Input values must conform the execution of true/false possiblities for the decision in line 3 and 4

Possible Test cases

| Decision/ Branch | value of variable | value of predicate | Test case: value of a, num |
|---|---|---|---|
| | | | a = 1, -45, 3<br>num = 3 |
| while | 1<br>4 | True<br>False | |
| if | 1<br>2 | True<br>False | |

The test should fulfill both

* Branch adequacy Criterion

* Statement adequacy Criterion

Consider a statement

if (a<min and b>max and (not int c))

This statement has 3 conditions are predicate

1. a<min 2. b>max 3. not in c

Decision Coverage:

It must checks all the possible condition atleast once for all the branch/loop predicates.

Condition Coverage:

* It must needs atleast one execution of all possible conditions and combinations of decision.

* Coverage Criterion can be arranged in a hierarchy of Strengths from weakest to Strongest

1. Statement
2. Decision
3. Decision/ Condition

Example:

```
If (age <65 and married ==true)
    do x
    do y...
  else
    do z
```

Condition 1: Age loss than 65

Condition 2: Married is true

Test cases for simple decision Coverage

| Value for age | Value for married | Decision outcome (compound predicate) | Test cases ID |
|---|---|---|---|
| 30 | True | True | 1 |
| 75 | True | False | 2 |

Test cases for condition coverage

| Value for age | Value for married | Condition1 outcome | Condition2 outcome | Test cases ID |
|---|---|---|---|---|
| 75 | True | False | True | 2 |
| 30 | False | True | False | 3 |

## Test cases for Decision Condition Coverage

| Value for age | Value for married | Condition 1 outcome | Condition 2 outcome | Decision outcome (compound Predicate) | Test cases ID |
|---|---|---|---|---|---|
| 30 | True | True | True | True | 1 |
| 75 | True | False | True | False | 2 |
| 30 | False | True | False | False | 3 |

* The large the number of test cases that must be developed to insure complete coverage.

* "Multiple condition coverage" or "Multiple decision conditions", the complexity of test case design increases with the strength of the coverage criterion.

* The test decide the criterion based on the

↳ code

↳ reliability requirement

↳ Available resources.

# IT8076-SOFTWARE TESTING

# UNIT-3
# LEVELS OF TESTING

# 2 MARKS & 16 MARKS
# WITH ANSWERS

## Unit - III

### Part - A

1. List out types of system Testing. [Nov/Dec-16]

- * Functional testing
- * Performance testing
- * Stress testing
- * Configuration testing
- * security testing
- * Recovery testing

2. Compare and contrast Alpha and Beta Testing [Nov/Dec-16]

| Alpha Testing | Beta Testing |
|---|---|
| 1. It is always performed by the developers at the software development site | 1. It is always Performed by the Customer at their own site |
| 2. It is done by testing team | 2. It is done by testing team and customers |
| 3. It is always performed in virtual environment | 3. It is performed in real-time environment |
| 4. It comes white box and black box testing | 4. It is comes only the black box testing. |

3. Why is it important to design test harness for testing? [Apr/may-17] [May/june-16] (nov/Dec 19,17)

* "The auxiliary code developed to support testing of units and components is called a test harness".

* The harness consists of drivers that can the target code and stubs that represents modules its calls.

* The tester is considering a stand-alone function/procedure/class, rather than a complete system, code will be needed to call the target unit and also to represents modules that are called by the target unit. This code is called the test harness.

4 What are the issues in testing object oriented system (nov/Dec 19) [May/june -16]

1. Adequately Testing Classes
2. observation of object states and state changes
3. The retesting of classes - I and II

5. Define regression testing. [May/june-16] (nov/Dec 18

* Regressing testing is used to check for defects propagated to other modules by changes made to existing program.

* Regressing testing is used to reduce the Side effects of the changes.

6 What is alpha Testing? [May/june-16]

* Alpha test take place at the developer's Site. A cross-section of potential users and members of the developer's organization are invited to use the software

* Developers observe the users and note problems

7. State the purpose of Defect Bash testing. [Apr/May-15]

* Defect bash is an adhoc testing where people does the different roles in an organization test the product together at the same time.

* It is fully based on individual decision and creativity.

* Defect bash/adhoc testing brings plenty of good practices that are popular testing industry.

8. write the major activities followed in internationalization Testing. [Apr/May-15]

* Enabling Testing
* Locale testing
* Internationalization testing & validation
* Fake language testing
* Language testing
* Localization testing

9. write the components of test plan [Nov/Dec-14]

* Test plan identifier
* Introduction
* Items to be tested
* Features to be tested
* Approach
* pass/fail criteria
* Testing tasks
* Test environment

0. List out the various types of System testing [Nov/Dec-14]

i) Functional Testing ii) performance testing

iii) Stress Testing iv) security testing

v) Recovery Testing vi) configuration testing

M

Explain the different Integration testing Strategies for Procedures and functions with Suitable diagrams. [May/June-16] [Nov/Dec-16]

What is Integration Testing? Explain with Examples the different types of Integration testing.

Discuss in detail about different types of integrating testing [APR/MAY-15]

## Integration Testing:-

* Integration is defined as the set of interactions among components.

* Testing the interaction between the modules and interaction with other systems externally is called integration testing.

* "At the integration level several components are tested as a group and the tester investigates components interactions".

Goals:

↳ Integration test for procedural code has two major goals.

1. To detects defects that occur on the interfaces of units

2. To assemble the individual units into working sub systems and finally a complete

system that is ready for system test.

* Integrating testing works best as an iterative Process in Procedural -oriented system. one unit at a time is integrated into a set of Previously integrated modules which have passed a set of integration tests.

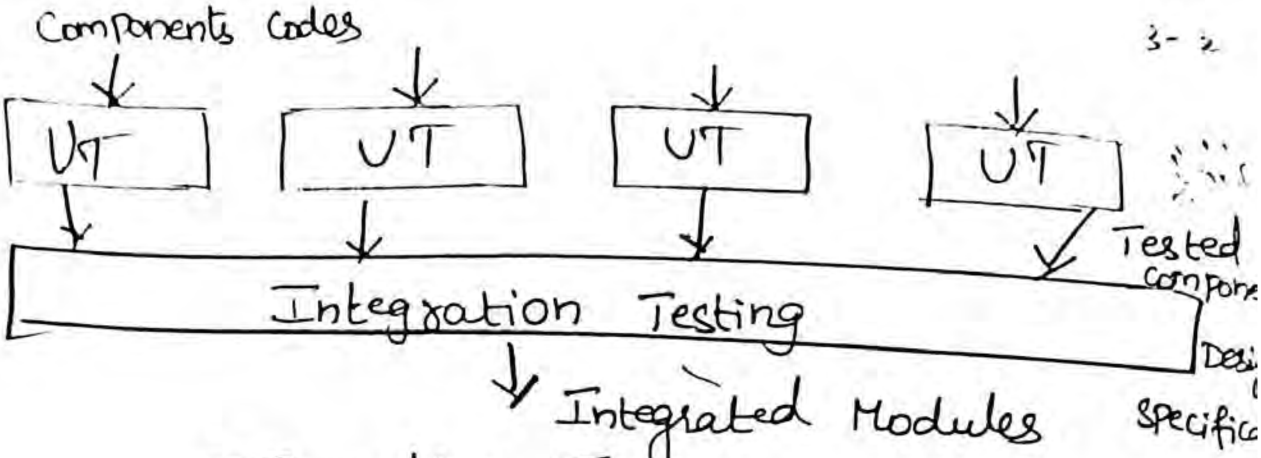
* The interfaces and functionality of the new unit in combination with the previously integrated units is tested.

Integration strategies for Procedures and Functions:

For procedural and functional oriented system these are major integration strategies.

1) Top-down integration
2) Bottom-up integration
3) Bi-directional integration
4) System integration.

Top-down and Bottom-up integration strategies only one module at a time is added to the growing subsystem



Integration Testing

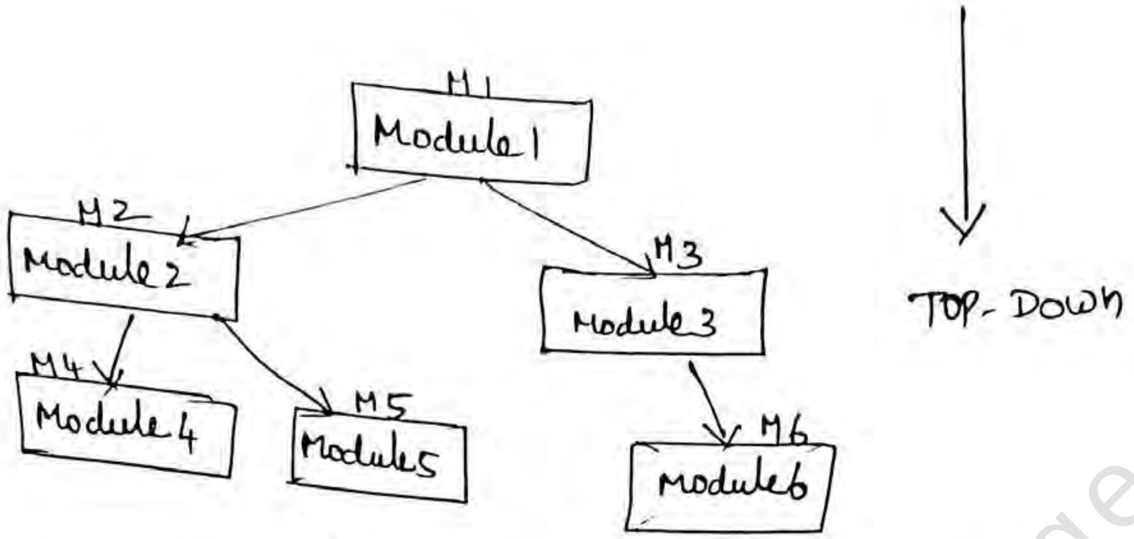
1) Top-Down Integration:

* Integration testing involves testing the top most component interface with other components in same order as navigate from top to bottom till cover all the components

* If a set of components and their related interfaces can deliver functionality without expecting the presence of other components or with minimal interface requirements in the software/product then the set of components and their related interfaces is called as a "Sub-systems".

* Each sub-system in a product can work independently with or without other sub-systems

* This makes the integration testing easier and focus on required interfaces rather than getting worried about each & Every combination of components.



* Begin top-down integration with module1 Create two Stubs to represents Module 2 and Module3. when the tests are passed, then replace the two Stubs by the actual Module one at a time.

* one can traverse the Structure Chart and integrate the modules in a depth (or) Breadth-first manner.

Depth-first Manner (approach):

↳ order of integration M1, M2, M4, M5, M3, M6

Breadth-first approach:

↳ order of integration M1, M2, M3, M4, M5, M6

Advantages:

↳ The upper-level modules are tested early in integration. If they are complex and need to be redesigned there will be more time to do so.

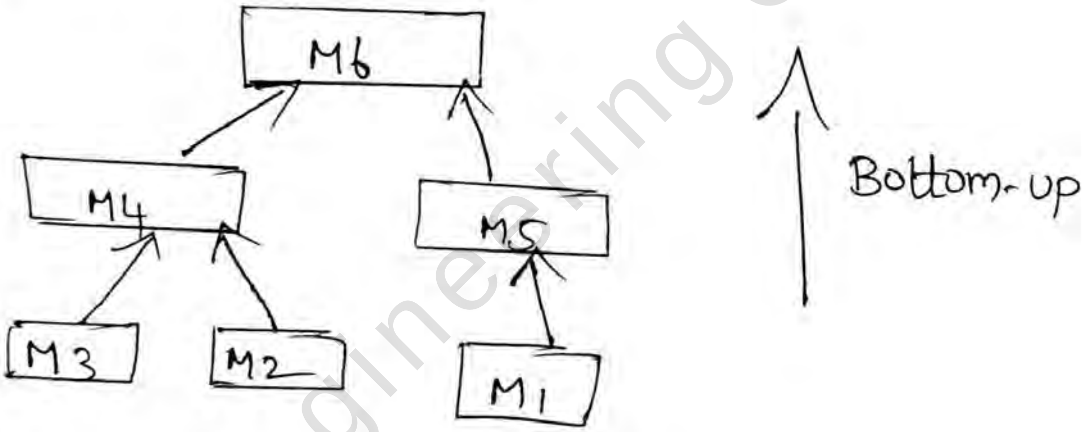
Disadvantages:

* Top-down integration requires the development of complex Stubs to drive significant data upward.

* Basic functionality is tested at the end of cycle.

2) Bottom-up Integration:

* Bottom-up Integration is just the opposite of top-down integration.



* Bottom-up integration of the modules begins with testing the lowest modules, those at the bottom of the structure chart (M3, M2, M1, M4, M5, M6)

* Components or systems are substituted by drivers. Developer used a temporary program, instead of main module construction, It is called "Drivers"

* Drivers are needed to test these modules.

The next step is to integrate modules on the next upper levels of the structure chart whose subordinate modules have already been tested.

eg: M3 and M2 are tested, then select M4 and integrate it with M3 and M2. the actual M4 replaces the drivers for these modules.

* Main difference between Top-down & Bottom-up is the arrows from top to bottom (downward-pointing arrows) indicate interaction or control flow.

* The arrows from bottom to top (upward-Pointing arrows) indicate integration approach or integration path.

* TOP-down integration approach is best suited for waterfall & V models.

* Bottom-up integration approach for the iterative and agile methodologies.

* In Practical Scenario the approach selected for integration depends more on the design and architecture of a product and on associated priorities.

Advantage:

* The low-level modules are usually well tested early in the integration process. This is important if these modules are candidates for reuse.

Disadvantage:

* It is required to create the test drivers for modules at all levels, except the top control

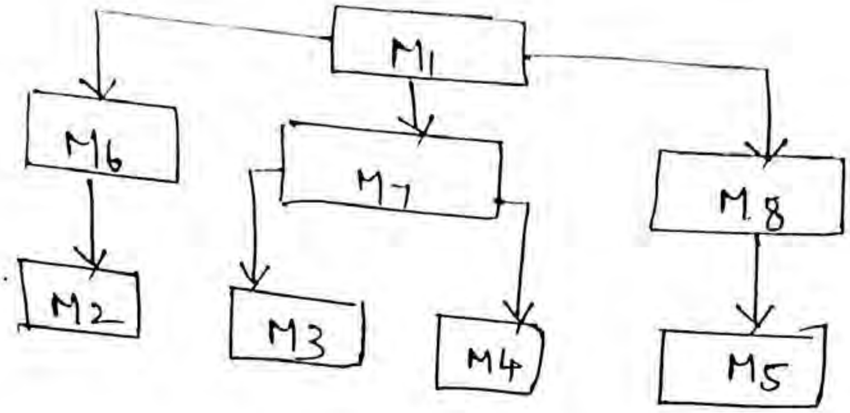
3) Bi-Directional Integration:

* Bi-directional integration is a combination of the top-down and bottom-up integration approach used together to derive integration steps.

* Let us assume the software components become available in the order mentioned by the component numbers.

* The individual components 1,2,3,4 & 5 are tested separately and bi-directional integration is performed initially with the use of stubs and drivers.

* Drivers are used to provide upstream connectivity while Stubs provide downstream connectivity



Bi-directional Integration

* A driver is a function which redirects the requests to some other component and stubs simulate the behavior of a missing component.

* After the functionality of these integrated components are tested, the drivers and stubs are discarded.

* once components 6, 7 and 8 become available the integration methodology then focuses only on those components, as these are the components which need focus and are new. This approach is also called "sandwich integration".

Steps for integration using sandwich Testing

| Step | Interfaces tested |
|---|---|
| 1 | 6-2 |
| 2 | 7-3-4 |
| 3 | 8-5 |
| 4 | (1-6-2)-(1-7-3-4)-(1-8-5) |

eg 1-3 use a bottom-up integration approach and step 4 use a top-down integration approach

Advantage:

* Easily Combine modules (Sub modules and main module)

* It helps to developers effectively.

Disadvantage:

* It is temporary one & unstructured one.

4) System Integration:

* System integration means that all the components of the system are integrated and tested as a single Unit.

* Integration testing, which is testing of interfaces, can be divided into two types.

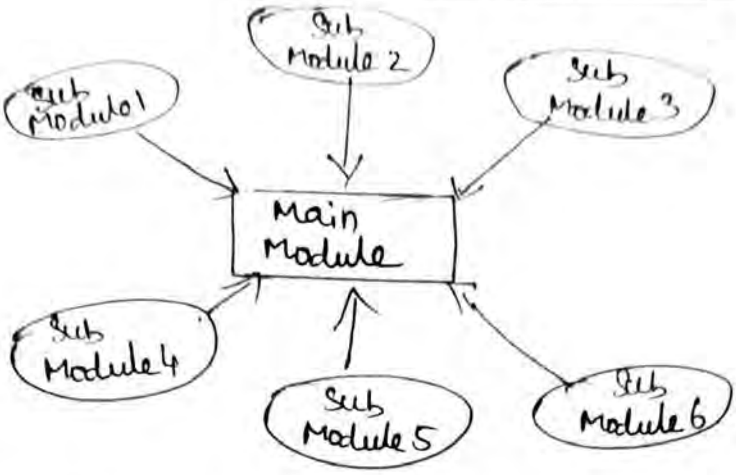
↳ Components or sub-system integration

↳ Final integration testing or system integration

* Instead of integrating component by component and testing, this approach waits till all components arrive and one around of integration testing is done. This approach is also called "Big. bang integration"

* It reduces testing affort and removes duplication in testing.

Big-Bang Approach



* System integration using the big bang approach is well suited in a product development Scenario where the majority components are already available and stable and very few components get added or modified.

* In this case, instead of testing components interface one by one, it make sense to integrate all the components at one go and test once, Saving effort and time for the multi-step Component integration.

Advantage:

* Saves time and effort

Disadvantage:

* Difficult to trace the case of failure.

What is Unit testing ? Explain with an Example the process of designing the Unit tests, running the Unit tests and recording results. [May/june-2016] (nov/Dec 19,18)
(apr/may 18

## Unit Test:

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

* A Unit is the smallest possible testable software Components.

* A unit in a typical procedure-oriented software system.

a) Performs a single Cohesive function

b) Can be compiled Separately

c) Is a task in a work breakdown Structure

d) Contains code that can fit on a single Page (or) Screen.

* A unit is said to be procedure (or) function, by written | using a Procedural Programming languages.

Some Components Suitable for Unit Test:

Procedures and Functions

Classes/objects and Methods

Procedure Sized reusable components (Small - Sized COTS componen or components from an in-house reuse library

* The principal goal for Unit Testing is insure that each individual software unit is functioning according to its specification.

* To Prepare for Unit test the developer/tester must perform several tasks. They are

1) plan the general approach to Unit testing.
2) Design the test cases, and test procedures.
3) Define relationships between the tests.
4) Prepare the auxiliary code necessary for Unit test.

## Unit Test Planning:

* It define "what to test", "How to test" "when to test" and "who to test"

* Test plan is a project level document

* It may be prepared as a component of the

i) Master test plan or
ii) Stand - alone plan

* It Should be developed in conjunction with the master test plan and the Project plan for each project.

* Documents that provide inputs for the unit test plan are

a) project plan b) Requirement

c) specification d) Design Documents that describe the target unit

* set of development Phases for Unit test planning are

Phase1: Describe Unit test approach and risks

phase2: Identify unit features to be tested

phase3: Add levels of detail to the plan.

* In each Phase a set of activities is assigned based on IEEE Unit test standard.

Phase1: Describe Unit test approach and risks

The general approach to unit testing is outlined. The test planner:

a) Identifies test risks

b) Describes techniques to be used for designing the test cases for the units

c) Describes techniques to be used or data validation and recording of test results

ii) Describes the requirements for test harness and other software that interfaces with the units to be tested.

* The planner identifies completeness requirements such as states, functionality control and data flow patterns.

* The planner also identifies termination conditions for the unit tests. This includes coverage requirements and special cases.

* Special cases may result in abnormal termination of unit test. Strategies for handling these special cases need to be documented.

* The planner estimates resources needed for unit test, such as hardware, software and staff and develops a tentative schedule under the constraints identified at that time.

Phase 2: Identify Unit features to be tested:

* This phase requires information from the unit specification and detailed design description.

* The planner determines which features of each unit will be tested.

Eg: Functions, Performance requirements, states, state transitions control structures, messages and data flow patterns.

* If Some features will not be covered by the tests, they should be mentioned and the risks of not testing them be assesed.

* Input/output characteristics associated with each unit should also be identified, such as variables with an allowed ranges of values and Performance at a Certain level.

Phase 3: Add levels of detail to the plan

* The planner refines the plan as produced in the previous two phases.

* The Planner adds new details to the approach, resource and scheduling portions of the unit test plan.

Eg: Existing test Cases that can be reused for this Project can be identified in this Phase.

* Unit availability and Integration scheduling information Should be included in the revised Version of the test plan

* The Planner must be Sure to include a description of how test results will be recorded.

## Designing the Unit Tests:

* Part of the preparation work for unit test involves unit test design. It is importan to Specify

a) The test cases

b) The test procedures

Test case data should be tabularized for ease of Use and reuse.

Test Case Specification notation:

*Arrange the Components of a test case into a semantic network with parts, object-ID, Test-Case-ID, purpose and List-of-Test-case-steps.

* Test design specification includes lists of relevant states, messages, exceptions and interrupts.

* Developer| tester Should describe the relationship between the tests.

* Test suites can be defined that bind related tests together as a group. All of this test design information is attached to the unit test plan.

* Test cases, test procedures and test Suites may be reused from past Project if it has been careful to store, so that they are easily retrievable & reusable.

* Test case design at the Unit level can be based on use of the black and white box test strategies.

* Both of these approaches are useful for designing test cases for functions and procedures.

The test harness:

* In addition to developing the test cases, supporting code must be developed to exercise each unit and to connect it to the outside world.

* The tester is considering a stand-alone function/procedure/class, rather than a complete system, code will be needed to call the target unit and also to represent modules that are called by the the target unit. This code is called test harness.

* The auxiliary code developed to support testing of units and components is called a test harness.

* The harness consists of drivers that call the target code and stubs that represent modules it calls

* The development of drivers and stubs requires testing resources.

* The drivers and stubs must be tested themselves to insure they are working properly and they are reusable for subsequent releases of the software.

Eg A driver could have the following options and combinations of options.

a) call the target Unit

b) do 1, and Pass inputs parameters from the table

c) do 1, 2 and display Parameters.

d) do 1, 2, 3 and display results.

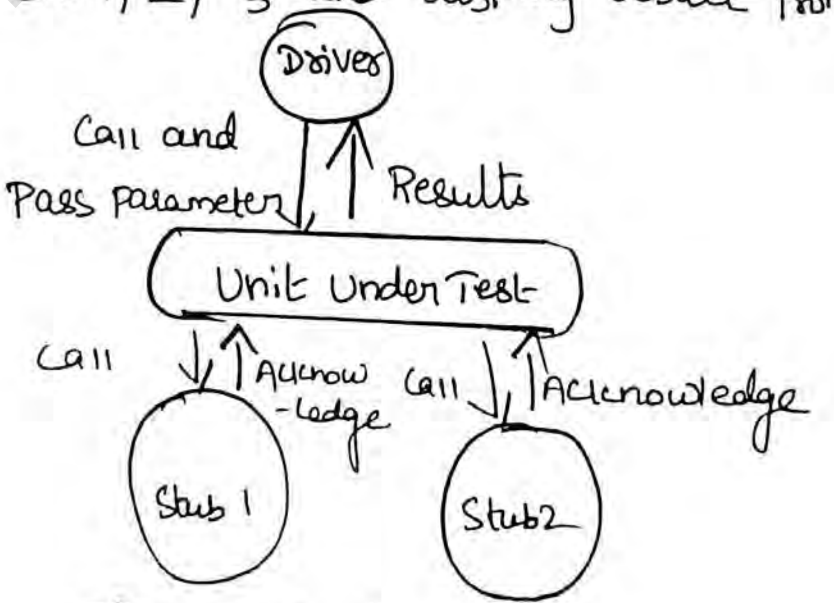
The stubs could also exhibit different levels of functionality. A stub could:

a) display a message that it has been called by the target unit

b) do 1 and display any input Parameters passed from the target unit

c) do 1, 2 and pass back a result from a table

d) do 1, 2, 3 and display result from table



The Test harness

## Running the Unit test and Recording Results.

Unit tests can begin when

a) The Units becomes available from the developers

b) The test cases have been designed and reviewed.

c) The test harness and any other supplemental Supporting tools are available.

* The testers then proceed to run the tests and record results

* Test logs can be used to record the results of specific tests.

* The Status of the test effort for a unit, and a summary of the test results could be recorded in a simple format.

Summary work sheet for unit test results

Unit Test worksheet

Unit Name: ____________________

Unit Identifier: ____________________

Tester: ____________________

Date : ____________________

| Test case ID | Status (run/not run) | Summary of results | Pass/fail |
|---|---|---|---|

* The tester have to carefully record, review and check test results at any level of testing.

* The tester must determine from the results whether the unit has passed or failed the test.

* If the test is failed, the nature of the problem should be recorded in a test incident report.

* When a unit test fails a test there may be several reasons for the failure. The most likely reason is a fault in the unit implementation (code)

Other reasons are

1) A fault in the test case specification
2) A fault in test procedure execution
3) A fault in the test environment
4) A fault in the unit design.

* The causes of the failure should be recorded in a test summary report, which is a summary of testing activities for all the units covered by unit test.

* When a unit test has been completely tested & finally passes all of the required tests its ready for integration.

* Test summary report is a valuable document for the group responsible for integration & system tests.

3. How would you Identify the hardware and Software for configuration testing and explain what testing techniques applied for website testing?

(nov/Dec 19)
(apr/may 18)

Configuration testing:

* It is also called "Hardware compatibility Testing."

* During this test, test engineer validates that whether our application build run an different technology hardware devices or not?

* It is the process of checking the operation of the softwares, you are testing with all these various types of hardware.

Eg Different technology printers, Different LAN topologies etc.

* The different configuration possibilities are

1. The PC - well known Computer manufactures, such as DELL, HP, etc

2. Components - Various System boards, components, device drivers (CD-ROM, HDD, FDD) video, sound, fax modem

3. Peripherals - printers, scanners, keyboard

4. Interfaces - RJ-11, RJ-45, Fire wire

5. Options and memory - Various amount of memory

6. Device Drivers - It is a low level software

Isolating Configuation bugs:

* It is usually dynamic white box testing and Programmer debugging effort.

* A configuation problem can occur for several reasons all requiring someone to carefully Examine the Code, while running the software under different configuations to find the by

1. Your software may have a bug that appears under a boad Class of configuations.

Eg Greeting card program work fine with laser Printers but not with inkjet printers.

2. The software may have a bug specific only to one particular configuation.

3. The hardware device or its device drivers may have a bug that only one the uses a Unique display setting.

Eg software is run with a specific video card, the PC crashes

4. The hardware device or its device drivers may have a bug that can be seen with lots of other software

Sizing up the job:

* There are huge number of display cards, bund cards, modem available in network.

* These combinations are not possible to test.

These sizing up problem is solved by

1. Equivalence partitions
2. Boundary value analysis.

Approaching the task:

* When planning the configuation testing to keep the following steps carefully.

1. Decide the types of hardware will need
2. Decide what hardware brands, models and device drivers are available
3. Decide which hardware features, modes & options are possible.
4. Identify software unique features that work with the hardware configuration.

Obtaining the hardware:

* It reauired dozens of hardware set up for configurating testing. It is a expensive one.

Few ideas for overcoming this problem:

1. A great plan every tester on the team to have different hardware. It helps different configuation set up in user concern.

2. create and maintain good relationship between hardware manufacturers. It help to solve bugs easily.

3. Collect all the required hardware in our team and then, Purchase all the cheaper hardware.

4. If the user have budget, work with your project manager to contract out our test work to a professional configuration and compatibility test lab.

Identifying hardware Standards:

* Hardware specifications, standards to be tested by relevant concern illustration.

↳ These information collected from relevant concern websites.

Eg Apple hardware - developer . apple . com / hardware

windows logo/software - msdn. Microsoft . com / software

Configuration testing other hardware:

1. create equivalence partitions of the hardware based on input from the people who work with the equivalent, user project manager or user sales people.

2. Develop test cases and collect the selected hardware and run the tests

3. Follow Configuration testing approaches.

Website Testing:

Before Performing a testing on developed/web sites or web applications, its necessary to determine the following 3 things.

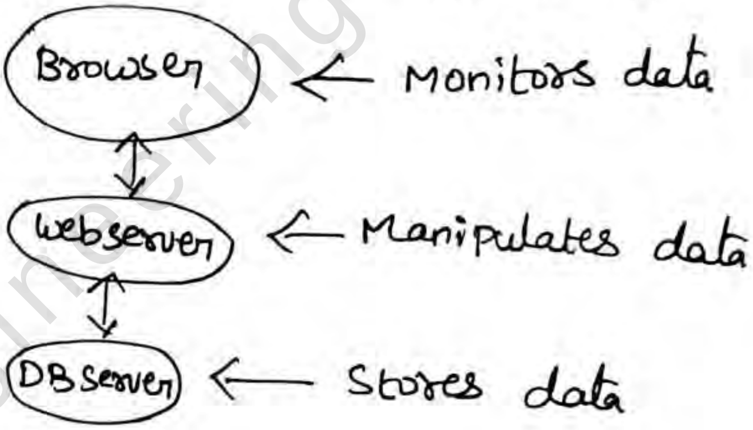
1. whether web based risk should be included in the test plan.

2. which types of web based testing is used?

3. selecting the appropriate web based test tools for the test execution phase.

*web sites testing done for 3 tier applications

web page/websites



objective:

* The objective of this test program is to assess the adeauacy of the web components of the software application.

Concerns:

1. Browser compatibility 2. Functional correctness
3. Integration 4. usability 5. security 6. performance
7. Verification of code.

Web Page Fundamentals

Internet web pages contains

1. Text of different sizes, 2. Fonts and colors
3. Graphics and photos 4. Hyperlinked text, images, graphics
5. Rotating ads 6. Drop down selection boxes
7. Text fields, buttons etc

Website creation is also complex, because.

1. Dynamic changes of text 2. Dynamic layout
3. Customizable layouts 4. Customizable contents

Different testing types to performed for website testing:

1. Black Box Testing (BBT)

* Its functional testing, Its used to test
1. Text, 2. Hyperlinks 3. Forms 4. Graphics
Eg HTML validation, Link checking, Functions Testing

2. Gray Box Testing (GBT)

* It is a mixture of white box testing (WBT) and black box testing (BBT)

* It help to test, HTML code (ie HTML validation)

4. White Box Testing (WBT)

* Its used for testing the internal part of the source code

* It must require coding knowledge

* Must hared knowledge about
1. Dynamic content 2. Database driven web pages
3. programmatically created web pages 4. security.

4. Configuation and compatibility Testing:

* For web site checking you must keep or follow possible configuation hardware and software

1. Hardware platform 2. Browser plug in 3. Text size
4 Modem speeds 5. Browser options

5 Usability Testing:

* It gives the better appearance, look and feel for websites.

TOP mistakes in web design.

1. Gratuitous use of bleeding edge technology
2. Scrolling text, marauees and constantly running animations.
3. Long scrolling pages
4. Non-Standard link colors
5. out dated information

Automation tools for website Testing:

Performance tools - Stress tested, load runner

Java testing tools - java NCSS, J cover, J link

Link checking tools - link tiger, link scan, link storm

Regressing checking tools - webrat, cubic test

1) Briefly Explain the levels of testing with an Example. [Nov/Dec-14] (nov/Dec 19,18)

* It is defined by a given environment. Environment is a collection of People, hardware, software, interfaces,

Need for levels of testing:

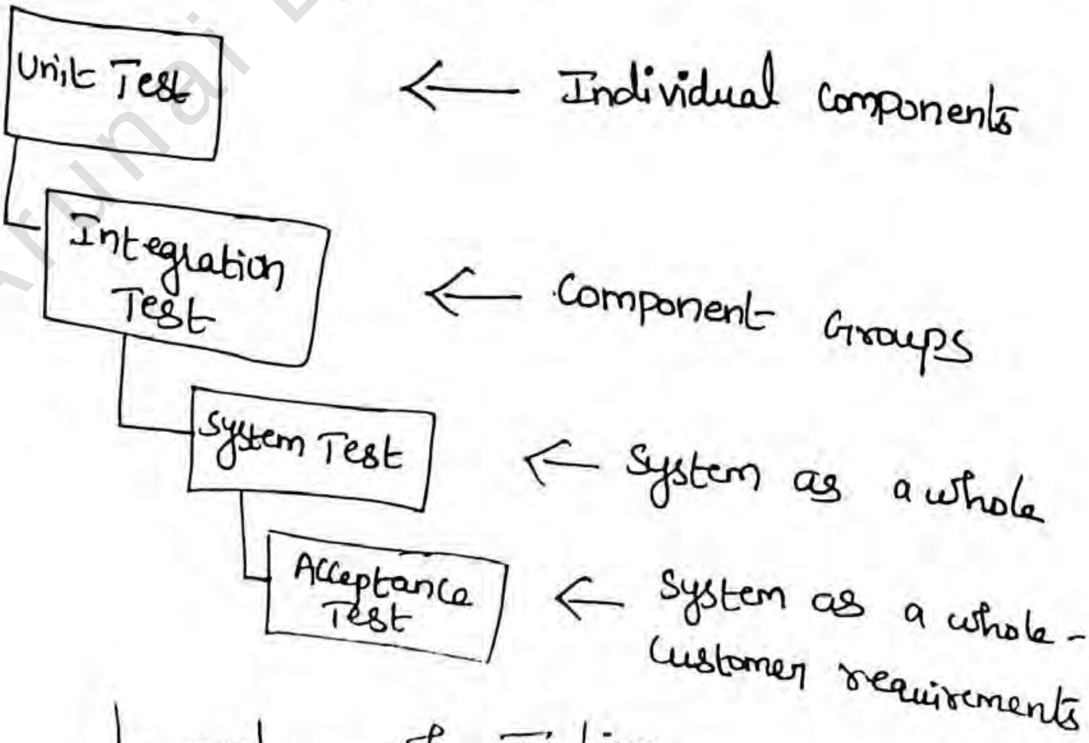
* It helps to enhance the quality of software Testing.

* To produce a more unified testing methodology applicable across several projects.

* Testing process to be abstracted easily.

* To identify missing Areas.

* To prevent overlap and repetition between the development life cycle phases.



Levels of Testing

The levels of Testing are

1. Unit Testing 2. Integration Testing
3. System Testing 4. Acceptance Testing

1. Unit Testing:

* A Unit test is smallest testable piece of Software. A principle goal is to detect functional and Structural defects in the unit.

* It can be compiled, linked & loaded.

Eg: Functions/procedures, classes, Interfaces.

* Its done by programmer.

* Individual components are to be tested, and assemble together.

* It is a better use a Buddy Testing

Types:

1. Execution Testing
   a) Statement coverage b) Branch coverage
   c) Path coverage.
2. operations Testing
3. Mutation Testing.

Buddy Testing:

* Team approach to coding and testing

* one programmer codes the other tests and vice versa

2) Integration Testing: Apr/may 18 Nov, dec17 3, 16

* At the integration level several components are tested as a group and the tester investigates component interactions.

* Test for correct interaction between system units.

* Defects are find between integrated modules

* Each modules are coded by different people. System/modules built by merging existing libraries.

* It helps to tests the interfaces among units.

* Integration testing is done by developer / tester

* It is done on programmers work branch.

* Integration testing is done when test cases written when detailed specification is ready.

* It helps to improve continuous throughput of system.

* It discovers inconsistencies in the combination of units.

Types:

1. Top down approach
   ↳ Use of stubs
2. Bottom up approach
   ↳ Use of Drivers
3. Hybrid Approach

3. System Testing:

* At the system level the system as a whole is tested and a principle goal is to evaluate attributes such as usability, reliability and performance.

* The major testing levels for both object-oriented and procedural-based types of system are similar.

* The nature of the code that results from each developmental approach demands different testing strategies.

Eg To identify the individual components and to assemble them into subsystems.

* System test begins when all of the components have been integrated successfully. It usually requires the bulk of testing resources.

* Eg Lab hardware, special Hardware, special software,

* At the system level the tester looks for defects, but the focus is on evaluating Performance, usability, reliability and other quality-related requirements.

* System Testing is done by the testing team.

Types:

1. Functionality Testing 2. Recoverability Testing
3. Interoperability Testing 4. Performance Testing
5. Scalability Testing 6. Reliability Testing
7. Regression Testing 8. Security Testing.

Testing approaches are used two major types of Programming language.

1) procedure -oriented Programming language.

2) object-oriented Programming language.

The written code needs testers to use the different strategies to

i) To find the test components

ii) To find component groups

System development with procedural languages are represented as a compassed things of,

* passive data *Active procedures.

* when test case are developed then it must to generate the input data to pass to the procedures inorder to reveal defects.

* The object oriented systems are taken into a composition of active data along with allowed operations on that data.

*

Procedural System:

* In a procedural system, Lower level of abstraction is described as a function or a procedure.

* The Higher level of abstraction is described by group of producers (or) functions.

* Both level to be combined and finally produces the system as whole, which is the highest level of abstraction.

4. Acceptance Testing:

* During acceptance Testing the development Organization must show that the software meets all of the client's requirements.

* A successful acceptance test provides a good opportunity for developers to request recommendation letters from the client.

* Software developed for the mass market often goes through a series of tests called alpha and beta tests.

Alpha tests - Alpha tests bring potential users to the developer's site to use the software. Developers note any problems

Beta tests - Beta tests send the software out to Potential users who use it under real-world conditions and report defects to the developing organization.

Levels of Testing and Software Development paradigms:

* when an approach is used to design and develop a software system, its based on

1. Tester's plan 2. Design of tests

Two approaches to system development are

1. Bottom up approach 2. Top-down approach.

2. object oriented system:

* In a object objected system, lower level of abstraction is described as the method or member function.

* Next highest level is described by the class that encapsulates data and model.

* Next highest level is cluster. It helps to combine more than classes.

* Finally, the system is described clusters and auxiliary code

* Testing is straight forward one,

Eg Encapsulation - Hide details from testers.

* object oriented code is characterized by
  - ↳ use of messages
  - ↳ Dynamic binding state changes etc.

Explain hao to test OO System in detail. [Nov/Dec-14]
Discuss the levels of testing adopted to test OO systems. Apr/May-15

Testing OO Systems:

Basic concepts of OO system that are relevant for testing are

1. Classes - A class is a representation of a real-time object.

2. objects - objects are the dynamic instantiation of a class.

3. Constructor - Specific instantiation are done using constructor.

4. Encapsulation - The implementation of the methods is hidden from the use. This enables the person writing the methods to optimize the implementation without changing the external behavior. This is called Encapsulation.

5. Inheritance - Defining a new classes from already existing classes.

6. polymorphism - Same method name but performing different functions.

Object oriented Testing:

* Testing an object oriented system should tightly integrate data and algorithm.

Testing OO System has

1. Unit testing aclass
2. Putting Classes to work together (integration testing of a classes)
3. System Testing
4. Regressing Testing
5. Tools for testing object oriented systems.

1. Unit testing a Class:

* Classes are the building blocks for an entire OO system.

Unit test the classes for the following reasons.

1) A class is intended for heavy reuse
2) Any defects get introduced at the time of class defined.
3) A class may have different features.
4) A class is a combination of data and methods.

The Alpha-omega method:

* It takes the object under test from the alpha State to the omega state by sending the message to every method at least once.

* It shows that all methods in aclass are minimally operable. It support Extensive testing.

* An alpha-omega method has six basic steps

1 New or Constructor method
2. Accessor (get) method
3. Boolean(Predicate) method
4. Modifier (set) method
5. Iterator method
6. Delete (Destructor) method

2. Putting classes to work together (Integration testing of a class)

* It focuses on groups of classes that collaborate of communicate in some manner.

* Integration of operations once at a time into classes is often meaningless.

* Regression testing is important as each thread, cluster, subsystem is added to the system.

* Integration applied 3 different incremental strategies.

1. Thread based testing - Integrates classes required to respond to one input or event.

2. Use based testing - Integrates classes required by one use case.

3. Cluster testing - Integrates classes required todemonstrate one collaboration.

3. System Testing and Interoperability testing:

* Different classes may be combined together by a client and this combination may load to new defects.

* The complexity of interactions among classes can be substantial.

* It is important to ensure that proper unit & component testing is done before attempting system Testing

* proper entry/exit conditions should be set for the system Testing.

Design error- handling paths that test external information.

* conduct a series of tests that simulate bad data
* Recard the results of tests to use as evidence.

Types: Recovery, stress, load, performance etc.

4. Regression Testing of OO systems:

* Changes to any one component could have potentiall Unintended side effects on the clients that use the components

* Frequent integration and regression runs become very essential for testing OO systems.

* Because of the cascaded effects of changes resulting from properties like inheritance, it make sense to catch the defects as early as passible.

Tools for testing of OO system:

There are several tools that aid in testing OO systems.

1) Use cases
2) Class diagrams
3) Sequence diagrams
4) Activity diagrams
5) State diagrams.

1) use cases:

* It represents the various tasks that a user will perform when interacting with the system.

* Uses cases go into the details of the specific Steps that the user will go through in accomplishing each task and the system responses for each steps

2) Class diagrams:

* It represent the different entities and the relationship that exists amoung the entities

* Class diagrams elements are

i) Boxes - Classes

ii) Association - Relationship between two classes by a line

iii) Generalization - Child class derived from parent class

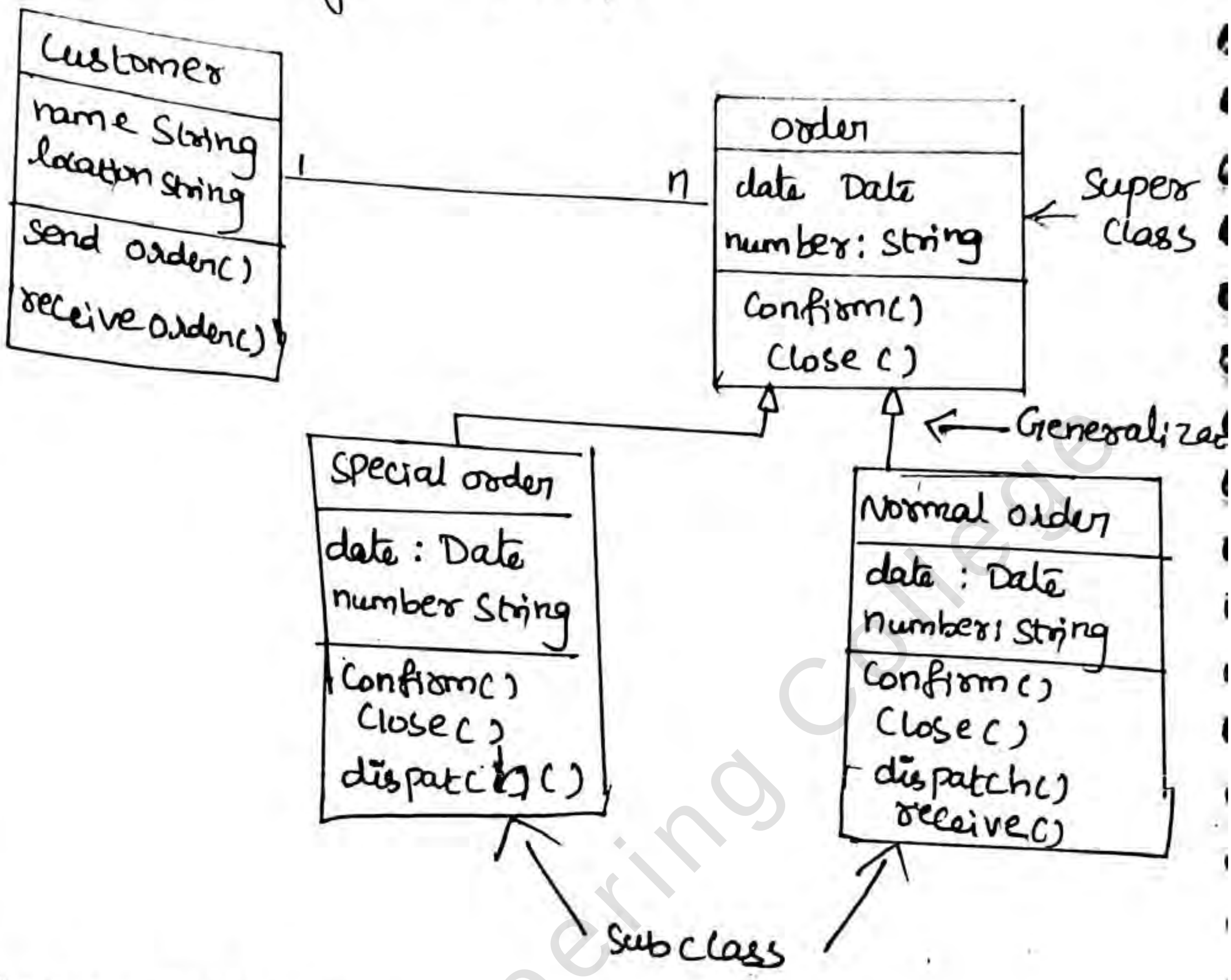
A class diagram is useful for testing in following ways

1. It identifies the element of a class
   Eg ECP, BVA etc

2. The associations help in identifying tests for referential integrity constraints across classes

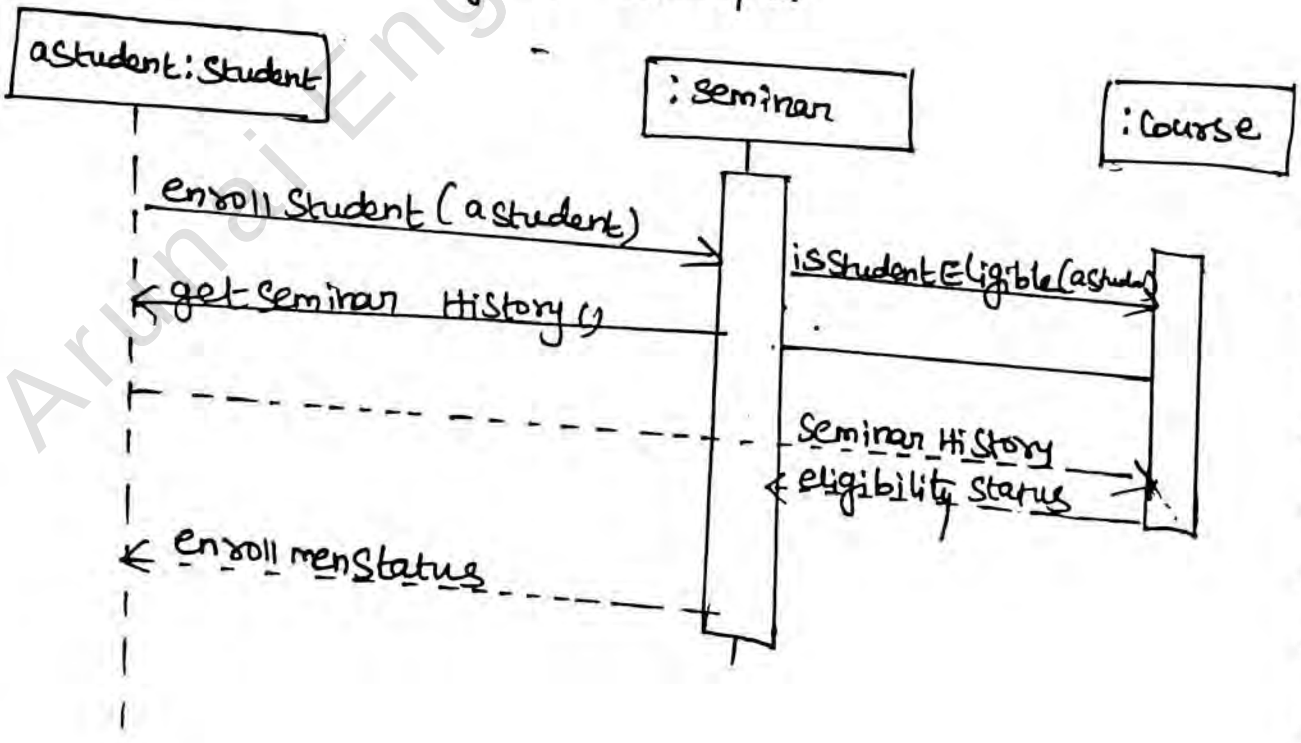
3. Generalizations helps in identifying class hierarchies

Class diagram - Example



3. Sequence Diagrams:

Sequence Diagram Example:



* Sequence diagrams represents a sequence of messages passed among to accomplish in a given application scenario or use case.

* A Sequence diagram helps in testing by

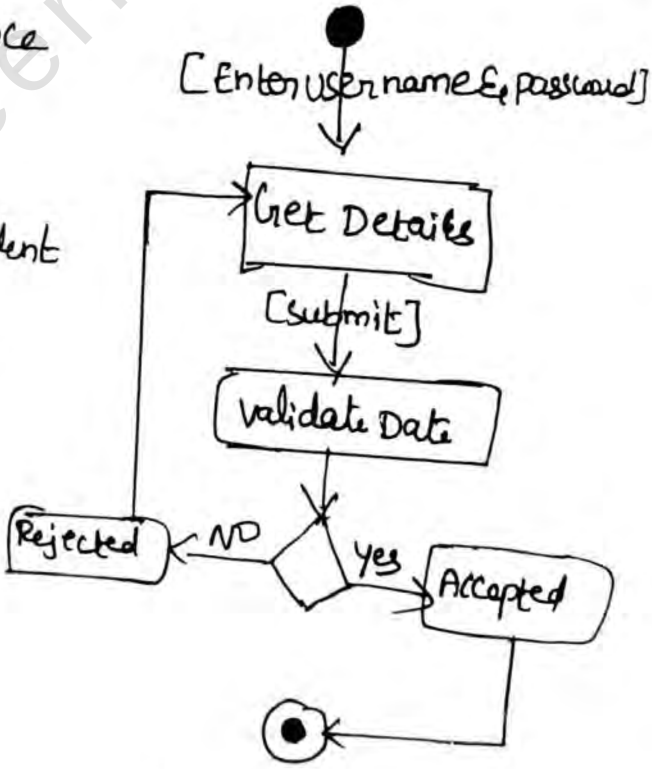
1. Identifying temporal end-to-end messages
2. Tracing the intermediate points in end-to-end transaction.
3. providing for several typical message calling Eg: Blocking Call, Non blocking call

4. Activity Diagram:

* It represents the sequence of activities.

* Its simply represent independent Program Paths whenever code complexity is arrive.

* To identify the possible message flows between object and classes.

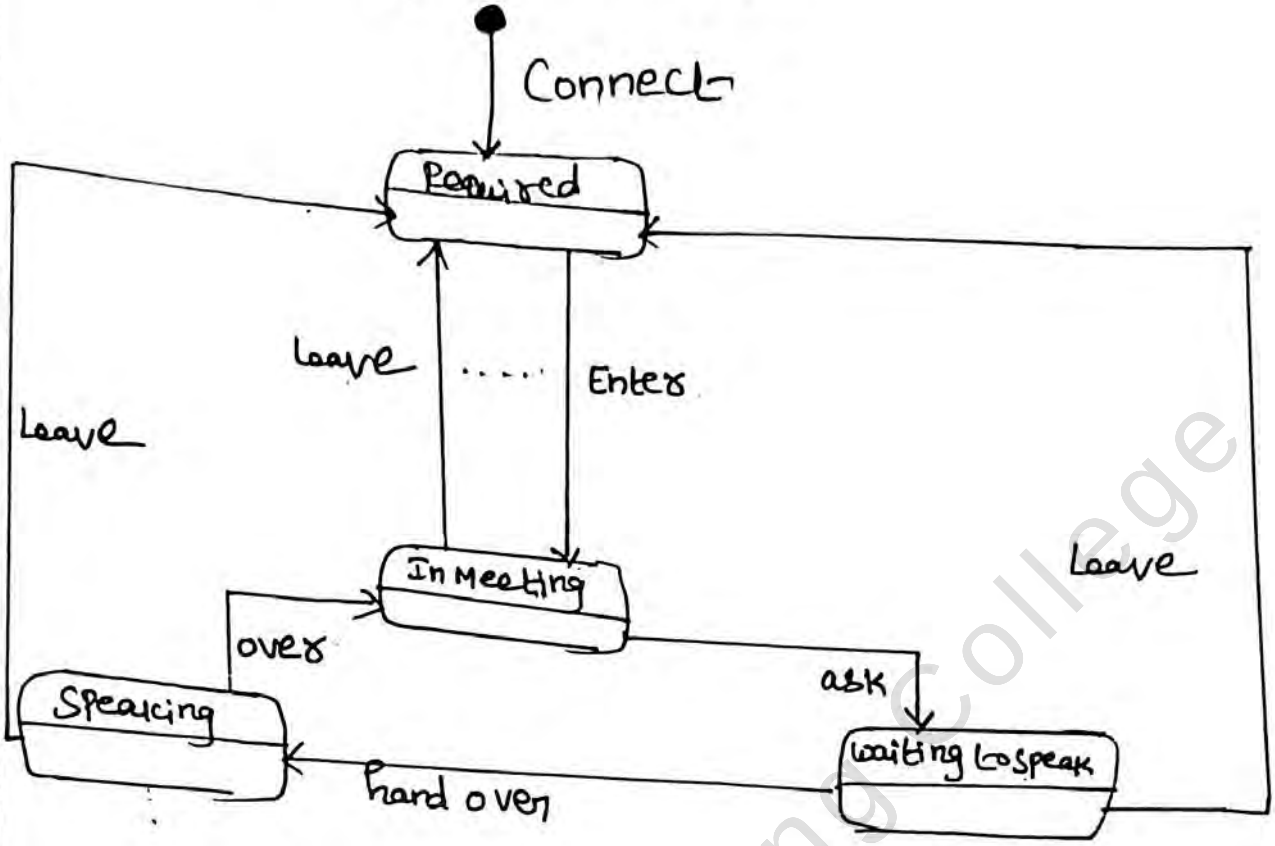


Activity diagram-Example

5. State Diagram:

* when an object can be modeled as a state machine, then the techniques of state based testing can be directly applied.

Stale diagram - Example



# IT8076-SOFTWARE TESTING

# UNIT-4
# TEST MANAGEMENT

# 2 MARKS & 16 MARKS
# WITH ANSWERS

# UNIT - IV

## Part - A

Make distinctions between structures of single Product and Multi product companies. [May|june-16]

| Structures in single product companies | Structures in Multi product companies |
|---|---|
| * In Single product companies all product is treated as single business unit | * In multi product companies, every product is treated as separate business unit. |
| * Most product companies Start with a single product | * Dependence among various Product |
| * Every Engineers report into the project Manager. project Manager handles each & every Part of the projects. | * Development & Testing are the same level of importance in the concern. Testing team report directly to the CTO as a part to the design & Development teams |

Mention the reasons to Create a WBS [May|june-16]

Work breakdown Structure (WBS) - It is a hierarchical or tree like representation of all the tasks are required to completed a project (apr, may 18)

The reason to create a WBS

1) Easy to understand and follow
2) Checklist for the project
3) Accurate Estimation

3. 4) Easily monitored and tracked

5) Easier and faster project planning

6) Assigned to individual team members, and accountable

7) Reduces the project risk.

---

3. List the organization structures for testing teams.
[may/june-16]

Organization structures is viewing upon two dimension.

1. Organization Type
   - * product organizations
   - * Service organizations

2. Geographic Distribution
   - * Single site
   - * Multi site

---

4. What are the Skills needed by a test Specialist?
[may/june-16]
[nov/dec-14]
[Apr/may-15]
(nov, Dec 19,17)

i) Personal and Managerial Skills
   - * organizational and planning skills
   - * The ability to resolve conflicts

ii) Technical Skills
   - * knowledge of process issues
   - * knowledge of configuration management

---

5. Differentiate decision and condition coverage. Apr/May-17

| Decision Coverage | Condition coverage |
|---|---|
| 1. It checks whether every edge of the program is covered or not. | 1. It checks whether every conditional statements are covered or not |
| 2. It evaluates each edge of every control structure are covered or not<br>(ie) IF and CASE Statement | 2. It evaluates for only TRUE or FALSE condition for each conditional statement not every edge of those conditional statement. |

6. State the limitations of Statement Coverage. [Apr/May-17]

* The statement coverage is also known as line coverage or segment coverage.

* It covers only the true conditions.

Limitations of statement coverage

1. It cannot test the false conditions

2. It does not report that whether the loop reaches its termination condition

3. It does not understand the logical operators.

7. Q. Discuss the role of manager in a test group [Nov/Dec-16]

The role of manager in a test group

The Manager's view involves commitment and support for those activities and tasks that related to improving testing process quality.

* Task forces, policies, standards,
* planning
* Resource allocation
* Support training/ education
* Interact with user/ Clients

8. Q. Write the components of Test plan [Nov/Dec-14]

Test plan components are represented as

1. Test plan identifier
2. Introduction
3. Test items
4. Features to be tested
5. Approach
6. Feature pass/ fail Criteria
7. Suspension/ resumption criteria
8. Test deliverables
9. Testing tasks
10. Test environment
11. Responsibilities
12. Staff and training needs
13. Scheduling
14 Risks and Mitigations
15. Testing costs
16. Approvals.

1. Explain the different Challenges and issues faced in the testing service organizations Discuss how those Challenges can be addressed [may/June - 16]

Challenges and Issues in Testing services organizations:

* All testing organizations face certain common challenges.

* In the case of a testing services organization to which testing is outsourced, some of these challenges are exaberbated, primarily because of the arm's length distance from the development team.

Main challenges are

1. The outsider effect and estimation of resources
2. Domain expertise
3. Privacy and customer isolation issues
4. Apportioning hardware and software resources and Costs
5. Maintaining a "Bench".

1. The outsider effect and estimation of resources

* The testing Services organization is an "outsider" to the development organization

Some of the implication are as follows:

1) They are not holding the product internal or code

2) It is not passible to know the product history for them

3) They are not maintaining the same level of rapport with the development teams

4) Internal development and testing teams do not necessarily to have the information about the software and hardware resources required.

2. Domain Expertise:

* A testing team in a product organization can develop domain Expertise in a specific domain.

Eg:-Domain

1. ERP domain
2. Logistics domain
3. Banking domain etc

* Domain Expertise extract the information from customers. It is hard to retrieve the information.

* Softwaxe oxganization can develop specialized expextise in the ERP domain.

* The organization can hire a few domain specialists who can augment the testing team.

* Such specialists will find it interesting to join product organization because they can find a natural transition from their domain to the more attractive IT arena

3) privacy and customer isolation issues:

* A Testing Services Organization will have to work with multiple customers.

* AS an Organization works with customers in a given domain (Eg Pharmaceutical or financial services) it develops not only general expertise but also domain expertise in a particular domain

* Two factors contribute to this being a major challenge.

1. The testing service organization has a common infrastructure and hence physical isolation of the different teams may be difficult

2. The customers can go to one project from other project.

* At that time it must to confirm that Specific knowledge required in one project cannot be taken to other project

4) Apportioning hardware & software resources and Costs

* when a testing service organization bids for and works with multiple customers, it would have to use internal hardware and software resources.

* Some of these resources can be identified directly and specifically for a particular account.

* There are other resources that get multiplexed across multiple projects.

Eg Communication infrastructure such as satellite links, common infrastructure such as email servers and Physical infrastructure costs.

5) Maintaining a "Bench"

* To keep a" People on the bench", because customer suddenly give a new project to the testing service organization.

* These People not allocated to any other Projects but ready in the wings to take on new projects or to convince customers.

Explain the components of test plan in detail

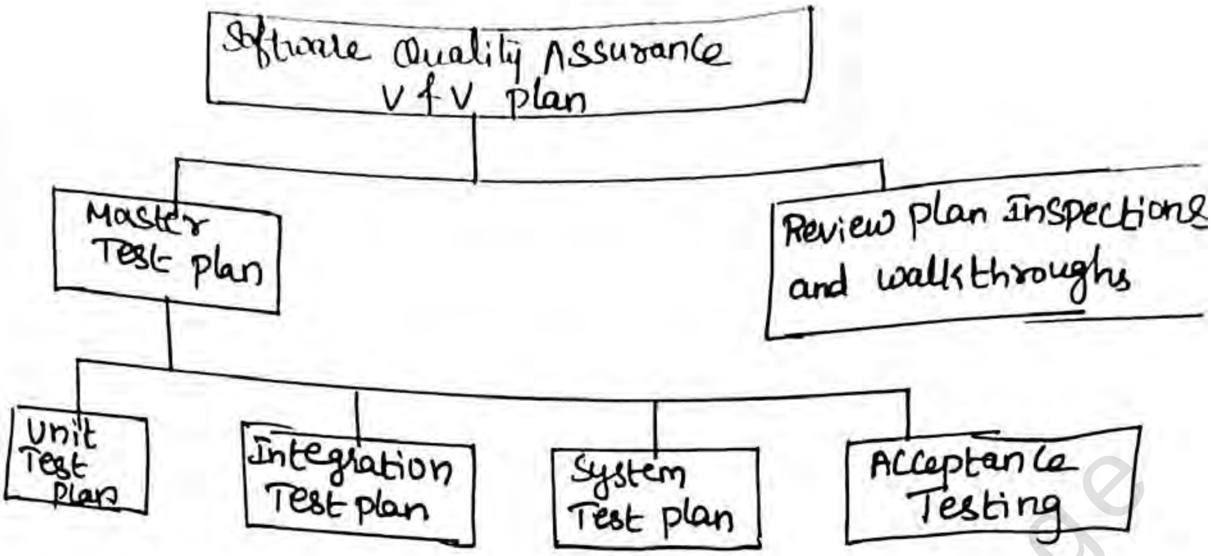
what is a test plan? List and Explain the test plan Components [Nov/Dec-16]

Test plan Components: (nov/Dec 19,18) (apr/may 18) [May/june-16]

* It is also called test plan format / Test plan Unit.

Test plan Components are represented as

1. Test plan Identifier
2. Introduction
3. Items to be tested.
4. Features to be tested
5. Approach
6. Pass / fail Criteria
7. Suspension / resumption Criteria
8. Test deliverables
9. Testing tasks
10. Test Environment
11. Responsibilities
12. Staffing and training needs
13. Scheduling
14. Risks and Mitigations
15. Testing Costs
16. Approvals



1. Test Plan Identifier:

* Every test plan holds unique number or name.

* It help to used to identify a particular project

* Organizational standards Explain

↳ Format of test plan identifier

↳ Version representation.

*It is used to find whether it is a configuration item or not in the Configuration management system

2. Introduction:

* The test planner gives an overall description of the project

* It includes

1. Software system development
2. Software items to be checked
3. High level description of testing goals
4. Testing Approaches 5. References

Eg Organizational policy, standards, Quality plan, Configuation plan.

3. Test items

* This is list of the entities to be tested and should include names, identifiers and version numbers for each entry.

* The items listed could include procedures, Classes, modules, Libraries, Subsystems, References Functions, services

4. Features to be tested:

* Features may be described as distinguishing Characteristics of a software component or system.

Eg: Performance, portability, Functionality.

* The test plan references to test design specifications for each feature and Each combination of features are identified to establish the associations with actual test cases.

5. Approach:

* This section of the test plan provides broad coverage of the issues to be addressed when testing the target software.

* The level of descriptive detail should be Sufficient so that the major testing task and task durations can be identiffed.

* More details will appear in the accompanying test design specification.

* The planner should also include for each feature or Combination of features the approach that will be taken to ensure that each is adequately tested.

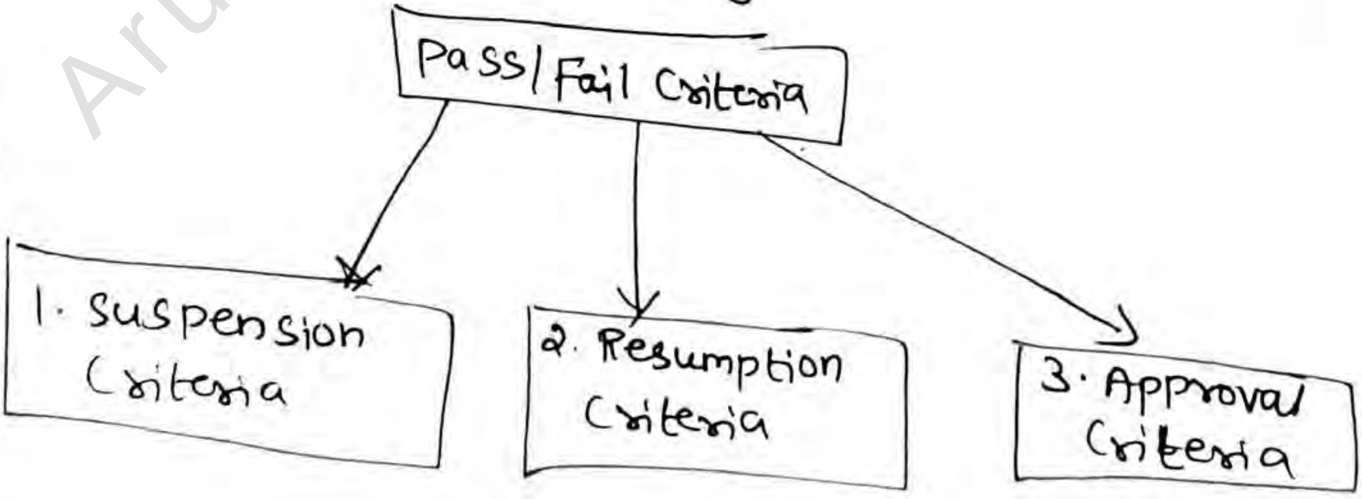
* Tools and techniques necessary for the tests should be included.

*↳ Expectations for test completeness

*↳ How the degree of completeness will be determine Should be described

6. Item pass/Fail Criteria

* It Specify the criteria that will be used to determine whether each test item (software/product) has passed or failed testing.



7 Suspension Criteria:

* It specify the criteria used to suspend all or a portion of the testing activity on test items associated with the plan

Resumption Criteria:

* Specify the conditions that need to be met to resume testing activities after suspension.

* Specify the test items that must be repeated when testing is resumed

Approval Criteria:

* Specify the conditions that need to be met to approve test results.

* Define the formal testing approval process

8. Test Deliverables:

* Identify the deliverable document from the test process.

* Deliverable may also include other documents that results from testing such as test logs, test transmitted reports, test incident reports and test summary reports.

9. Testing tasks:

* The test planner should identify all testing related tasks and their dependencies.

* Using a Work Breakdown Structure (WBS) is useful here.

* A work breakdown structure is a hierarchical or tree like representation of all the tasks that are required to complete a project

10. The testing Environment:

The test planner describes the software and hardware needs for the testing effort.

Eg Any special equipment or hardware needed such as emulators, telecommunication equipment or other devices should be noted.

11. Responsibilities:

* Identify the groups responsible for managing, designing preparing, executing, witnessing, checking, transmitting, developing, tracking & Monitoring, Interacting and resolving testing activities

* These groups may include the developers testers, operations staff, technical support staff data administration staff & the user staff

4 Cost of training

5 Costs of maintaining the DB

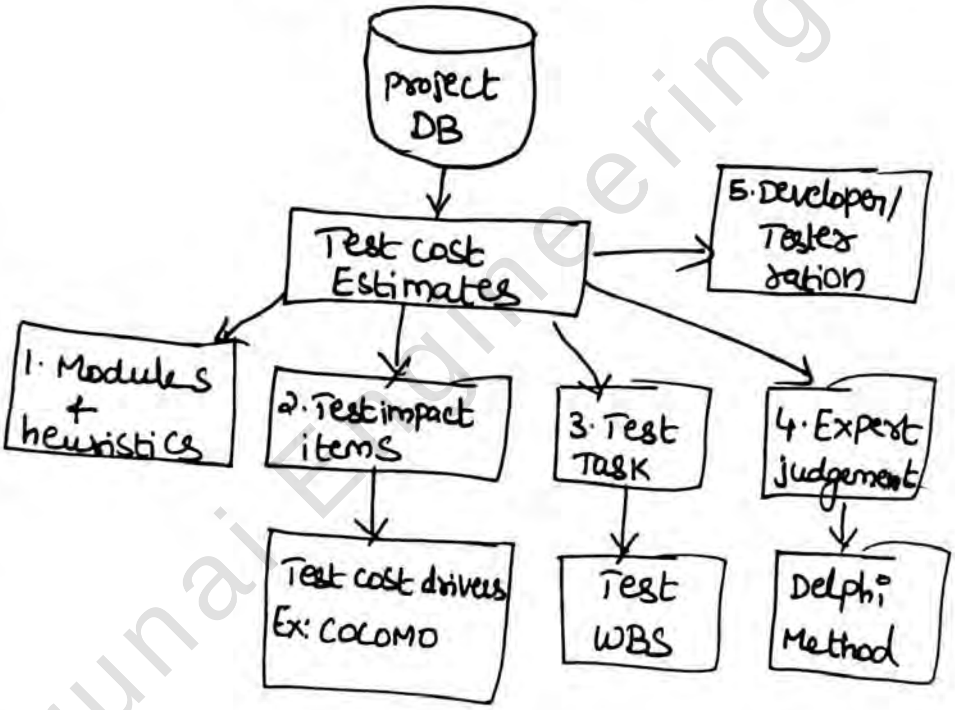
* Test Cost impact item are

1. Maturity Level of organization
2. Nature of Software product
3. Scope of test requirement
4. Tester ability

* Project Planners follow some Cost estimation models

Eg COCOMO model

Test cost Estimation Techniques



16. Test Approvals:

* Identify the plan approvels

* Specify the list the name, Signature & data of plan approvals.

12. Staffing & Training needs:

* The test planner should describe the staff and the skill levels needed to carry out test-related responsibilities.

* Any special training needed to perform a task should be noted.

13. Scheduling:

* Identify the high level schedule for each testing task.

* Establish specific milestones for initiating & completing each type of test activity, for the receipt of each test input & for the delivery of test output.

14 Risk & Mitigations:

* Identify significant constraints on testing such as test item availability, test resource availability and time constraints

* Identify the risks & mitigations associated with testing tasks including schedule, resources, approach & documentation.

15 Testing costs:

It includes in the plan

1. cost of planning
2. costs of designing the tests
3. costs of test environment

3) Briefly discuss the various groups in Test plan and Policy development with their role [NOV/DEC-14]

The Role of various groups in Test planning and Policy Development.

* Each group views the testing process from a different perspective that is related to their particular goals, needs and requirements

3 critical groups in testing are

1. Managers
2. Testers/Developers
3. User/Clients

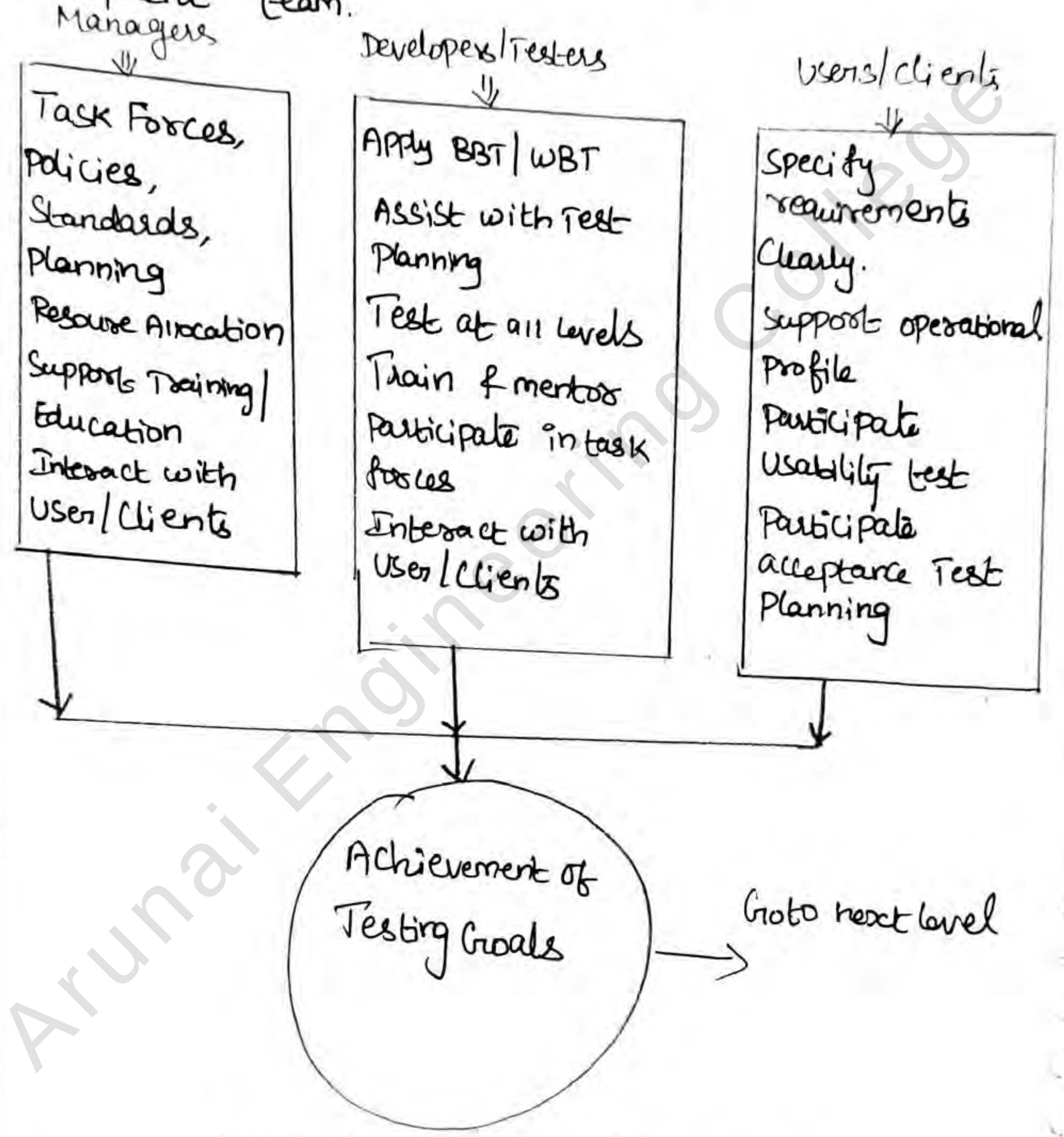
↳ The Manager's view involves commitment and support for those activities and tasks related to improving testing Process quality.

↳ The Developer (Tester's view) encompasses the technical activities and tasks that when applied, constitute best testing practices

↳ The User/Clients view is defined as a cooperating or supporting view.

* Developers have an important in the development of testing goals & policies.

* They serve as members of the goal/policy development team.



Summary of critical Group roles

* The developers or testers work with client/user groups on quality-related activities and tasks that concern user-oriented needs.

* The developers/testers activities are,

1. working with management to develop testing & debugging policies & goals.

2. Participating in the teams that oversee policy compliance & change management.

3. Familiarizing them selves with the approved set of testing/debugging goals & policies

4. Keeping up-to-date with revisions & making suggestions for changes when appropriate

5. when Developing test plan, setting testing goals for each project at each level of test that reflect organizational testing goals and policies.

6. Carrying out testing activities that are in compliance with organizational policies.

↳ Users/clients plan an indirect role in the formation of an organization's testing goals & policies.

* Since these goals & policies reflect the organizations effort to ensure customer/client/user satisfaction

Management Support:

1. Establishing an organization wide test planning Committee with funding.

2. Ensuring the testing policy statement & Quality standards support test planning with commitment of resources, tools, templates and training.

3. Ensuring that all projects are in compliance with the test planning policy.

4. Ensuring that all developers/testers complete all the necessary post test documents such as test log, test incident, test summary reports.

Project Manager-

* They support test planning maturity goals by preparing the test plans for each project with inputs & support from developers.

Developers-

* who are experienced in testing support this maturity goal by participating in test planning.

* They assist the project manager in determining test goals, selecting test methods, procedure & tools & developing the test case specificiation, test procedure specification.

* From the user/client point of view support for test planning is in the form of articulating their requirements clearly and supplying input to the acceptance test plan.

a) What are the Skills needed for a test specialist

b) Explain the organizational Structure for testing teams in single product companies [NOV/DEC-16]

(apr/may 18)

a) Skills Needed for a test specialist: (nov, Dec 19,17) (apr,may 18)

* The nature of technical and managerial responsibilities assigned to the tester that are listed many managerial and personal skills are necessary for success in the area of work.

Personal and Managerial Skills

1) organizational and planning Skills

2) The ability to keep track of and pay attention to, detail

3) The determination to discover and solve problems

4) The ability to work with others and resolve conflicts

5) Mentor and train others

6) The ability to work with users and clients

7) Strong written and oral communication skills

8) The ability to work in a variety of environments

9) The ability to think creatively.

* The first three skills are necessary because testing III detail and problem oriented.

* In addition, testing involves policy making, a knowledge of different types of application areas,

Planning and the ability to organize and monitor information, tasks and people.

Technical Skills

1. General Software Engineering principles and Practices

2. Understanding of testing principles and practices

3. Understanding of basic testing strategies and methods.

4. Ability to plan, design and execute test cases

5. knowledge of Process issues

6. knowledge of networks, databases and os

7 knowledge off configuration management

8. knowledge of test-related documents

9. Ability to define, collect and analyze test-measurements

10. Ability, training and motivation to work with testing tool

11. knowledge of quality issues.

* A good understanding of testing principle and Practices

* A good understanding of basic testing strategies, methods and techniques.

* A knowledge of process issues

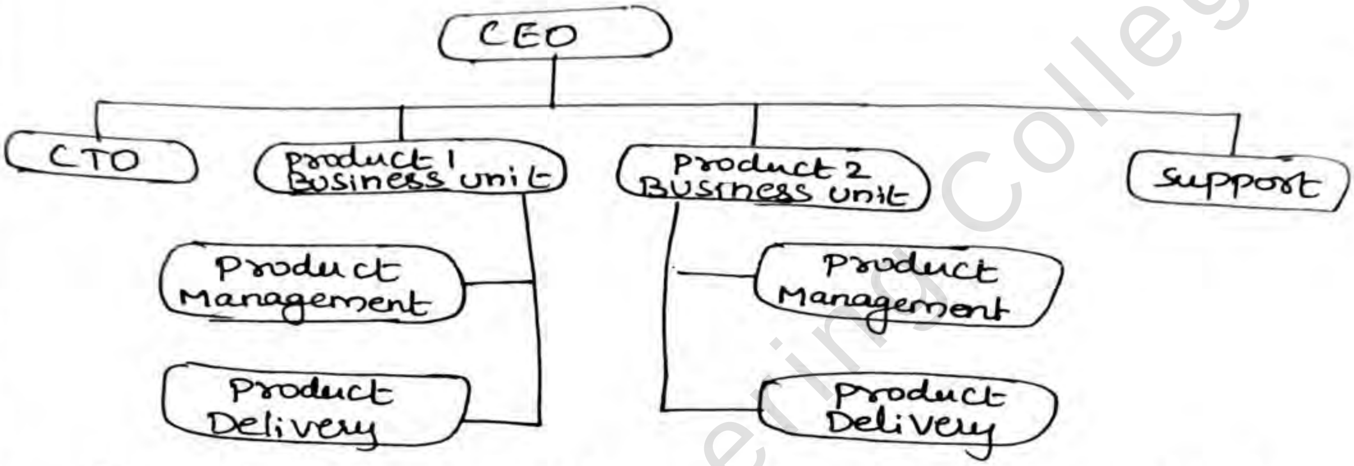
* A knowledge of configuration management

* The ability to define, collect and analyze test-related measurements.

b) Organization Structures for Testing Teams

Structure - in single product companies.

* Product companies in general have a high-level organization structure.



Organization structure of a multi-product company.

* The CTO'S office sets the high level technology directions for the company.

* A business unit is in charge of each product that the company produces.

* A product business unit is organized into a product management group and a product delivery group

* The product management group has the responsibility of merging the CTO'S direction with specific market needs to come out with a product road map

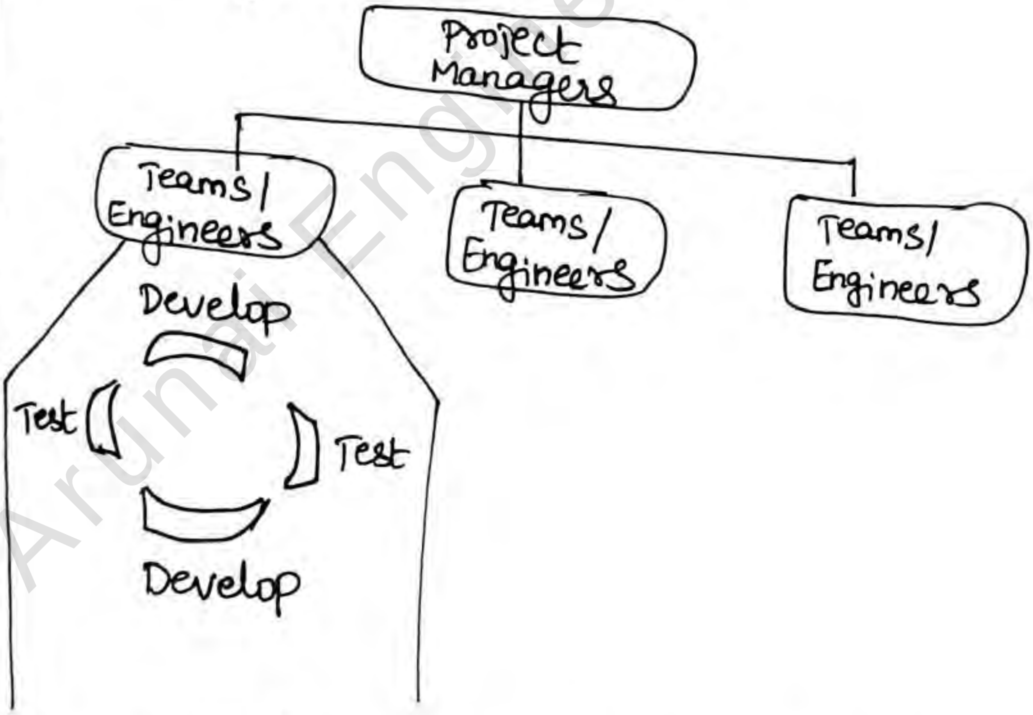
* The product delivery group is responsible for delivering the product and handles both the development and testing function

Testing team structures for single-product companies:

*Most product companies start with a single product

* The product delivery team members distribute their time among multiple tasks and often wear multiple hats.

*Very thin line separating the development team and "testing team"



Typical organization structures in early stages of a product

Advantages:

1. Exploits the rear-loading nature of testing activities

2. Enables engineers to gain experience in all aspects of life cycle.

3. IS amenable to the face that the organization mostly only has informal processes

4 Some defects may be detected early.

Disadvantages:

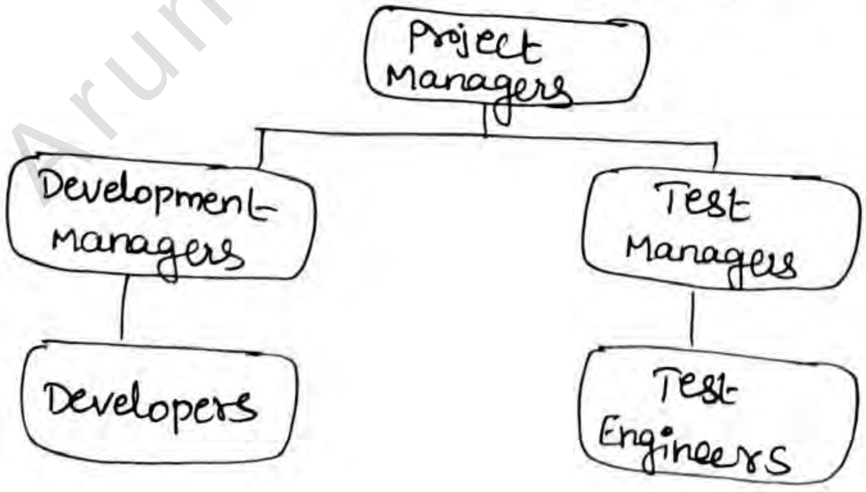
1. Accountability for testing & Quality reduces

2. Developers do not in general like testing and hence the effectiveness of testing suffers.

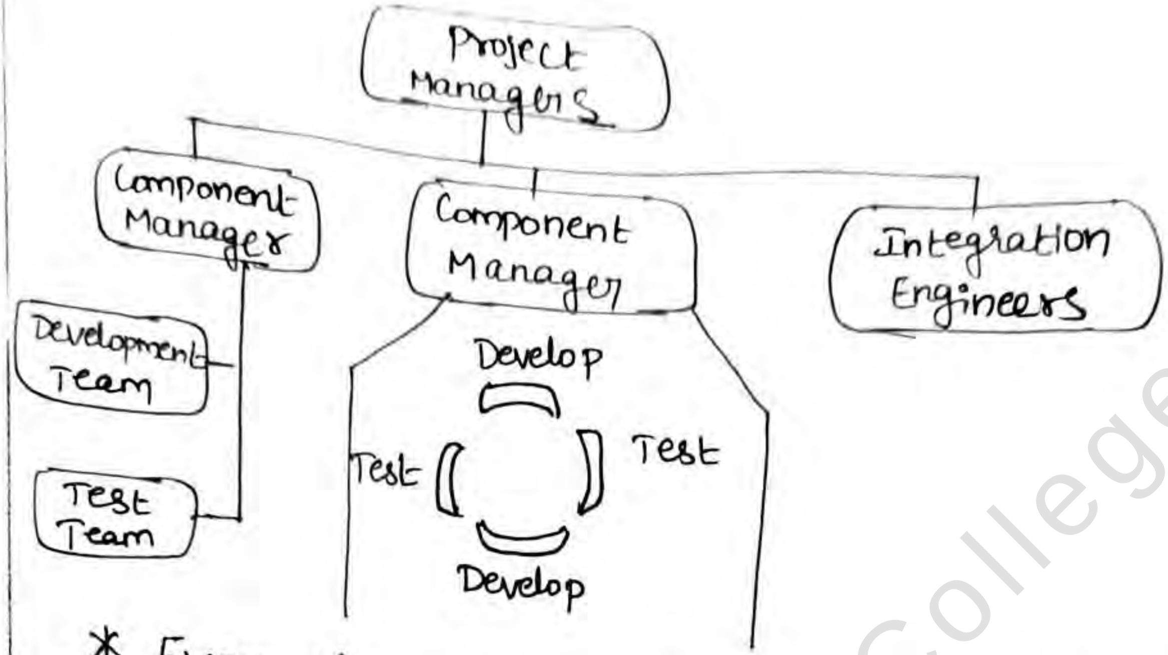
3. Schedule pressures generally compromise testing.

4. Developers may not be able carry out the different types of tests.

Separate groups for testing & Development



Component wise Testing teams



* Even if a Company produces only one product, the Product is made up of a number of Components that fit together as a whole

* Every components is developed & tested by separate team

* All these components are integrated by using a single integration test team

* This team report to the Project Manager.

5) M Explain issues caused by people and organization in Software Testing. [NOV/DEC-14]

People and organization Issues in Testing:

Perceptions and misconceptions about Testing

Perception-1

"Testing is not technically challenging"

* Testing is simple and repetitive job

* It does not require specilized Skills

* It is a manual Task

* If product is simple, easy to test

* "There is testing in all development and development in all testing."

Misconception:

Testing needs following things

1. It needs a holistic understanding of the entire project

2. Test engineer must be a "Domain expert" (or) "Domain Specialists"

3. Testing requires thorough understanding of multiple domains.

4. Tester must specilization in programming and test ~~Script languages~~.

5. For better design and integration tester/developer must understand the usage of tools.

6. opportunities for conceptualization and out-of-the box thinking.

7. Significant investments are made in testing today Sometimes a lot more than in development

Perception - 2

"Testing does not provide Me a Career path or Growth"

* Career path & rules defined for the each Organization People differently.

* It helps to improve development, Standard, and different software engineering functions.

* "Testing is not a devil and development is not an angle", "opportunities abound equally in testing & development".

Misconceptions:

1. No much Career path opportunities in testing field.

2. Define separate rules for development engineer, senior development engineer, Business Analyst, domain expert.

Perception-3 1, 13

"I am put in Testing - what is wrong with me?"

* If a person is not suitable for development, for the same or similar reason he or she may not be suitable for testing either

* People are sometimes made to feel that they are in testing because they could not fit in anywhere else

Consider some of the causes and effects of such messages.

1) Filling up positions for testing should not be treated as a second string function.

2) If a person having capability in the field of testing then the person is appointed for testing.

3) Then compensation schemes should not discreminate against any specific function.

4) Appropriate recognition should be given to the engineer who participate in activities such as testing, maintenance and documentation

Perception-4

"These folks are my adversaries"

* Testing & Development teams should reinforce each other and not be at logger heads.

* The main function of testing is to find errors in the software. It is easy to the adversary

attitude to develop between testing & Development team

perception - 5

"Testing is what I can do in the end if I get time"

* Testing is not what happens in the end of the project - it happens through out and continues even beyond the release.

* Testing is a planned one. It is not constructed end of the cycle.

* Adequate time, resources, software, knowledge are essential for testing.

* otherwise it creates risk (ie Loss)

These problems (risks) are to be avoided In testing are,

1. Stipulate completion/acceptance criteria for testing team to accept a product from the development team.

2. Giving the testing team to work freely to mandate a minimum quality in the product before it can be released.

perception - 6

"There is no sense of ownership in Testing"

* Testing has deliverables just as development has and hence testers should have the same sense of ownership

* The ownership is sometimes not created for testing functions.

* A possible contributor to this feeling of lack of ownership is the apparent lack of "deliverables" for testing functions.

Perception -7:

"Testing is only Destructive"

* Testing is destructive as much it is constructive, like the two sides of a coin.

* Tester must perform/destructive

1. The part which does not work
2. what work's going on the product
3. Express the work in the product
4. Analyze the risk of the product before release.
5. Giving the mitigation plan for perfect perspective.
6. Save company money & image.
7. Say the ways to solve the problem.

Providing carrier paths for testing Professionals:

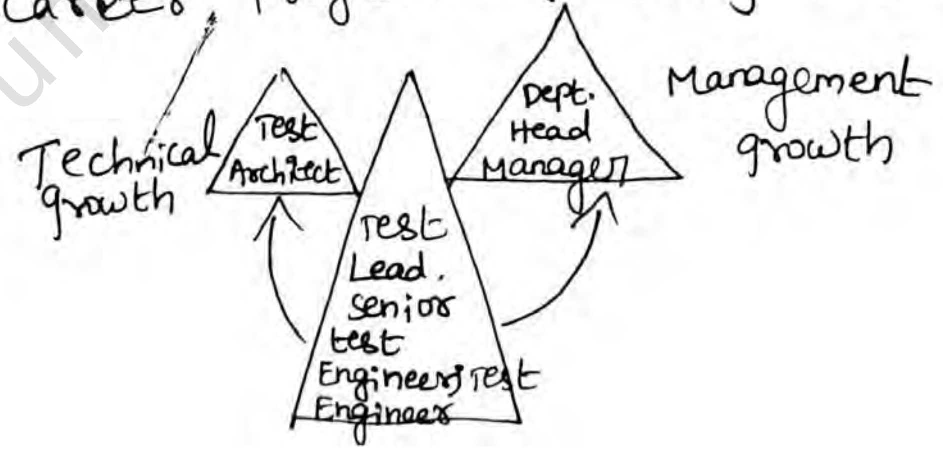
* When performing carrier path testing, must noted following areas.

1. Technical challenge
2. Learning opportunities
3. Increasing responsibility
4. Increasing authority
5. Increasing independence
6. Ability of an organization success
7. Rewards
8. Recognition

Three stages of individual in testing group goes through

1. Follow stage: 2. Formulation stage
3. Test lead stage.

Career progression for testing Professionals



Comparison between Testing & Development Team:

1. Testing is frequently said to be acrunch time function.

* Testing is treated carefully in the product release time.

* Because it throws some unique planning, management challenges, deliverables.

2. More "elasticity" is allowed in project's in early phases.

* planning testing projects affords less flexibility than development projects.

3. Testing functions are arguably the most difficult ones to staff

* Testing is not a smaller function. only minimum number of people chooses the testing as a job.

* It is hard to attract and retain top talent for testing functions and its also carried under time pressure.

4. Testing functions hold several External dependencies when compared with the development functions

* Testing is done at the end of the project life cycle.

The role of Ecosystem and a call for action

Role of education System.

* There are collective and much higher-level actions that need to be done.

* The entire eco system covering the eduction System, and the community as a whole

i) Role of Education system:

* Education System does not provide sufficient importance in the testing field

The reasons are

1. Many formal core courses on programming but only few universities offer core courses on software testing.

2. only few "lab Courses" for common testing tools, compare to other programming lab courses.

3. No complete weightage for coding & testing.

4. There is no reward for test engineers and their skills.

5. Products are based on Quality, not on demands So demand reduces the ability/quality.

* To improve the awareness of testing is essential for software testing knowledge, skills, attitude towards testing etc.

ii) Role of senior management:  4-M,

* The senior management of organizations plays a vital role in a development of testing peoples

* It is simply not enough to use words like "Quality is our top screen"

(or)

"People are out top priority"

* senior management people commitment has to be must translated into "visible action".

Some of the concrete works are

1. Ensuring a fair distribution of talented people in the testing arena.

2. Not allowing development engineers to look down upon Test engineers.

3. Dont treat test engineers as a low grade employers.

4. Encourage active & talented test peoples.

5. To confirm the Perfect job rotation amound development, testing & support functions.

iii) Role of Community:

* Important roles of testing community are

1. Give the freedom for tester, Tester should begin with a sense of pride in their job.

2. Test engineers need not be just followers of technology. They can take an active part in shaping & using new technology.

3. Sharing experience & knowledge to forum people.

4. Generally if learning the product and testing it is done at the same time then it gives inefficient testing.

6 Briefly discuss the testing team in organizational Structure [Nov/Dec-14]

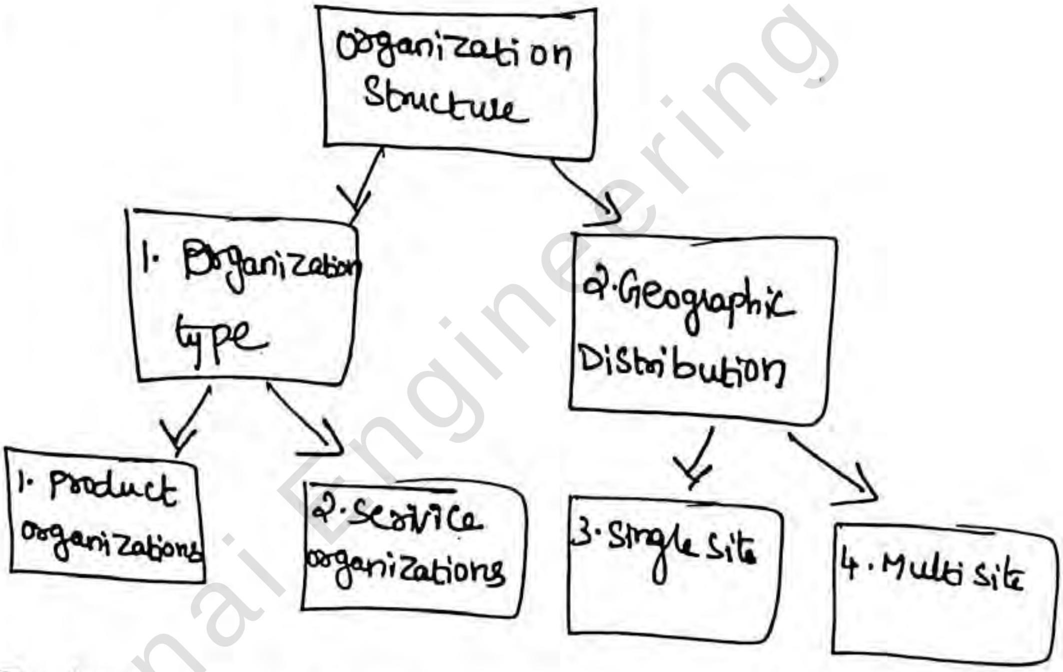
(nov/Dec 18,17)

Organizational Structures for testing team:

Dimensions of organization structure:

* organization structures gives a road map for the team members to take over their career paths

* organization structures is viewing upon two dimensions.



Product organizations

* produce software products and have a "womb to tomb" responsibility for the entire product.

Service organizations:

* Does not have the complete product responsibility.

* They provide testing services to other organizations

* These organizations are specilist for testing services only.

Geographic Distribution

Single site team

* All the team members are located 4 worked in one Place

Multi-site team

* Entire team is spreaded and worked across many locations.

Structure for single-product companies:

Structure for Multi-product Companies:

* In multi-product companies, every product is treated as separate business unit

* The organization of test teams in multi product companies is dicated largely by the following factors.

1. How tightly coupled the product are interms of technology
2. Dependence among various products
3. How synchronous are the release cycle of products.

To organize the testing teams for a multi product Company, there are many chances. They are

1. A Central "test think-tank/brain trust" team, which formulates the test strategy for the organization.
2. one test team for all the products
3. Different test teams for different types of tests
4. Different test teams for each product.
5. A Hybrid of all the above methods.

Testing teams as part of "CTO's office:

* Development & Testing are the Same level of importance in the concern.

* Testing team report directly to the CTO as a part to the design & Development teams.

Advantages:

1) Developing a Product architecture that is testable or Suitable for testing.

2) Testing team will have better product and technology Skills.

3) The testing team can get a clear Understanding of what design & architecture are built for and plan their tests accordingly.

4) The technical road map for Product development and test.

5) The CTO'S team can evolve a consistent, Cost effective Strategy for test automation.

* CTO deals only the architecture & test teams in this model

* The reasons to report to the CTO is that the team is Cross divisional & Cross functional.

In order that such a team reporting to the CTO be effective

1. It Should be Small team size

2) It should be team of equals

3. It should have organization - wide representation.

Single Test Team for all products:

* Single testing team is responsible for all type of Products.

* Single product team divided into multiple components

* Every component is developed by an independent team.

* The testing team can be made to report to the "CTO think-tank."

Testing teams organized by product:

*It is based on accountability, decision making and Scheduling

* Complete responsibility is assign to testing team

* Unit head organize the testing & development team.

Separate testing teams for different phases of testing

* Testing is a single & homogeneous activity

* Testing is done by several types of testing and different levels of Skills.

* Types of Testing - Black box testing, white box Testing, Regression testing.

* Levels of Skills - Test Script, Test case, Test report

* Each of these different types of tests may be carried out at different point of time.

Organization people to perform different types of testing

| Type of Test | Reports into | Rationale |
|---|---|---|
| White box Testing | Development team | It is inherently close to code, developers, develop & also runs the code |
| Black box Testing | Testing team | It is first level of external testing which a product may hold |
| Integration Testing | organization wide testing team | Combine multiple components, multiple products |
| System Testing | Product Management | It takes place by doing the testing in real time scenarios |
| Performance Testing | A Central bench marking group | Interproduct dependencies |
| Acceptance Testing | Product management | Proxy for customer acceptance |
| I18N Testing | Local teams (or) I18N teams | Knowledge of local languages and convention |
| Regression Testing | All test team | * Part of Smoke test<br>* continue to report into the Product testing teams |

Challenges of Global Teams

1) Cultural Challenges

2) Work allocation Challenges

3) parity across teams

4) Ability to track effectively

5) Dependence of Communication infrastructure

6) Time difference

# IT8076-SOFTWARE TESTING

## UNIT-5
## TEST AUTOMATION

## 2 MARKS & 16 MARKS WITH ANSWERS

# UNIT-V

## Part- A

1) Q What are Skills needed for test automation?

[May/June-16]
Nov, Dec 18

*Analytical Skills
*Logical thinking
* A global approach
* Curiosity & creativity
* Scripting & coding experience
*Continue to learning
*Trouble Shooting knowledge
*Functional knowledge.

2. Q State any two generic requirements for test tool and framework. /Mention the criteria for selecting test tool [May/June-16] [Nov/Dec-16]

1. No hard coding in the test Suite
2. Reuse of code for different types of testing test cases.

3. Mention the challenges in automation [Apr/May-17] (nov/Dec 19,17)

The Challenges in automation

1. Better test coverage
2. Testing the complete application
3 Mis understanding of company processes
4. Relationship with developers

5. Regressing testing

6. Lack of Skilled testers

7 Testing always under time Constraint

8. Understanding the requirements

9. Management commitment

10. Tool cost

11. Learning / Training of tools

12. Early payback, pay off etc

Q 4. Distinguish between milestone and deliverable. [NOV/DEC-16]

| Milestone | Deliverable |
|---|---|
| * Milestone are tangible events that are expected to occur at a certain time in the Projects life time. | * Deliverable is a measurable and tangible outcome of the project |
| * Milestones are set for the project managers and project team to ensure project progress | * Deliverables are the project result which is delivered to the clients |
| * They are not delivered to the clients | * They are delivered at the end of project phases, such as design and implementation |

Mention the types of testing amenable to automation [Apr/May-17]

The types of testing amenable to automation

1. Stress, reliability, scalability and performance testing

2. Regression tests

3. Functional tests

What is a metric? Give Examples for software metrics. [May/june-16]

* It is a quantitative measure of the degree to which a system, system component or process possesses a given attribute.

* It start with collecting a set of data, It helps for decision making.

Eg 1. How many defects are existed within the module?

2. How many test cases are executed per person?

State the advantages of Using automated tools for Software testing. [May/june-16]

1. Better test coverage

2. Perform some Special test

Eg Load, Stress etc

3. Easily setting Test preconditions

4. It helps in immediate testing.

8. write the different types of reviews practiced by Software industry. [Apr|May-15] / Mention the various types of Review [Nov|Dec-14]

1. Inspections
2. walkthrough

9. Mention the components of review plans [Nov|Dec-14]

- * Review Goals
- * Items being reviewed
- * Preconditions for the review
- * Roles, Team size, participants
- * Training requirements
- * Review Steps
- * Time requirements

what is the scope of automation. [Nov|De-14]

- * Identifying the types of testing amenable to automation
- * Find automating area's less prone to change
- * Automate tests that pertain to standards
- * Some management aspects

Explain the various generations of automation and the required skills for each [Apr/may-17]

Briefly discuss the overview of software test automation with Skill needed, scope and its Architecture [Nov/Dec-14] (nov/Dec 19,18)

Test Automation:

* Developing software to test the software is called test automation.

* Automation saves time as software can execute test cases faster than human do.

* The time thus saved can be used effectively for test engineers to

1) Develop additional test cases to achieve better coverage.

2) Perform some esoteric or specialized tests like ad hoc testing or

3) Perform some extra manual testing.

Skills needed for Automation:

* There are different "Generations of automation". The skills required for automation depends on what generation of automation the company is in or desires to be in the near future.

The automation of testing is broadly classified into three generations.

1) First generation - Record and Playback
2) Second generation - Data - Driven
3) Third generation - Action - Driven

1) First generation - Record and playback

* Record and playback avoids the repetitive nature of executing tests.

* A test engineeers records the sequence of actions by keyboard characters or mouse clicks and those recorded scripts are Played back later, in the Same order as they were recorded.

* It is simple to record and save the script This generation of tools has several disadvantages.

* The scripts may contain hard- Coded values thereby making it difficult to Perform general types of tests.

Eg: when a report has to use the current date and time it becomes difficult to use a recorded script.

The handling error condition is left to the testers and the played back scripts may require a lot of manual intervention to detect and correct error conditions.

* when the application changes, all the scripts have to be recarded, thereby increasing the test maintenance costs

2) Second generation - Data - Driven:

* This method helps in developing test scripts that generates the set of input conditions and corresponding expected output.

* This enables the tests to be repeated for different input and output conditions.

* The approach takes as much time and effort as the produt.

* However changes to application does not require the automated test cases to be changed as long as the input conditions and expected output are still valid.

* This generation of automation focuses on input and output conditions using the black box testing approach.

3) Third generation Action-Driven

* This technique enables a layman to create automation tests. There are no input and expected output conditions required for running the tests

* All actions that appear on the application are automatically tested, based on a generic set of controls defined for automation.

* The set of actions are represented as objects and those objects are reused.

* The users needs to specify only the operations (such as log in, download & so on) and everything else that is needed for those actions are automatically generated.

* Automation in third generation involves two major aspects

i) Test case automation and
ii) Framework design

* The skills needed for automation are classified into four levels for three generations as the third generation of automation introduces two levels of skills for development of test cases and framework.

Classification of Skills for automation

| Automation first generation | Automation second generation | Automation third generation | |
|---|---|---|---|
| Skills for test case automation | Skills for test case automation | Skills for test case automation | Skills for framework |
| Scripting languages | Scripting languages | Scripting languages | Programming languages |
| Record-playback tools usage | Programming languages | Programming languages | Design and architecture Skills for framework creation. |
| | knowledge of data generation techniques usage of the product under test | Design and architecture of the product undertest usage of the frame work | Generic test requirement for multiproducts |

Scope of Automation:

* The automation requirements define what needs to be automated looking into various aspects.

* The specific requirements can vary from product to product from situation to situation, from time to time.

* Tips for identifying the scope for automation.

* Identifying the types of testing and amenable to automation

* Automating areas less prone to change.
* Automate tests that pertain to standards
* Management aspects in automation.

Identifying the types of Testing Amenable to Automation

Certain types of tests automatically lend themselves to automation.

* Stress, reliability, scalability and performance testing.
* Regression tests
* Functional tests

1. Stress, reliability, scalability and performance testing:

* These types of testing require the test cases to be run from a large number of different machines for an extended period of time, such as 24 hours, 48 hours and so on.

* It is just not possible to have hundreds of users trying out the product day in and day out

* They may neither be willing to perform the repetitive tasks, nor will it be possible to find that many people with the required skill set.

2. Regression tests:

Regression test are repetitive in nature These test cases are executed multiple times during the Product development phase.

* Given the repetitive nature of the test Cases, automation will save significant time and effort in the long run.

3. Functional tests:

* These kinds of tests may require a complex Set up and thus require specialized Skill, which may not be available on an ongoing basis.

* Automating these once, using the expert Skill sets, can enable using less-skilled people to run these tests on an ongoing basis.

Automating Areas Less prone to change:

* In a product Scenario, the changes in requirements are quite common.

* In such Situation what to automate is easy to answer.

* Automation Should consider those areas where requirements go through lesser or no changes.

* Normally change in requirements cause Scenario and new features to be impacted, not the basic functionality of the product

Automate Tests that Pertain to standards

* one of the tests that products may have to undergo is compliance to standards.

Eg A product providing a JDBC interface should satisfy the standard JDBC tests.

* These tests undergo relatively less change. Even if they do change, they provide backend compatibility by which automated scripts will continue to run.

* Automating for standards provides a dual advantage. Test suites developed for standard are not only used for product testing but can also be sold as test tools for the market.

* A large number of tools available in the commercial market were internally developed for in-house usage.

* Hence, automating for standards creates new opportunities for them to be sold as commercial tools.

* Testing for Standards have certain legal and organization requirements.

* To Certify the software or hardware, a test suite is developed and handed over to different Companies.

* The certification suites are executed every time by the supporting organization before the release of software and hardware.

* This is called Certification testing.

## Management Aspects in Automation

* what to automate it takes into account the technical and management aspects as well as the long-term vision.

* Adequate effort has to be spent to obtain management Commitment

* The automated test cases need to be maintained till the product reaches obsolescence.

* Automation involves effort over an extended Period of time, management permissions are only given in phases and Part by part.

* Hence automation effort should focus on those areas for which management commitment exists already.

* Return on investment is another aspect to be considered seriously.

* Effort estimates for automation should give a clear indication to the management on the expected return on investment.

Explain the design and architecture for automation and outline the challenges [NOV/DEC-16]

Explain the design and architecture for software test automation [May/june-16] [Apr/May-15] (nov/Dec 19,17)

## Design and Architecture for Automation:

- * External Modules
- * Scenario and configuration file modules
- * Test cases and test framework modules
- * Tools and results modules
- * Report generator and reports/metrics modules

### External Modules:

* There are two modules that are external modules to automation TCDB and defect DB, all the test cases, the steps to execute them and the history of their execution are stored in the TCDB.

* The test cases, in TCDB can be manual or automated.

* The interface shown by thick arrows represents the interaction between TCDB and the automation framework only for automated test cases.

* Manual Test Cases do not need any interaction between the framework TCDB.

* Defect DB or defect database or defect repository contains details of all the defects that are found in various products that are tested in a particular organization.

* It contains defects and all the related information

(ie) when the defect was found, to whom it is assigned, what is the current status, the type of defects, its impact & so on.

* Design and architecture is an important aspect of automation.

* As in product development, the design has to represent all requirements in modules.

* Architecture for test automation involves two major heads

i) a test infrastructure that covers a test case database

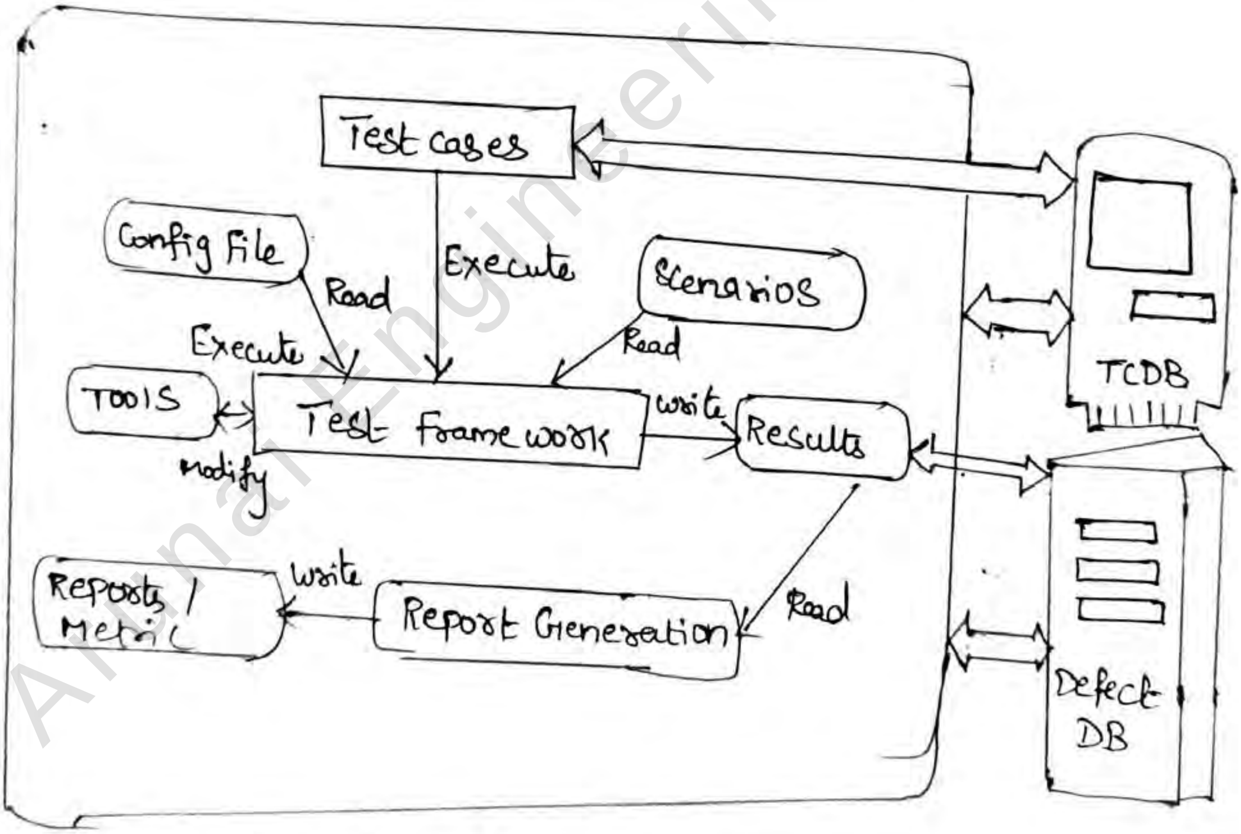
ii) a defect database or defect repository.

* Using this infastructure, the test framework provides a backbone that ties the selection and execution of test cases.

* For automated test cases, the framework can automatically submit the defects to the defect DB during execution.

* These external modules can be accessed by any module in automation framework not just one or more modules.

* The light shaded thick arrows (with lines) show specific interactions and dark shaded arrows (with lines) show multiple interactions.



Components of test Automation

* Thin arrows - internal interfaces & Direction of flow
* Thick arrows - External interfaces

# Scenario and Configuration File Modules

* Scenarios are nothing but information on "how to execute a particular test case".

* A configuration file contains a set of variables that are used in automation.

* The variables could be for the test framework or for other modules in automation such as tools and metrics or for the test suite or for a set of test cases or for a particular test case

* A configuration file is important for running the test cases for various execution conditions and for running the tests for various input and output conditions and states.

* The values of variables in this configuration file can be changed dynamically to achieve different execution, input, output and state conditions.

Test cases and Test framework Modules

* A test case means the automated test cases that are taken from TCDN and executed by the framework.

*Test Case is an object for execution for other modules in the architecture and does not represent any interaction by itself.

* A test framework is a module that combines "what to execute" and "how they have to executed".

*It picks up the specific test cases that are automated from TCDB and picks up the Scenarios and executes them.

* A test framework is considered the core of automation design. It subjects the test cases to different Scenarios.

Eg: If there is a scenario that requests a particular test case be executed for 48 hours in a loop

* The test framework executes those test cases in the loop and times out when the duration is met.

* The test framework contains the main logic for interacting , initiating and controlling all modules.

* A test framework can be developed by the organization internally or can be baught from the vendor.

Tools and Results Modules

* when a test framework Performs its operation these are a set of tools that may be required.

Eg: when test cases are stored as source code file in TCDB they need to be extracted and compiled by build tools.

* In order to run the compiled code certain runtime tools and utilities may be required.

Eg: IP Packets Simulators or user login Simulators or machine simulators may be needed.

* when a test framework executes a set of test cases with a set of Scenarios for the different values provided by the configuration file

* The results for each of the test case along with Scenarios and variable values have to be stored for future analysis and action

Report Generator and Report/Metrics Modules

* once the results of a test run are available, the next step is to prepare the test reports and metrics

* Preparing reports is complex and time-consuming effort and hence it should be part of the automation design.

* There should be customized reports such as executive report, which gives very high level status technical reports, which gives a moderate level of detail of the tests run.

* Detailed or debug reports which are generated for developers to debug the failed tests cases and the product.

* The periodicity of the reports is different such as daily, weekly, monthly and milestone reports, having reports of different levels of detail and different periodicities can address the need of multiple constituents and provide significant returns.

* The module that takes the necessary inputs and Prepares a formatted report is called a report generator.

* All the reports and metrics that are generated are stored in the reports/Metrics module of automation for future use and analysis.

what are metrics and measurements? Illustrate the types of Product metrics. [NOV/DEC-16]

Elaborate differenttypes of s/w metrics and measurements used [Apr/may-15]

## What ale Metrics and Measurements:

* Metrics derive information from raw data with a view to help indecision making

* Some of the areas that such information would shed light on are

1) Relationship between the data points

2) Any cause and effect correlation between the observed data points

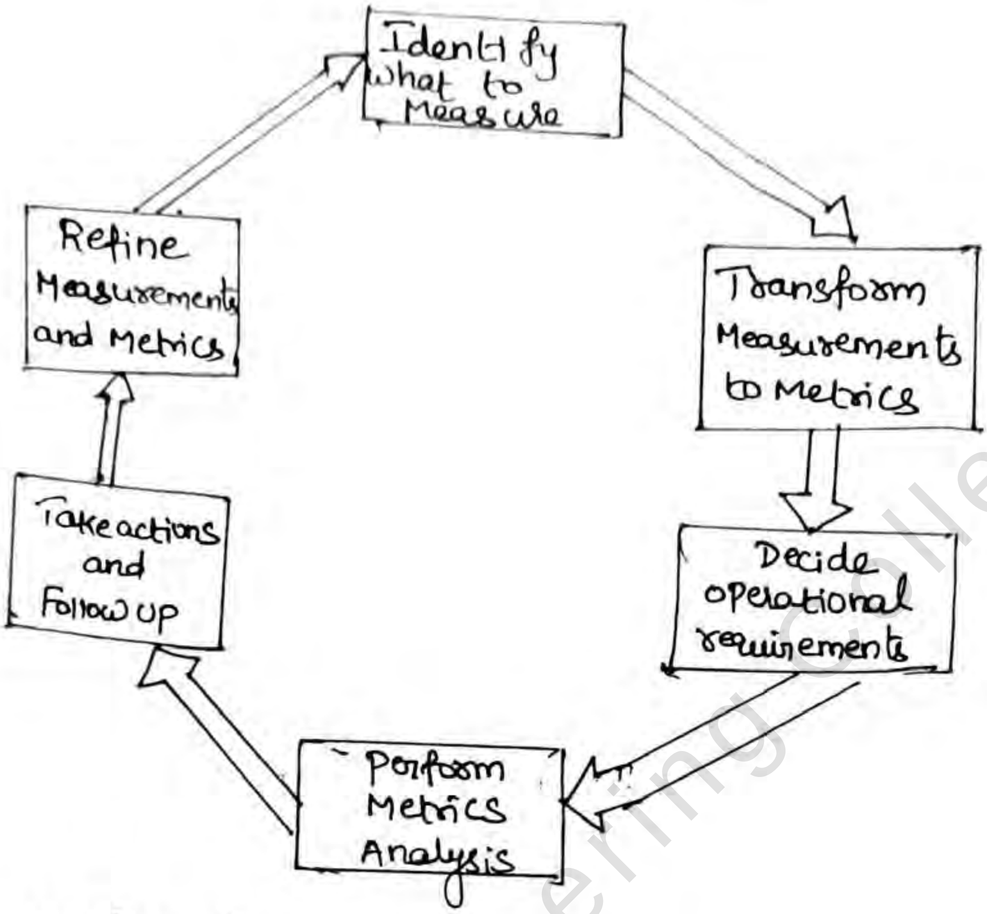
3) Any Pointers to how the data can be used for future planning and continiuous improvements.

* The metrics and analysis of metrics may convey the reason when data points are combined.

* Relating several data points and consolidating the result in terms of charts and pictures simplifies the analysis and facilities use of metrics for decision making.

* Collecting and analyzing metrics involves effort and several steps

# Steps in a Metrics program


Identify what to Measure
Transform Measurements to Metrics
Decide operational requirements
Perform Metrics Analysis
Take actions and Follow up
Refine Measurements and Metrics

* The first step involved in a metrics program is to decide what measurements are important and collect data accordingly.

Eg of Measurements, the effort spent on testing, number of defects and number of test cases.

While deciding what to measure the following aspect need to be kept in mind.

1) what is measured should be of relevance to what we are trying to achieve.

For testing functions, user obviously be interested in effort spent on testing number of test cases, ~~number of defects reported from test cases.~~

2) The entities measured should be natural and should not involve too many overheads for measurements.

3) what is measured should be at the right level of granularity to satisfy the objective for which the measurement is being made.

Data Drilling:

* The level of granularity of data obtained depends on the level of detail required by a specific audience.

* The measurements and the metrics derived from them will have to be at different levels for different people.

* An approach involved in getting the granular detail is called data drilling.

Why Metrics in Testing:

* Metrics are needed to know test case execution productivity and to estimate test completion date.

* The defect trend collected over a period of time gives a rough estimate of the defects that will come through future test cycles

* The defect fixing trend collected over a period of time gives another estimate of the defect-fixing capability of the team.

* The defect fixes may arrive after the regular test cycles are completed.

* These defect fixes will have to be verified by regressing testing before the product can be released.

* The measurements collected during the development and test cycle are not only used for release but also used for post-release activities.

* Looking at the defect trend for a period help in arriving at approximate estimates for the number of defects that may get reported post release.

* Metrics and their analysis help in preventing the defects proactively, saving cost & effort

* Metrics are used in resource management to identify the right size of product development teams.

* Metrics in testing help in Identifying

↳ when to make the release ↳ what to release

~~↳ whether the product is being released with known quality~~

4 Elaborate different types of software metrics and measurements used [Apr/may-15] [NOV/DEC-16]

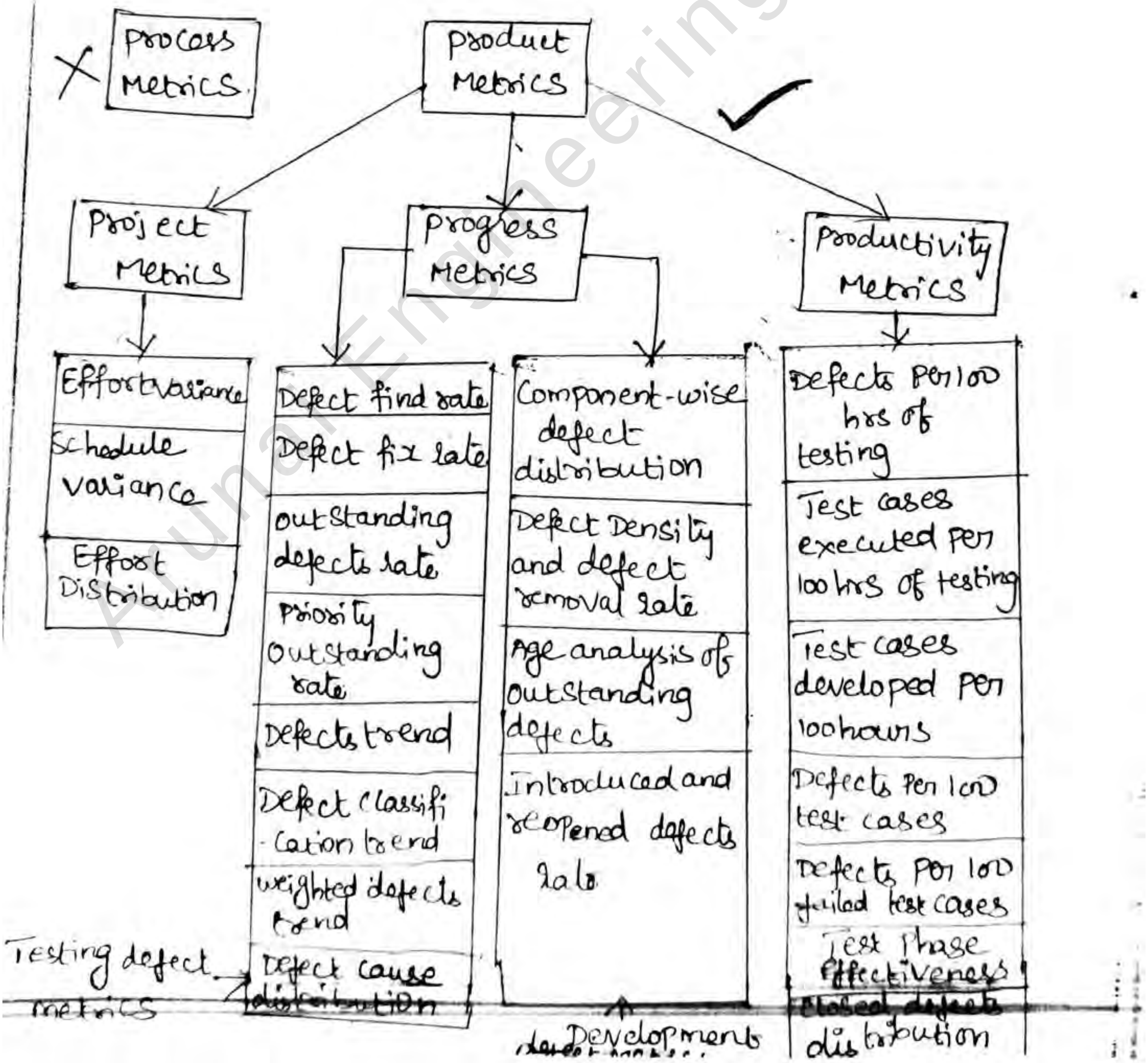
## Types of Metrics:

*Metrics can be classified into different types based on what they measure and what area they focus on.

*Metrics can be classified as

i) product metrics

ii) process metrics


Process Metrics
Product Metrics
Project Metrics
Progress Metrics
Productivity Metrics
Effort variance
Schedule variance
Effort Distribution
Defect find rate
Defect fix rate
outstanding defects rate
Priority outstanding rate
Defects trend
Defect classification trend
weighted defects trend
Defect cause distribution
Component-wise defect distribution
Defect Density and defect removal rate
Age analysis of outstanding defects
Introduced and reopened defects rate
Defects per 100 hrs of testing
Test cases executed per 100 hrs of testing
Test cases developed per 100 hours
Defects per 100 test cases
Defects per 100 failed test cases
Test Phase Effectiveness
Closed defects distribution
Testing defect metrics
Development

1.
Product metrics can be further classified as

1. Project metrics:

* A set of metrics that indicates how the project is planned and executed.

2. progress metrics:

* A set of metrics that tracks how the different activities of the project are progressing.

* The activities include both development activities and testing activities.

* progress metrics is monitored during testing phases.

* Progress metrics helps in Finding out the status of test activities and they are also good indicators of product quality.

* The defects that emerge from testing provide a wealth of information that help both development team and test team to analyze and improve.

* Progress metrics, for convenience, is further classified into test defect metrics and development defect metrics

3. Productivity metrics:

* A set of metrics that takes into account various productivity numbers that can be collected and used for planning and tracking testing activities.

* These metrics help in planning and estimating of testing activities.

Project Metrics:

* A typical project starts with requirements gathering and ends with product release.

* All the phases that fall in between these points need to be planned and tracked.

* In the planning cycle, the scope of the project is finalized.

* The project scope gets translated to effort estimate for each of the phases and activities by using the available productivity data available.

* This initial effort is called baselined effort

* As the project progresses and if the scope of the project changes or if the available productivity numbers are not correct, then the effort estimates are re evaluated again and this re-evaluated effort estimate is called revised effort.

The basic measurements that are very natural, simple to capture and form the inputs to the metrics in this section are

1) The different activities and the initial baselined effort and schedule for each of the activities this is input the beginning of the project/phase.

2) Calculating effort variance for each of the phase (as calculated by the formula below) provides a quantiative measure of the relative difference between the revised and actual effort.

3) The actual effort and time taken for the various activities, this is entered as and when the activities take place.

4) The revised estimate of effort and schedule these are re-calculated at appropriate times in the project life.

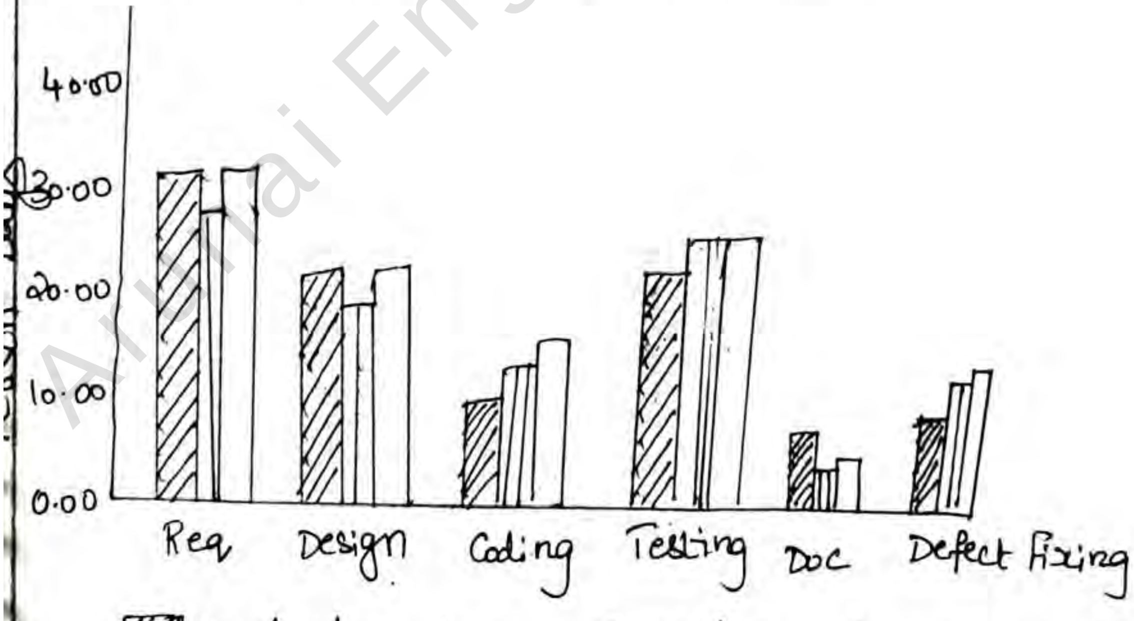
Effort variance (planned vs Actual)

* when the baselined effort estimates, revised effort estimates and actual effort are plotted together for all the phases of SDLC, it provides many insights about the estimation process.

* As different set of people may get involved in different phases, it is a good idea to plot these effort numbers phase-wise.

* If there is a substantial difference between the baselined and revised effort, it points to incorrect initial estimation.

Sample variance percentage by phase



Phase-wise Effort variation

Variance % = [(Actual effort - Revised Estimate)/Revised Estimate]*

| Effort | Req | Design | Coding | Testing | DOC | Defect Fixing |
|---|---|---|---|---|---|---|
| Variance % | 7.1 | 8.7 | 5 | 0 | 40 | 15 |

* If this is the case, the right parameters for variance calculation is the baselined estimate.

* In this case analysis should point out the problems in the revised estimation process.

Schedule variance (planned Vs Actual)

* Most software projects are not only concerned about the variance in effort, but are also concerned about meeting schedules.

* This leads us to the schedule variance metric.

* Schedule variance, life effort variance is the deviation of the actual schedule from the estimated schedule.

* There is one difference, depending on the SDLC model used by the project, several Phases could be active at the same time.

* Further the different Phases in SDLC are interrelated and Could share the Same set of individuals

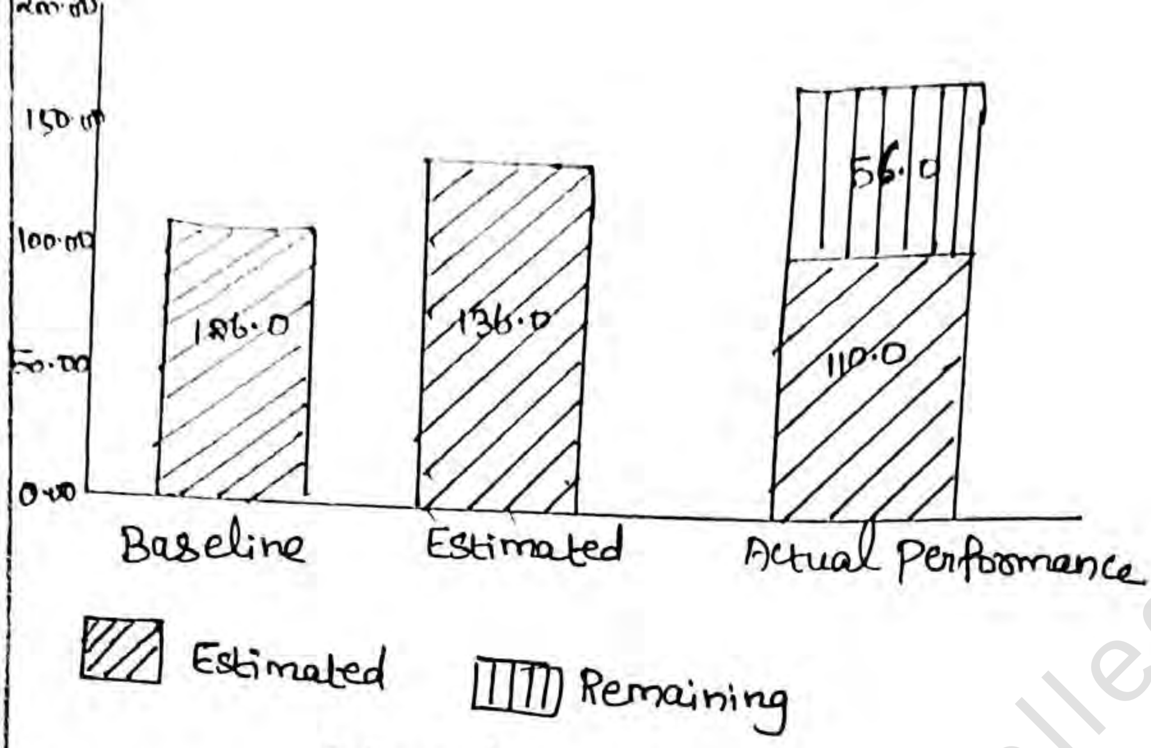
* Because of all these complexities involved, Schedule variance is calculated only at the overall project level, at specific milestones, not with respect to each of the SDLC phases

* Using the data in the chart, the variance Percent can be calculated Using a Similar formula.

* Considering the estimated Schedule and actual Schedule.

* Schedule variance is calculated at the end of every milestone to find out how well the project is doing with respect to the Schedule.

* To get a real Picture on Schedule in the middle of project execution, it is important to calculate "remaining days yet to be spent" on the project and plot it along with the "actual Schedule spent".



Schedule Variance

Interpretation of ranges of effort and Schedule Variation

| Effort Variance | Schedule Variance | Probable cause/result |
|---|---|---|
| Zero or acceptable variance | Zero Variance | A well-executes Project |
| Zero or acceptable variance | Acceptable Variance | Need Slight improvements in effort/ schedule estimation |
| unacceptable variance | zero or acceptable Variance | Under estimation, needs further analysis |
| unacceptable variance | Unacceptable variance | under estimation of both effort and Schedule |
| Negative Variance | Zero or acceptable variance | over estimation and schedule both effort and Schedule estmation need improvement |
| Negative variance | Negative variance | over estimation and over schedule both effort and schedule estimation need improvement |

*Remaining days yet to be spent can be calculated by adding up all remaining activities.

*If the remaining days yet to be spent on project is not calculated and plotted, it does not give any value to the chart in the middle of the project, because the deviation cannot be inferred visually from the chart.

*The remaining days in the schedule becomes zero when the release is met.

Effort Distribution Across Phases:

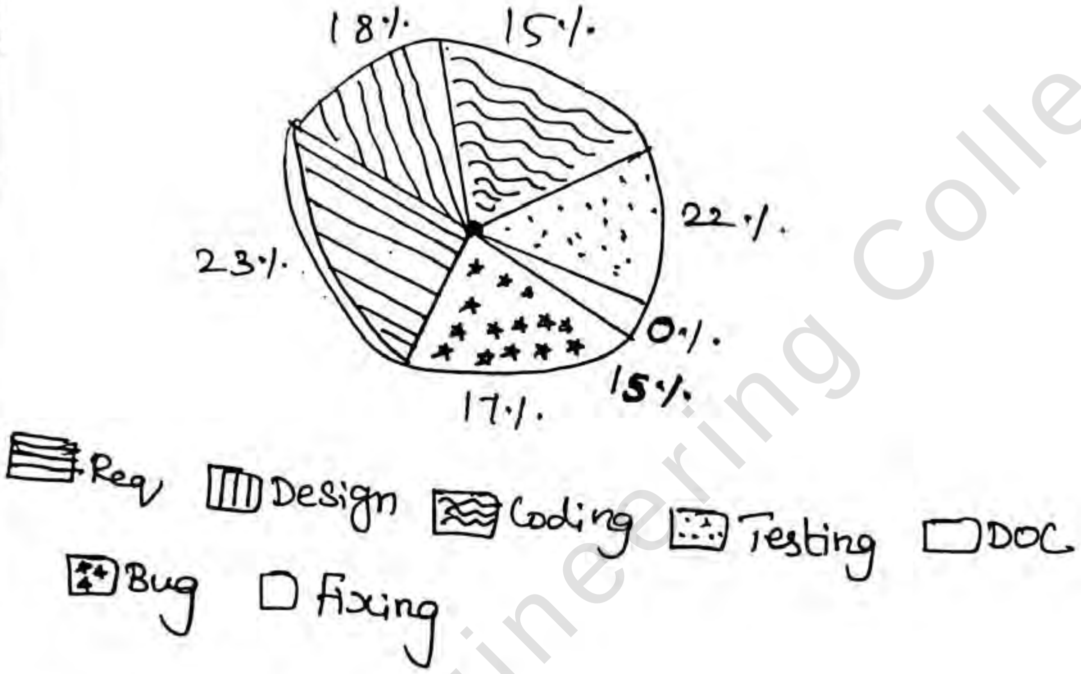
*Adequate and appropriate effort needs to be spent in each of the SDLC phase for a quality product release.

*Variance calculation helps in finding out whether commitments are met on time and whether the estimation method works well.

*In addition, some indications on product quality can be obtained if the effort distribution across the various phases are captured and analyzed.

Eg1: Spending very little effort on requirements may lead to frequent changes but one should also leave sufficient time for development and testing phase.

2. Spending less effort in testing may cause defects to crop up in the customer Place but Spending more time in testing than what is needed may make the product lose the market window.



* Mature organizations spent at least 10-15% of the total effort in requirements and approximately the same effort in the design phase.

* The effort percentage for testing depends on the type of release and amount of charge to the Existing code base and functionality.

* Typically, organizations spend about 20-50% of their total effort in testing.

5. Explain the different types of Test defect metrics under Progress metrics based on what they measure and what area they focus on [Apr/may-17] [May/june-16] 5-17

## Progress Metrics: (apr/may 18) (nov/Dec 17)

* Any project needs to be tracked from two angles

1. How well the project is doing with respect to effort and Schedule.

2. Equally important angle is to find out how well the product is meeting the quality requirements for the release.

* Defects get detected by the testing team and get fixed by the development team.

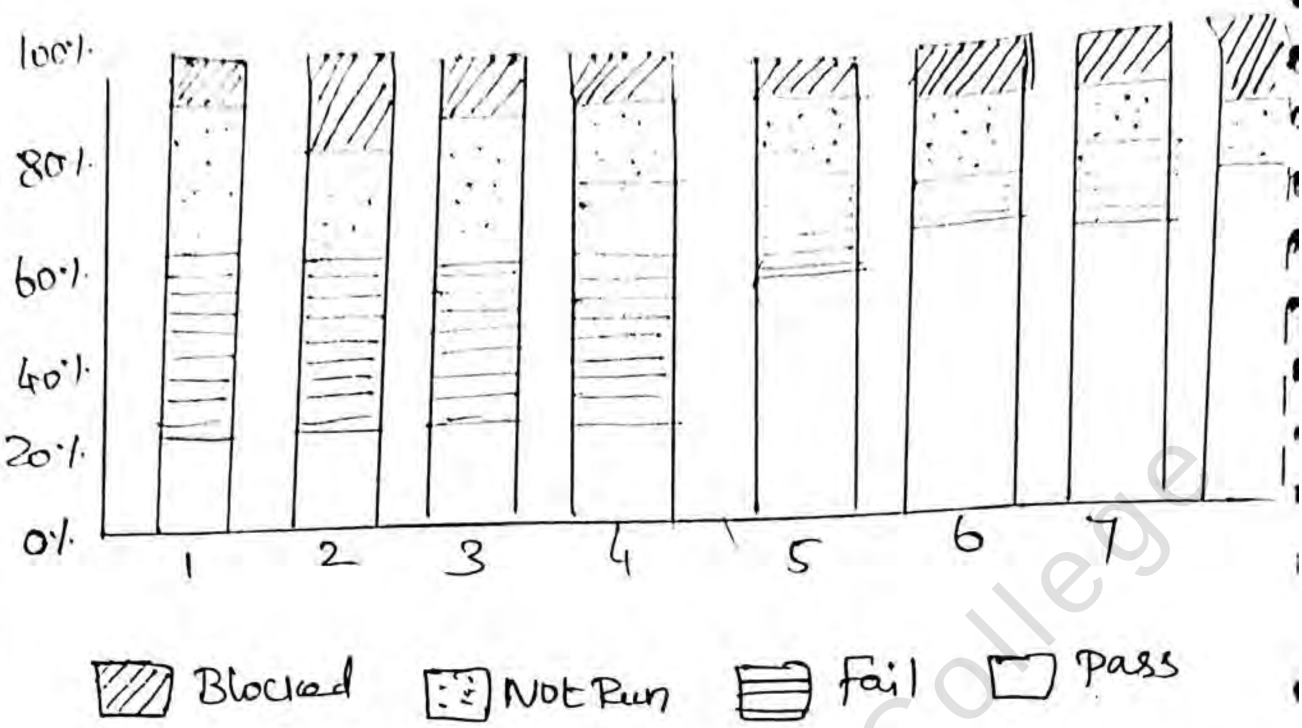
* Defect metrics are further classified into test defect metrics - which help the testing team in analysis of product quality and testing.

* Development defect metrics - which help the development team in analysis of development activities

Two parameters that determine product quality and its assessment

1. How many defects have already been found
2. How many more defects may get Unearthed.

* If only 50% of testing is complete and if 100 defects are found, assuming that the ~~defects are Uniformly distributed over the products~~



Progress of Test Case Execution

* The progress chart gives the pass rate and fail rate of executed test cases, pending test cases and test cases that are waiting for defects to be fixed.

* Representing testing progress in this manner will make it is easy to understand the status and the further analysis.

* Another perspective from the chart is that the pass percentage increases and fail percentage decreases showing the positive progress of testing and product quality.

* A Scenario represented by such a progress Chart Shows that not only is testing progressing well, but also that the product quality is improving.

* If, on the other hand, the chart had shown a trend that as the weeks progress, the "not run" cases are not reducing in number or "blocked" cases are increasing in numbers or "pass" cases are not increasing

* It would clearly point to quality problems in the product that prevent the product from being ready for release.

## Test Defect metrics:

* The next set of metrics help us understand how the defects that are found can be used to improve testing and product quality.

* Some organizations classify effects by assigning a defect priority Eg P1, P2, P3 & so on.

* The priority of a defect provides a management Perspective for the order of defect fixes.

Eg A defect with priority P1 indicates that it Should be fixed before another defect with priority P2.

* Some organizations use defect severity levels Eg S1, S2, S3 & soon.

* The Severity of defects provides the test team a perspective of the impact of the defect in product functionality.

Eg A Defect with severity level

S1 = Either the major functionality is not working or the software is crashing

S2 = Mean a failure or functionality not workin (

Defect priority and Defect severity - Sample Interpretation:

| priority | what it means |
|---|---|
| 1 | Fix the defect on highest priority<br>Fix it before the next build |
| 2 | Fix the defect on high priority before next test cyc |
| 3 | Fix the defect on moderate priority when time permits before release |
| 4. | Postpone this defect for next<br>Release or live with this defect. |
| severity | what it means |
| 1. | The basic product functionality Failing or product crashes |
| 2. | Unexpected error condition or a functionality not working |
| 3. | A minor functionality is failing or behaves differently than expected. |
| 4 | cosmetic issue and no impact on the issues |

A common defect definition and classification

| Defect classification | what it means |
|---|---|
| Extreme | Product Crashes or unusable<br>Needs to be fixed immediately |
| Critical | Basic functionality of the product not working.<br>Needs to be fixed before next test cycle Starts |
| Important | Extended functionality of the product networking.<br>Does not affect the progress of testing<br>Fix it before the release. |
| Minor | Product behaves differently<br>No impact on the test team or customers |
| Casmetic | Minor hesitant<br>Need not be fixed for this release |

Defect Find Rate:

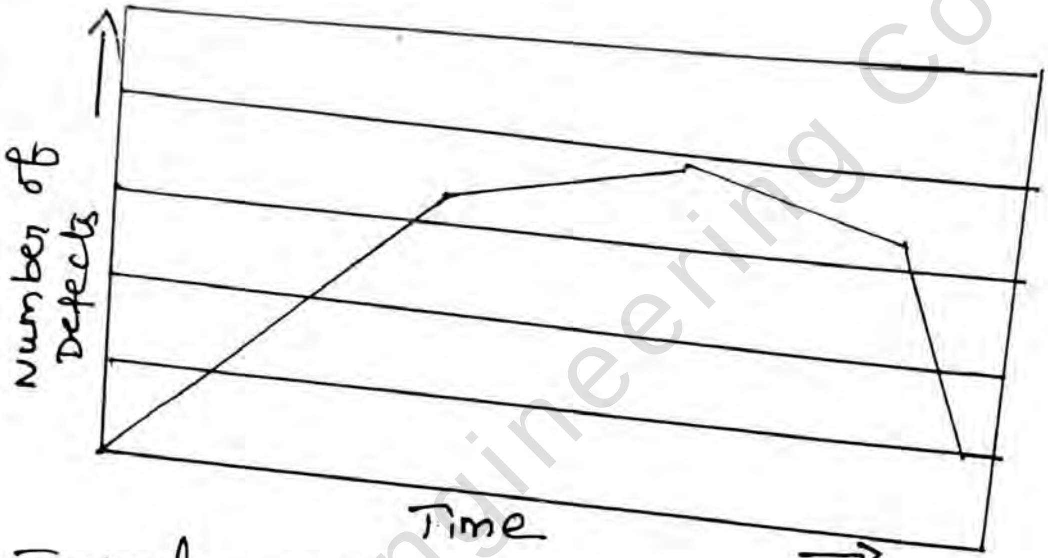
* The purpose of testing is to find defects early in the test cycle.

* when tracking and plotting the total number of defects found in the products at regular intervals (daily or weekly) from beginning, to end of a product development cycle, it may show a pattern for defect arrival

* The idea of testing is to find as many ~~defects as possible early in the cycle.~~

* once a majority of the modules become available and the defects that are blocking the tests are fixed.

* The defect arrival rate increases. After a Certain period of defect fixing and testing. the arrival of defects tends to slow down and a continuation of that trend enables prod release.



Typical pattern finding defects in a product

Defect Fix Rate:

* The purpose of development is to fix defects as soon as they arrive.

* If the goal of testing is to find defects as early as possible, it is natural to expect that the goal of development should be to fix defects as soon as they arrive.

* If the defect fixing curve is in line with defect arrival a "bell curve" as shown above figure.

2, 5M)

Outstanding Defects Rate

* The number of defects outstanding in the product is calculated by subtracting the total defects fixed from the total defects found in the product

* In a well-executed project, the number of outstanding defects is very close to zero all the time during the test cycle.

* The defects need to be fixed as soon as they arrive and defects arrive in the pattern of bell curve.

* If the defect fixing pattern is constant like a straight line, the outstanding defects will result in a ball curve again

Priority Outstanding Rate:

* Having an eye on the find rate, fix rate and outstanding defects are no enough to give an idea of the sheer quantity of defects

* The modification to the outstanding defects rate curve by plotting only the high priority defects and filtering out the low-Priority effects is called priority outstanding defects

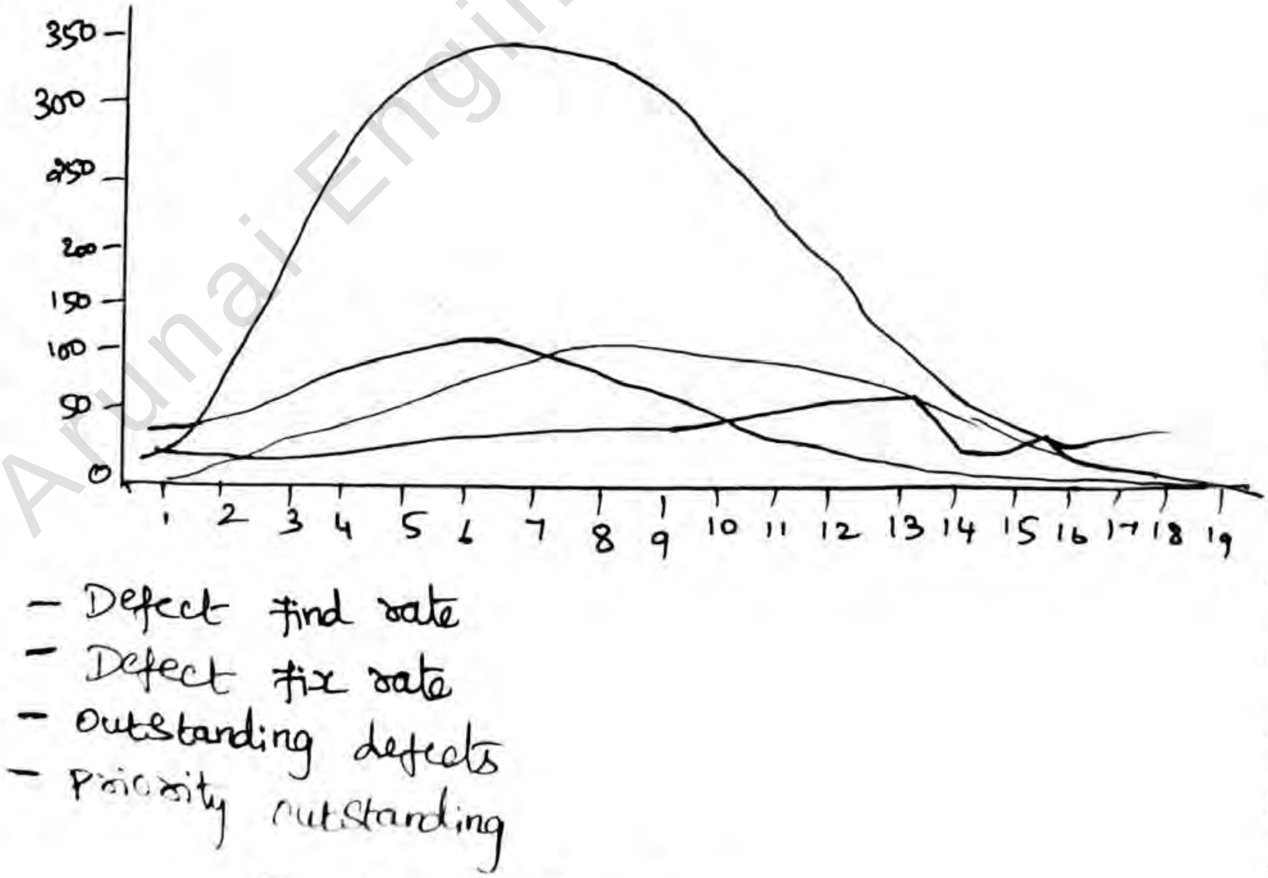
i) High priority defects - Require a change in design or architecture.

If they are found late in the cycle, the release may get delayed to address the defect.

ii) Low-priority defect - It is close to the release date and it requires a design change a likely decision of the management would be not to fix that defect.

Defect Trend:

* The effectiveness of analysis increases when several perspective of find rate, fix rate, outstanding, and prior outstanding effects are combined.



Defect Trend

1) The find rate, fix rate, outstanding defects and Priority outstanding follow a bell curve pattern indicating readiness for release at the end of the 19th week.

2) A sudden downward movement as well as upward spike in defect fixes rate needs analysis.

3) There are close to 75 outstanding defects at the end of the 19th week.

* By looking at the priority outstanding which shows close to zero defects in the 19th week, it can be concluded that all outstanding defects need analysis before the release.

4) Defect fix rate is not in line with outstanding defect rate.

* If defect fix rate had been improved, it would have enabled a quicker release cycle (reduced the schedule by four to five weeks) as incoming defects from the 14 week were in control.

5) Defect fix rate was not at the same degree of defect find rate. Find rate was more than the fix rate till the 10th week.

* Making find rate and fix rate equal to each other would have avoided the outstanding defects peacing from the 4th to 16th week.

6) A Smooth porpority outstanding rate suggest that priority defects were closely tracked and fixed.

Defect classification Trend:

* Providing the Perspective of defect classification in the chart helps in finding out release readiness of the product.

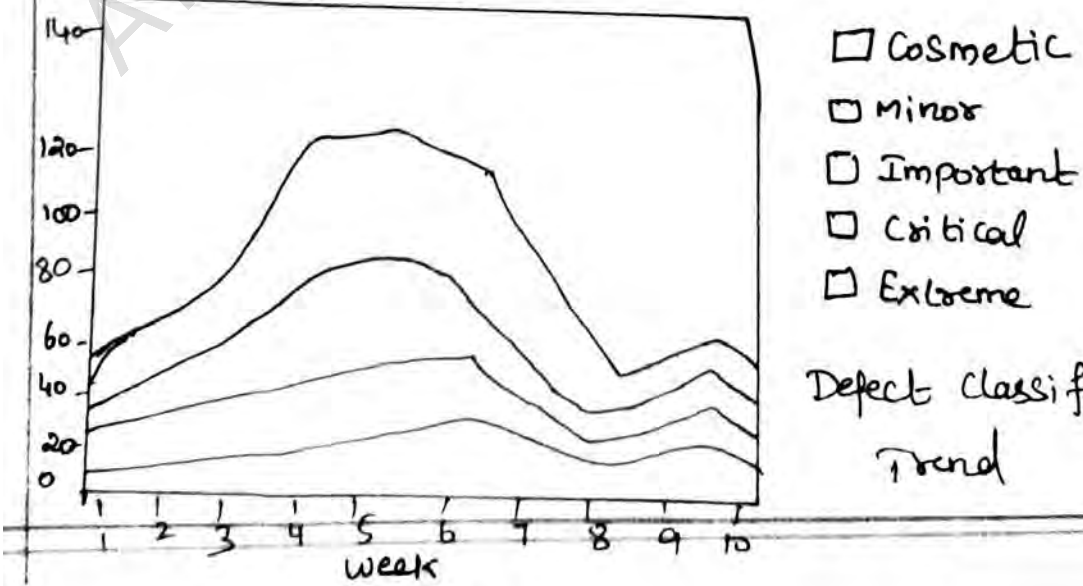
* Some of the data drilling or chart analysis needs further information on defects with respect to each classification of defects, extreme, critical, important, minor and cosmetic

* when talking about the total number of outstanding defects, some of the questions that can be asked are

✓ How many of them are extreme defects?

✓ How many are critical

✓ How many are important?



Defect Classification Trend

*Explain the various types of progress and productivity metrics based on what they measure and what area they focus on. 5-22 [may/june-16]

## Productivity Metrics:-

* Productivity metrics Combine several measurements and parameters with effort spent on the product.

1) Estimating for the new release

2) Finding out how well the team is progressing, Understanding the reasons for (both positive and negative) variations in results.

3) Estimating the number of defects that can be found.

4) Estimating release date and quality

5) Estimating the cost involved in the release

## Defects per 100 Hours of Testing:

* Program testing can only prove the presence of defects, never their absence.

* It is reasonable to conclude that there is no end to testing and more testing may reveal more new defects.

* If incoming defects in the product are reducing it may mean various things.

~~1. Testing is not effective~~

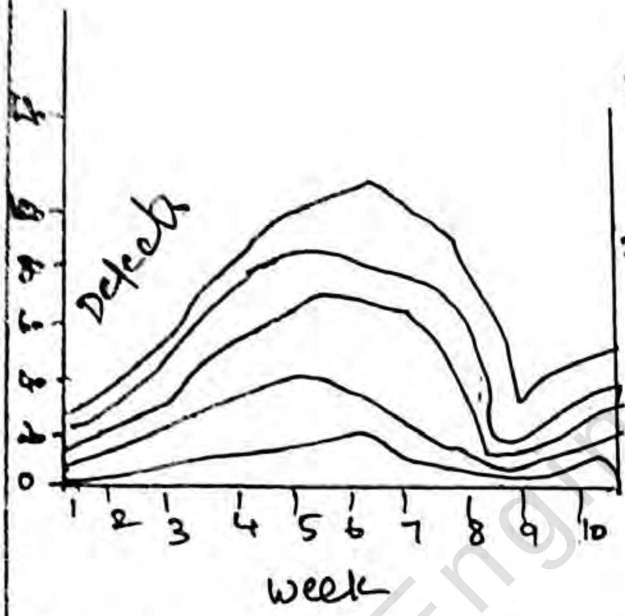
2) the Quality of the product is improving

3) Effort spent in testing is falling.

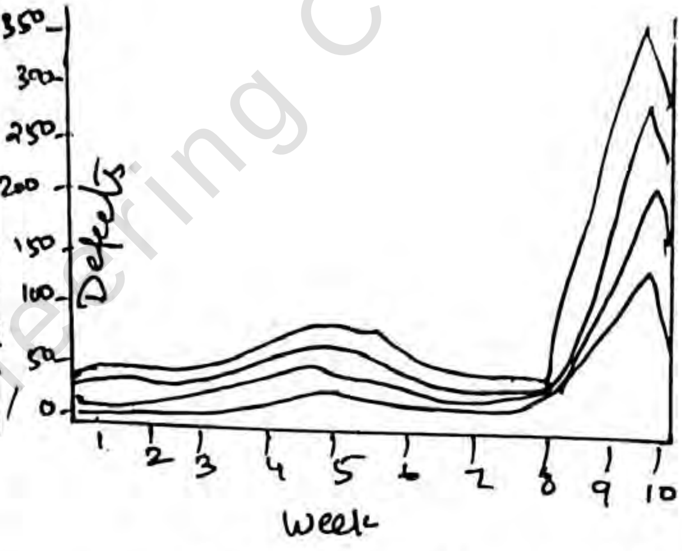
Defects per 100hours of testing

$$= \left( \frac{\text{Total defects found in the product for a period}}{\text{Total hours spent to get those defects}} \right) *$$

Defect classification trend.



Defects per 100 hours of testing



Defects per 100 hours of testing

* The Chart produced a bell curve indicating readiness for the release.

Test cases executed per 100 hours of testing:

* The number of test cases executed by the test team for a particular duration depends on team productivity and quality of products

Test cases executed per 100 hours of testing

$$= \left( \frac{\text{Total test cases executed for a period}}{\text{Total hours spent in test execution}} \right) * 100$$

Test cases developed per 100 hours of testing:

* Both manual execution of test cases and automating test cases require estimating and tracking of productivity numbers.

* The formula for test cases developed uses the count corresponding to added/modified and deleted test cases.

Test cases developed per 100 hours of testing

$$= \left( \frac{\text{Total test cases developed for a period}}{\text{Total hours spent in test case development}} \right) * 100$$

Defects per 100 test cases:

* The goal of testing is to find out as many defects as possible, it is appropriate to measure the "defect yield" of tests (ie) how many defects get uncovered during testing.

* This is a function of two parameters

1) The effectiveness of the tests in uncovering defects

2) The effectiveness of choosing tests that are capable of uncovering defects

Defects Per 100 test cases

$$= \left( \frac{\text{Total defects found for a period}}{\text{Total test cases executed for the same period}} \right) * 100$$

Defects Per 100 failed test cases

* Defects per 100 failed test cases is a good measure to find out how granular the test cases are. It indicates

1) How many test cases need to be executed when a defect is fixed.

2) what defects need to be fixed so that an acceptable number of test cases reach the pass state and

3) How the fail rate of test cases and defects affect each other for release readiness analysis

Defects Per 100 failed test cases

$$= \left( \frac{\text{Total defects found for a period}}{\text{Total test cases failed due to those defects}} \right) * 100$$

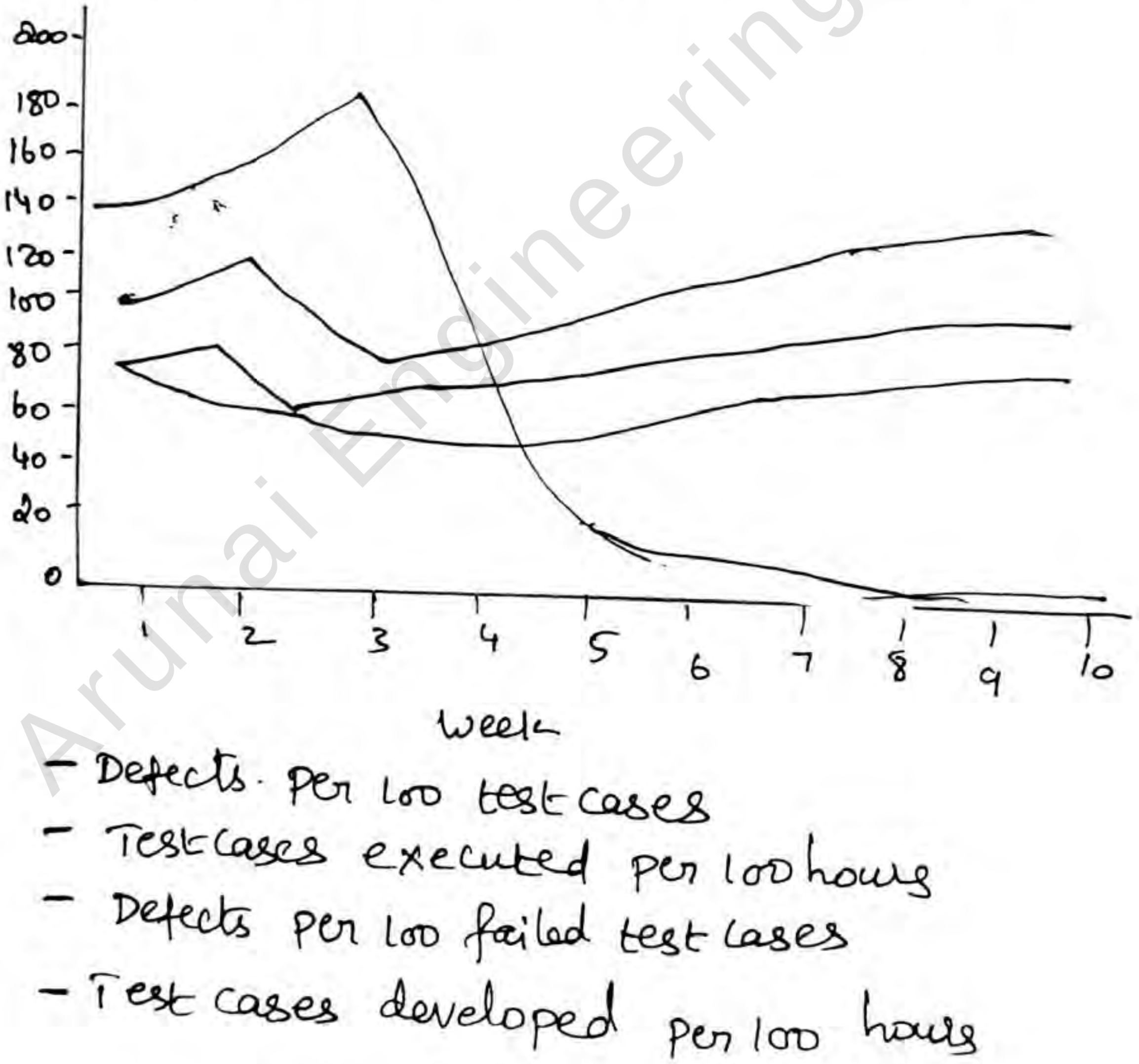
The following observations can be made by looking the chart

1) Defects Per 100 test cases showing a downward trend suggests product readiness for release

2) Test cases executed per 100 hours on upward trend suggests improved productivity and product quality (week 3 data needs analysis)

3) Test cases developed per 100 hours showing a slight upward movement suggests improved Productivity

4) Defects per 100 failed test cases in a band of 80-90 suggests eaual number of test cases to be verified when defects are fixed.



Productivity Metrics

Explain the types of reviews [Nov/Dec-15]

(apr/may 18)

## Types of Reviews:

* Reviews can be formal or informal. They can be technical or managerial.

* Managerial reviews usually focus on Project management and Project status

↳ Verify that Software artifact meets its specification

↳ To detect defects and

↳ Check for compliance to standards

* The Colleague requesting the review receives feed back about one or more attributes of the reviewed Software artifact.

* Informal reviews are an important way for Colleagues to communicate and get peer input with respect to their work.

Two major types of reviews

1) Inspections as a type of technical review

2) walkthrough

1).Inspections as a type of technical Review:

* .Inspections are a type of review that is formal in nature and require preview preparation on the part of the review team.

* The responsibility for initating and carrying through the steps belongs to the inspection leader (or moderator) who is usually a member of the technic staff or the software quality assurance team.

* The inspection leader plans for the inspection, sets the date, invites the participants distribute the required documents, runs the inspection meeting, appoints a recorder to record results and monitors the follow up period after review.

* The inspection participants address each item on the checklist.

* The recorder records any discrepancies, misunderstanding, errors and ambiguities, in genera any problems associated with an item.

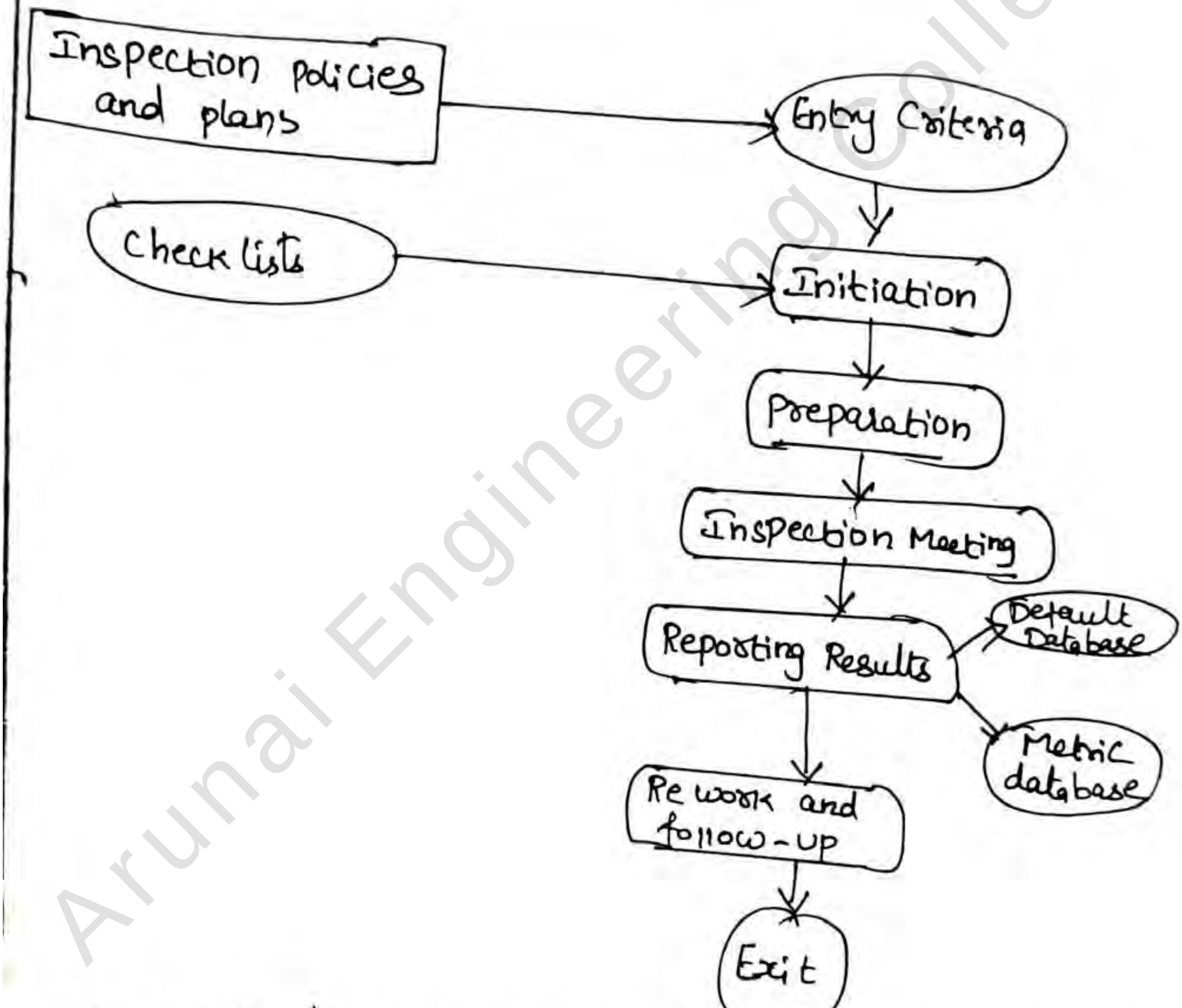
* The completed checklist is part of the review.

* The inspection process begins when inspection precondition are met as specified in the inspection policies, procedures and plans.

* The inspection leader announces the inspection meeting and distributes the items to be inspected the checklist

* Any other auxiliary material to the participants usually a day or two before the scheduled meeting

Steps in the Inspection process



* Participants must do their homework and study the items and the checklists

* Attention is paid to issue related to quality, adherence to standards, testability, traceability and satisfaction of the users/clients requirements

* The inspection Process requires a formal followp-up process.

* Rework sessions should be scheduled as needed and monitored to ensure that all problems identified at the inspection meeting have been addressed and resolved.

* only when all problems have been resolved and the item is either <u>re inspected</u> by the group or the moderator is the inspection process completed.

## Walkthrough as a Type of Technical Review:

* Walk through are a type of technical review where the producer of the reviewed material serves as the review leader and actually guides the progression of the review.

* walkthrough have traditionally been applied to <u>design and code.</u>

* The whole group "plays computer" to step through an execution lead by reader or presenter

* This is a good opportunity to "pretest" the design or code.

* If the presenter gives a skilled presentation of the material, the walkthrough participants are able to build a comprehensive mental model of the detailed design or Code are able to both evaluate its quality and detect defects.

* walk through may be used for material other than Code. Eg Data Descriptions, reference manuals or Even Specifications.

# IT8076-SOFTWARE TESTING

# PREVIOUS YEAR QUESTION PAPER

# Question Paper Code : 91760

B.E./B.Tech. DEGREE EXAMINATIONS, NOVEMBER/DECEMBER 2019
Sixth/Seventh Semester
Computer Science and Engineering
IT 6004 – SOFTWARE TESTING
(Common to Information Technology)
(Regulations 2013)
(Also Common to PTIT6004 – Software Testing for B.E. (Part-Time) –
Sixth Semester Computer Science and Engineering – Regulations 2014)

ime : Three Hours Maximum : 100 Marks

Answer ALL questions

PART – A **(10×2=20 Marks)**

1. Mention the role of process in Software Quality.
2. Mention the various sources of defects.
3. What is Error, Defect, Bug and Failure ?
4. What are the components of COTS ?
5. Compare Black box testing and White box testing.
6. Why it is important to design test harness for testing ?
7. What are the issues in testing Object Orient Systems ?
8. List the skills needed by a test specialist.
9. What are the challenges in test automation ?
10. Define progress metrics and process metrics.

PART – B **(5×13=65 Marks)**

11. a) Elaborate the software testing principles and summarize the tester role in software development organization.

(OR)

b) Explain Testing Maturity Model (TMM) and the test related activities that should be done for V-Model Architecture.

12. a) Discuss in detail about static testing and structural testing. Write the difference between these two testing concepts.

(OR)

b) Explain about the various black box test cases using equivalence class partitioning and boundary value analysis to test a module.

13. a) Explain briefly about the various types of system testing.

(OR)

b) Explain about the :

i) Unit test planning **(7)**

ii) Configuration testing and its objectives. **(6)**

14. a) i) Discuss in detail about various skills needed for a test specialist. **(7)**

ii) Write about Mutation Testing with an example. **(6)**

(OR)

b) Explain the components of test plan in detail.

15. a) Discuss the design and architecture for automation with neat sketch.

(OR)

b) Write short notes on :

i) Classification of automation testing.

ii) Scope of automation.

PART – C **(1×15=15 Marks)**

16. a) Explain the importance of security testing and explain the consequences of security breaches, also write the various areas which has to be focused during security testing.

(OR)

b) Explain in detail processing and monitoring of the defects with defect repository.

**Question Paper Code : 20728**

B.E./B.Tech. DEGREE EXAMINATION, NOVEMBER/DECEMBER 2018.

Sixth/Seventh Semester

Information Technology

IT 6004 — SOFTWARE TESTING

(Common to Computer Science and Engineering)

(Regulations 2013)

(Also common to PTIT 6004 – Software Testing for B.E. (Part-Time)
Sixth Semester – Computer Science and Engineering – Regulations 2014)

Time : Three hours Maximum : 100 marks

Answer ALL questions.

PART A — (10 × 2 = 20 marks)

1. Mention the role of process in software quality.

2. What is meant by Feature Defects?

3. Compare black box and white box testing.

4. Define COTS Components.

5. List the levels of Testing.

6. What is Regression testing?

7. Mention the duties of component – wise testing teams.

8. What is the need of Test Incident Report?

9. What is the need for Automated Testing?

10. Define Progress Metrics.

PART B — (5 × 13 = 65 marks)

11. (a) Discuss in detail about Software testing principles. (13)

Or

(b) (i) Write short notes on Origins of defects. (7)

(ii) Describe about Tester Support for Developing a Defect Repository. (6)

12. (a) Explain about the following methods of black box testing with example.

(i) Equivalence class partitioning.

(ii) Boundary value analysis. (13)

Or

(b) Discuss in detail about static testing and structural testing. Also write the difference between these testing concepts. (13)

13. (a) State Unit Test and describe about planning and Designing of Unit Test. (13)

Or

(b) Explain elaborately about the various types of system testing. (13)

14. (a) Explain the concepts of test planning in detail. Also mention the way of defining test plan. (13)

Or

(b) Describe the concepts of building a test group. (13)

15. (a) Write short notes on following. (13)

(i) Classifications of automation testing.

(ii) Scope of an automation.

Or

(b) Discuss in detail about selecting the test tool in test automation. (13)

PART C — (1 × 15 = 15 marks)

16. (a) Case Study : Several kinds of tests for a web application.

Abstract :

A UK based company entrusted us to test this project. It's a web application for government to collect data and calculate them to prioritize all the tasks.

Description :

This client is from Hertfordshire in UK, the project is an application for the government. In fact, it includes two parts: web site for data collection and presentation purpose, in parallel a windows application for administration purpose. Here the task is ensuring the quality of the web application, includes many aspects, such as function correctness, performance acceptance, UI appropriateness, and so on. Moreover, for testing function, we had to use the windows application to edit users, services and other data.

The client only gave us the software requirement specification and the applications tested, there wasn't any test plan, test strategy, test cases, even test termination criterion. On the one hand, we had to spend much time in communicating with client to make clearly about some important points; on the other hand, we had to get familiar with the application via operating it and reading requirements.

Then, how to improve the efficiency of regression test?

Or

(b) Illustrate various components of Test plan with an example. (15)

**Question Paper Code : 41285**

05/05/18

(AN)

B.E./B.Tech. DEGREE EXAMINATION, APRIL/MAY 2018
Sixth/Seventh Semester
IT6004 – SOFTWARE TESTING
Common to : B.E. Computer Science and Engineering/B.Tech. Information Technology
(Regulations 2013)

Time : Three Hours    Maximum : 100 Marks

Answer ALL questions

PART – A    **(10×2=20 Marks)**

1. List out the levels of the testing maturity model.

2. Define Test Oracle.

3. What are the factors affecting less than 100% degree of coverage ?

4. Write the formula for cyclomatic complexity.

5. What is the advantage of Bottom up integration ?

6. Give the examples of security testing.

7. Define a Work Breakdown Structure (WBS).

8. What is the function of Test Item Transmittal Report or Locating Test Items ?

9. What are the goals of Reviewers ?

10. What is Walk Through ?

PART – B    **(5×13=65 Marks)**

11. a) Give overview of the Testing Maturity Model (TMM) and the test related activities that should be done for V-model architecture. **(13)**

(OR)

b) Elaborate on the principles of software testing and summarize the tester role in software development organization. **(13)**

12. a) Demonstrate the various black box test cases using equivalence class partitioning and boundary value analysis to test a module for payroll system. (13)

(OR)

b) Explain about state transition testing. (13)

13. a) i) Write the importance of security testing and explain the consequences of security breaches, also write the various areas which has to be focused on during security testing. (7)

ii) State the need for integration testing in procedural code. (6)

(OR)

b) i) Explain about the unit test planning. (7)

ii) Explain about configuration testing and its objectives. (6)

14. a) Explain the components of test plan in detail. (13)

(OR)

b) i) List and explain the skills needed by a test specialist. (7)

ii) Name the reports of test results and the contents available in each test reports. (6)

15. a) Discuss the types of review. Explain various components of review plans. (13)

(OR)

b) Narrate about the metrics or parameters to be considered for evaluating the software quality. (13)

PART – C (1×15=15 Marks)

16. a) Explain in detail processing and monitoring of the defects with defect repository.

(OR)

b) Explain the organizational structures for testing teams in single product companies.

# Question Paper Code : 50754

B.E./B.Tech. DEGREE EXAMINATION, NOVEMBER/DECEMBER 2017
Sixth/Seventh Semester
Computer Science and Engineering
IT6004 – SOFTWARE TESTING
(Common to – Information Technology)
(Regulations 2013)

Time : Three Hours Maximum : 100 Marks

Answer ALL questions

PART – A (10×2=20 Marks)

1. What are the objectives of software testing ?

2. Define Test Bed.

3. Compare black box and white box testing.

4. What are the basic primes that are used in a structured program ?

5. Define Unit Test. Give an example.

6. Why is it important to design test harness for testing ?

7. List the various skills needed by a test specialist.

8. What is the role of Test Summary Report.

9. What are the challenges in test automation ?

10. What are the uses of walkthrough ?

PART – B (5×16=80 Marks)

11. a) i) Explain various design defects with suitable examples. (8)

ii) Analyse tester's role in software development organization. (8)

(OR)

b) Illustrate with example the principles of software testing. (16)

12. a) Illustrate equivalence class partitioning and boundary value analysis using suitable examples. **(16)**

(OR)

b) Explain the significance of control flow graph and cyclomatic complexity in white box testing with a pseudo code for sum of '*n*' numbers. **(16)**

13. a) Differentiate alpha testing from beta testing and discuss in detail about the phases in which alpha and beta testing is done. **(16)**

(OR)

b) Explain the different integration testing strategies for procedures and functions with suitable diagrams. **(16)**

14. a) Describe the components of test plan. Give examples. **(16)**

(OR)

b) i) Discuss in detail about various skills needed for a test specialist. **(8)**

ii) Explain the steps involved in forming a testing group. **(8)**

15. a) With a neat sketch discuss the design and architecture for test automation. **(16)**

(OR)

b) Discuss various metrics and measurements in software testing. Explain various types of progress metrics. **(16)**

**Question Paper Code : 72043**

03/06/2017 FN

B.E./B.Tech. DEGREE EXAMINATION, APRIL/MAY 2017.

Sixth/Seventh Semester

Information Technology

IT 6004 — SOFTWARE TESTING

(Common to B.E. Computer Science and Engineering)

(Regulations 2013)

Time : Three hours Maximum : 100 marks

Answer ALL questions.

PART A — (10 × 2 = 20 marks)

1. Mention the role of test engineer in software development organization.

2. What are the sources of defects?

3. Mention the ways by which test cases may be generated. Generate a test case for a scenario.

4. Error Vs Defect Vs Failure. Discuss.

5. Differentiate Black box with white box testing.

6. Why is it important to design test harness for testing?

7. State the limitations of statement coverage.

8. Differentiate decision and condition coverage.

9. Mention the challenges in automation.

10. Mention the types of testing amenable to automation.

PART B — (5 × 16 = 80 marks)

11. (a) State and explain all Software Testing principles. (16)

Or

(b) What are the typical origins of defects? Explain the major classes of defects in the software artefacts. (16)

12. (a) Illustrate with an example the following black box testing techniques:

(i) Equivalence Class Portioning. (8)

(ii) Boundary Value Analysis. (8)

Or

(b) Suppose you are testing defect coin problem artefacts, Identify the causes of various defects. What steps could have been taken to prevent the various classes of defects. (16)

13. (a) Explain the significance of Control flow graph and Cyclomatic complexity in white box testing with a pseudo code for sum of positive numbers. Also mention the independent paths with test cases. (16)

Or

(b) With examples explain the following black box techniques to testing

(i) Requirements based testing (4)

(ii) Positive and Negative testing (4)

(iii) State based testing (4)

(iv) User documentation and compatibility. (4)

14. (a) (i) How data flow testing aid in identifying defects in variable declaration and its use. (8)

(ii) Explain mutation testing with an example (8)

Or

(b) Explain Weyuker's eleven axioms that allow testers to evaluate test adequacy criteria. (16)

15. (a) Explain the various generations of automation and the required skills for each. (16)

Or

(b) Explain the different types of Test defect metrics under Progress metrics based on what they measure and what area they focus on.

## Question Paper Code : 80592

B.E./B.Tech. DEGREE EXAMINATION, NOVEMBER/DECEMBER 2016.

Sixth Semester

Computer Science and Engineering

IT 6004 — SOFTWARE TESTING

(Common to Seventh Semester Information Technology)

(Regulations 2013)

Time : Three hours Maximum : 100 marks

Answer ALL questions.

PART A — (10 × 2 = 20 marks)

1. Mention the objectives of Software Testing?
2. Define defects with an example. ✓
3. Sketch the control flow graph for an ATM withdrawal system.
4. Give a note on the procedure to compute cyclomatic complexity. ✓
5. List out types of system testing. ✓
6. Compare and contrast Alpha Testing and Beta Testing. ✓
7. Discuss on the role of manager in a test group.
8. What are the issues in testing object orient systems? ✓
9. Mention the criteria for selecting test tool. ✓
10. Distinguish between milestone and deliverable. ✓

PART B — (5 × 16 = 80 marks)

11. (a) Elaborate on the principles of software testing and summarize the tester role in software development organization. (16)

Or

(b) Explain in detail processing and monitoring of the defects with defect repository. (16)

12. (a) Demonstrate the various black box test cases using Equivalence class partitioning and boundary values analysis to test a module for Payroll system. (16)

Or

(b) (i) Explain the various white box techniques with suitable test cases. (8)

(ii) Discuss in detail about code coverage testing. (8)

13. (a) Explain the different integration testing strategies for procedures & functions with suitable diagrams. (16)

Or

(b) How would you identify the hardware and software for configuration testing and explain what testing techniques applied for website testing? (16)

14. (a) (i) What are the skills needed for a test specialist? 4·60 (8)

(ii) Explain the organizational structure for testing teams in single product companies. – 4·12 (8)

Or

(b) (i) Explain the components of test plan in detail. – 4·32 (8)

(ii) Compare and contrast the role of debugging goals and policies in testing. (8)

15. (a) Explain the design and architecture for automation and outline the challenges. (16)

Or

(b) What are metrics and measurements? Illustrate the types of product metrics. (16)

**Question Paper Code : 57486**

B.E./B.Tech. DEGREE EXAMINATION, MAY/JUNE 2016

Sixth Semester

Computer Science and Engineering

IT 6004 – SOFTWARE TESTING

(Regulations 2013)

Time : Three Hours Maximum : 100 Marks

Answer ALL questions.

PART – A (10 × 2 = 20 Marks)

1. Define the objective of software testing.

2. Differentiate Error, Defect and Failure. 1·35

3. What are the basic primes for all structured program ?

4. What are the errors uncovered by black box testing ?

5. Why is it important to design test harness for unit testing ?

6. What are the issues in testing object oriented systems ?

7. Make distinctions between structures of Single Product and Multi Product companies.

8. Mention the reasons to create a WBS.

9. State any two generic requirements for test tool and framework.

10. What are the skills needed in automation ?

## PART - B (5 × 16 = 80 Marks)

11. (a) (i) State and explain in detail the various software testing principles. (8)

(ii) Explain the developer and tester support for the development of a defect repository. (8)

OR

(b) (i) Define defect and illustrate the various origin of defects. (8)

(ii) What approach would you use to solve the concepts of defects with the coin problem ? (8)

12. (a) (i) Explain the significance of control flow graph and cyclomatic complexity in white box testing with a pseudo code for sum of positive numbers. Also mention the independent paths with test cases. (8)

(ii) Briefly explain the Weyuker's eleven axioms that allow testers to evaluate test adequacy criteria. (8)

OR

(b) (i) Demonstrate the various black box test cases using Equivalence class partitioning and boundary value analysis to test a module for an ATM. (8)

(ii) Explain how black box testing is performed in COTS components. (8)

13. (a) (i) Define a unit. Explain why test planning is so important for developing a repeatable and managed testing process ? (8)

(ii) Tabulate the key differences in integrating procedural oriented systems as compared to object oriented systems. (8)

OR

(b) Explain the different integration testing strategies for procedures and functions with suitable diagrams. (16)

14. (a) Explain the various impacts of globalisation and geographically distributed teams on product testing. 4.18 (16)

OR

(b) Explain the different challenges and issues faced in the testing service organization. Discuss how those challenges can be addressed. (16)

15. (a) Explain the various types of Progress and Productivity metrics based on what they measure and what area they focus on. (16)

OR

(b) Explain the design and architecture for software test automation. (16)

**Question Paper Code : 51745**

B.E./B.Tech. DEGREE EXAMINATION, MAY/JUNE 2016

Seventh Semester

Computer Science and Engineering

IT 2032/IT 702/10177 ITE 24/10144 CSE 15 – SOFTWARE TESTING

(Common to Information Technology)

(Regulations 2008/2010)

(Common to PTIT 2032/10144 CSE 15 – Software Testing for B.E. (Part-Time) Sixth Semester Computer Science and Engineering – Regulations 2009/2010)

Time : Three Hours Maximum : 100 Marks

Answer ALL questions.

PART – A (10 × 2 = 20 Marks)

1. Define software quality.

2. What is a test case ? Give example.

3. State the difference between white-box testing and black-box testing.

4. What is boundary value analysis ? Give example.

5. Define regression testing.

6. What is alpha testing ?

7. List the organization structures for testing teams.

8. What are the skills needed by a test specialist ?

9. State the advantages of using automated tools for software testing.

10. What is a metric ? Give examples for software metrics.

**PART – B (5 x 16 = 80 Marks)**

11. (a) "Principles play an important role in all engineering disciplines and are usually introduced as part of an educational background in each branch of engineering". List and discuss the software testing principles related to execution-based testing. **(16)**

**OR**

(b) What is a defect ? List the origins of defects and discuss the developer / tester support for developing a defect repository. **(16)**

12. (a) Consider the following set of requirements for the triangle problem :

R1 : If x < y + z or y < x + z or z < x + y then it is a triangle

R2 : If x ≠ y and x ≠ z and y ≠ z then it is a scalene triangle

R3 : If x = y or x = z or y = z then it is an isosceles triangle

R4 : If x = y and y = z and z = x then it is an equilateral triangle

R5 : If x > y + z or y > x + z or z > x + y then it is impossible to construct a triangle. Now, consider the following causes and effects for the triangle problem :

Causes (inputs) :

- C1 : Side "x" is less than sum of "y" and "z"
- C2 : Side "y" is less than sum of "x" and "z"
- C3 : Side "z" is less then sum of "x" and "y"
- C4 : Side "x" is equal to side "y"
- C5 : Side "x" is equal to side "z"
- C6 : Side "y" is equal to side "z"

Effects:

- E1 : Not a triangle
- E2 : Scalene triangle
- E3 : Isosceles triangle.
- E4 : Equilateral triangle
- E5 : Impossible

What is a cause-effect graph ? Model a cause-effect graph for the above. **(16)**

(b) Consider the following fragment of code :

```
i = 0;
while (i < n - 1) do
j = i + 1;
while (j < n) do
if A[i] < A[j] then
swap (A[i], A[j]);
end do;
i = i + 1;
end do;
```

Identify bug (s) if any in the above program segment, modify the code if you have identified bug (s). Construct a control flow graph and compute Cyclomatic complexity. (16)

13. (a) What is unit testing ? Explain with an example the process of designing the unit tests, running the unit tests and recording results. (16)

OR

(b) What is integration testing ? Explain with examples the different types of integration testing. (16)

14. (a) What is a test plan? List and explain the test plan components. (16)

OR

(b) Explain the role played by the managers, developers/testers, and users/clients in testing planning and test policy development. (16)

15. (a) What is software test automation ? State the major objectives of software test automation and discuss the same. (16)

OR

(b) Discuss with diagrammatic illustration the testing maturity model. (16)

# Question Paper Code : 71745

B.E./B.Tech. DEGREE EXAMINATION, APRIL/MAY 2015.

Seventh Semester

Computer Science and Engineering

IT 2032/ IT 702/ 10177 ITE 24/ 10144 CSE 15 — SOFTWARE TESTING

(Common to Information Technology)

(Regulation 2008/2010)

(Common to PTIT 2032/ 10144 CSE 15 – Software Testing for B.E. (Part-Time) Fifth /Sixth Semester Computer Science and Engineering- Regulation 2009/2010)

Time : Three hours Maximum : 100 marks

Answer ALL questions.

PART A — (10 × 2 = 20 marks)

1. Define Test Oracle and Test Bed.

2. Mention the quality attributes of software.

3. Define Test Adequacy Criteria.

4. Draw the notations used in cause effect graph.

5. State the purpose of Defect Bash testing.

6. Write the major activities followed in internationalization testing.

7. List down the skills needed by test specialist.

8. List the internal and external dependencies for executing WBS.

9. Write the different types of reviews practiced by software industry.

10. Differentiate effort and schedule.

## PART B — (5 × 16 = 80 marks)

11. (a) Discuss different testing principles being followed in Software Testing.

Or

(b) Describe the defect classes in detail with example.

12. (a) Define White Box testing. Draw CFG for the program P. Identify distinct paths and calculate cyclomatic complexity of P. Write suitable test cases to satisfy all distinct paths.

Program P

```
begin
int num, product
bool done;
product = 1;
input (done);
while (! done) {
input (num)
if(num>0)
  product = product * num;
input (done);
}
output (product);
end.
```

Or

(b) Consider an application *App* that takes two inputs *name* and age where *name* is a nonempty string containing at most 20 alphabetic characters and *age* is an integer that must satisfy the constraint $0 \leq \text{age} \leq 80$. The App is required to display an error message if the input value provided for age is out of range. The application truncates any name that is more than 20 characters in length and generates an error message if an empty string is supplied for *name*. Construct test data for *App* using the

(i) uni-dimensional equivalence partitioning

(ii) multi-dimensional equivalence partitioning

(iii) boundary value analysis technique.

13. (a) Discuss in detail about different types of integration testing.

Or

(b) Discuss the levels of testing adopted to test OO systems.

14. (a) Discuss the roles and responsibilities of testing services organization with suitable Organization structure.

Or

(b) Discuss the different test process activities of software testing in detail.

15. (a) Elaborate different types of S/W metrics and measurement used. ✓

Or

(b) Explain the design and architecture for test automation with examples.

**Question Paper Code : 91553**

B.E./B.Tech. DEGREE EXAMINATION, NOVEMBER/DECEMBER 2014.

Seventh Semester

Computer Science and Engineering

IT 2032/IT 702/10177 ITE 24/10144 CSE 15 – SOFTWARE TESTING

(Common to Information Technology)

(Regulation 2008/2010)

(Common to PTIT 2032/10144 CSE 15 – Software Testing for B.E. (Part-Time) Fifth/Sixth Semester Computer Science and Engineering – Regulation 2009/2010)

Time : Three hours                                                Maximum : 100 marks

Answer ALL questions.

PART A — (10 × 2 = 20 marks)

1. Define defects with an example.

2. Mention any two role of process in software quality.

3. How static testing is differing from structural testing?

4. What do you mean by code complexity testing?

5. Why levels of testing are preferred?

6. List out the various types of system testing.

7. Write the components of test plan.

8. Write any four skills are needed for test specialist.

9. Mention the various types of review.

10. What is the Scope of automation?

PART B — (5 × 16 = 80 marks)

11. (a) (i) Write detail note on principles of software testing. (8)

(ii) Explain the role of tester in a software development organization. (8)

Or

(b) Explain in detail how developer/tester support to develop a defect repository. (16)

12. (a) Write note on the following with an example.

(i) Smarter Tester

(ii) Random testing

(iii) Control graph

(iv) Boundary Value Analysis. (16)

Or

(b) (i) Explain the test case design strategies in detail. (8)

(ii) Compare and contrast between static testing and structural testing. (8)

13. (a) (i) Briefly explain the levels of testing with an example. (10)

(ii) Explain how to test OO system in detail. (6)

Or

(b) Explain types of testing in detail with suitable example. (16)

14. (a) (i) Explain issues caused by people and organization in software testing. – 4.1 (8)

(ii) Briefly discuss the testing team in organizational structure. (8)

– 4.11

Or

(b) Briefly discuss the various groups in Test plan and policy development with their role. – 4.56 (16)

15. (a) Briefly discuss the overview of software test automation with skill needed, scope and its Architecture. (16)

Or

(b) (i) Explain the types reviews. (8)

(ii) Discuss the components of review plans. (8)