BSc.(Information Technology)
(Semester VI)
2018-19

Software Quality Assurance
(USIT 601 Core)
University Paper Solution

By
Janhavi Vadke

Question 1

Q1a. Define the term quality and elaborate different views on quality.

Ans: There are different perception towards quality of products.

A. Customer-based definition of quality:

Quality product must meet customer needs, expectation and satisfaction.

B. Manufacturing based definition of quality:

It means customer is not fully aware about requirements & requirements are defined by architects designed on feedback survey.

C. Product-based definition of quality:

Production must add some new appreciable feature on comparison to similar product in the market.

D. Value-based definition of quality:

A product is the best combination of price and features required by the customers the cost of the product has direct relationship with the value that customer find in it

E. Transcendent quality:

It means that customer purchased the product because of specific feature absent/present in the product.

Quality views

1) producer's view -Meeting requirements is a producer's view of quality. This is the view of the organization responsible for the project and processes, and the products and services acquired, developed, and maintained by those processes. Meeting requirements means that the person building the product does so in accordance with the requirements. Requirements can be very complete or they can be simple, but they must be defined in a measurable format, so it can be determined whether they have been met. The producer's view of quality has these four characteristics:

- Doing the right thing
- Doing it the right way
- Doing it right the first time
- Doing it on time without exceeding cost

2) customer's view-Being fit for use is the customer's definition. The customer is the end user of the products or services. Fit for use means that the product or service meets the customer's needs regardless of the product requirements. Of the two definitions of quality, fit for use, is the more important. The customer's view of quality has these characteristics:

- Receiving the right product for their use
- Being satisfied that their needs have been met
- Meeting their expectations
- Being treated with integrity, courtesy and respect

Most Information Technology (IT) groups have two quality gaps: the producer gap and the customer gap. The producer gap is the difference between what is specified (the documented requirements and internal standards) versus what is delivered (what is actually built). The customer gap is the difference between what the producers actually delivered versus what the customer wanted.

3) Provider view – This is the perspective of the organization that delivers the products and services to the customer.

4) Supplier view – This is the perspective of the organization (that may be external to the producer's company, such as an independent vendor) that provides either the producer and/or the provider with products and services needed to meet the requirements of the customer.

Q1b. Explain the lifecycle of quality improvements.

Ans: *For improvement of quality, a cycle must be followed:*

Dr. Joseph Juran is a pioneer of statistical quality control with a definition of improvement cycle through (DMMCI) i.e. 'Define', 'Measure', 'Monitor', 'Control', 'Improve'. There is interrelationships among the customers, suppliers and processes used in development, testing etc. and people should establish quality management based on metrics program.

Following are three parts of the approach, namely :

(1) Quality planning at all levels : Quality planning happens at two levels :

Organization Level : It must be in the form of policy definition and strategic quality plans on the basis of vision, missions set by senior management.

Unit Level : Quality planning at unit level must be done by the people responsible for managing the unit. Project plan and quality plan at unit level must be consistent with the strategic quality plans at organization level.

(2) Quality control :

Quality control process examines the present product at various levels with the defined standards so that an organization may appraise the outcome of the processes.

It must measure the deviations with respect to the number of achievements planned in quality planning to reduce those deviations to minimum.

(3) Quality improvement :

Improvement processes attempts to continuously improve the quality of the process used for producing products.

There is no end to quality improvements and it needs or take newer challenges again and again.

Q 1c. What are the quality principles of Total Quality Management (TQM)?

Ans:

TQM principles

1. Develop constancy of purpose of definition and deployment of various initiatives

Management must create constancy of purpose for products and processes, allocating resources adequately to provide for long term as well as short term needs. The processes followed during entire lifecycle of product development from requirement gathering to final delivery must be consistent with each other over a larger horizon.

2. Adapting to new philosophy of managing people / stakeholders by building confidence and relationships

Management must adapt to the new philosophies of doing work and getting work done from its people and suppliers.

3. Declare freedom from mass inspection of incoming/produced output

There must be an approach for elimination of mass inspection followed by cost of failure as the way to achieve quality of products because mass inspection results into huge cost overrun and product produced is of inferior quality.

4. Stop awarding of lowest price tag contracts to suppliers

Organizations must end the practice of comparing unit purchase price as criteria for awarding contracts. Vendor selection must be done on the basis of total cost including price, rejections, etc .Aim of vendor selection is to minimize total cost, not merely initial cost of purchasing.

5. Improve every process used for development and testing of product.

Improve every process of planning, production and service to the customer and other support processes constantly.

6. Institutionalize training across the organization for all the people

An organization must include modern methods of training which may include on-the-job-training, classroom training, self-study etc. for all people to make better use of their capabilities. Skill levels of people can be enhanced to make them suitable for better performance by planning different training programs.

7. Institutionalize leadership throughout organization at each level

An organization must adopt and include leadership at all levels with the aim of helping people to do their jobs in a better way. Their focus must shift from number of work items to be produced to quality of output.

8. Drive out fear of failure from employees

An organization must encourage effective two-way communication and other means to drive out fear of failure from minds of all employees. Employees can work effectively and more productively to achieve better quality output when there is no fear of failure

9. Break down barriers between functions/departments

Physical as well as psychological barriers between departments and staff areas must be broken down to create a force of cohesion. People start performing as a team and there in synergy of group activities

10. Eliminate exhortations by numbers, goals, targets

Eliminate use of slogans , posters and exhortations of the work force, demanding 'Zero Defects' and new levels of productivity ,without providing methods and guidance about how to achieve it

11. Eliminate arbitrary numerical targets which are not supported by processes

Eliminate quotas for work force and numerical goals for managers to be achieved. Substitute the quotas with mentoring and support to people, and helpful leadership in order to achieve continual improvement in quality and productivity of processes.

12. Permit pride of workmanship for employees

People must feel proud of the work that they are doing, and know how they are contributing to organizational level. This means replacing 'management by objective' to 'management by fact'

13. Encourage education of new skills and techniques

Arrange a rigorous program of education and training for people working in different areas and encourage self-improvement programs for everyone. They need to accept new challenges

14. Top management commitment and action to improve continually

Clearly define top management's commitment to ever-improving quality and productivity and their obligation to implement quality principles throughout the organization.

Q1d Explain the structure of quality management system.

Ans:

Every organization has a different quality management structure depending upon its need and circumstances. General view of quality management is defined below in a diagram :

Following are the three main tiers of quality management system structure :

1st TIER – Quality Policy

・・Quality policy sets the wish, intent and direction by the management about how activities will be conducted by the organization.

・・Since management is the strongest driving force in an organization, its intents are most important

・・It is a basic framework on which the quality temple rests.

2nd TIER – Quality Objectives

・・Quality objectives are the measurements established by the management to define progress and achievements in a numerical way.

・・An improvement in quality must be demonstrated by improvement in achievements of quality factors (test factors) in numerical terms as expected by the management.

・・The achievements of these objectives must be compared with planned levels expected and results and deviations must be acted upon.

3rd TIER – Quality Manual

・・Quality manual also termed as policy manual is establishes and publishes by the management of

the organization.

・・It sets a framework for other process definitions and is a foundation of quality planning at organization level.

Q1e. How the quality and productivity are related with each other?

Ans:

Productivity is the relationship between a given amount of output and the amount of input needed to produce it. Profitability results when money is left over from sales after costs are paid. The expenditures made to ensure that the product or service meets quality specifications affect the final or overall cost of the products and/or services involved. Efficiency of costs will be an important consideration in all stages of the market system from manufacturing to consumption. Quality affects productivity. Both affect profitability. The drive for any one of the three must not interfere with the drive for the others. Efforts at improvement need to be coordinated and integrated. The real cost of quality is the cost of avoiding nonconformance and failure. Another cost is the cost of not having quality—of losing customers and wasting resources.

1)Improvement in quality directly leads to improved productivity. Quality experts such as W. Edwards Deming have stated that quality is positively associated with productivity because as the quality of a product or service increases, there is less need for correcting work or fixing mistakes, so productivity improves.
2)The hidden factory producing scrap, rework, sorting, repair, customer complaint is closed.
3)Effective way to improve productivity is to reduce scrap and rework by improving processes. By improving and Implementing sound management practices, use research and development effectively, adopt modern manufacturing techniques, and improve time management.
4) Quality improvements lead to cost reduction

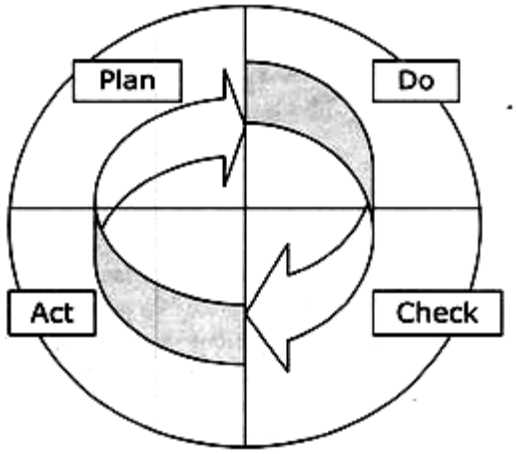
Q1f. Write a short note on continual improvement cycle.

Ans:

i) Continual (Continuous) improvement cycle is based on systematic sequence of Plan-Do-Check-Act activities representing a never ending cycle of improvements. It was initially implemented in agriculture them later in electronic industry and now it is famous in all industries.

(ii) PDCA improvement cycle can be thought of as a wheel of improvement continually(continuously) rolling up the problem-solving and achieving better and better results for organization at each iteration.

(iii) Following are the stages of PDCA cycle explained with the help of a diagram



(a) Plan:

- *Planning includes answering all questions like who, when, what, why, where, how etc. about various activities and setting expectations.*
- *Expected results must be defined in quantitative terms an actions must be planned to achieve answers to these questions.*
- *Baseline studies (i.e. where one is standing and vision defines where one wishes to be) are important for planning.*

(b) Do:

- *An organization must work in the direction set by the plan devised in earlier phase for improvement.*
- *Actual execution of a plan can determine whether the results as expected are achieved or not.*
- *'Do' process need inputs like resources, hardware, software, training etc. for execution of a plan.*

(c) Check:

- *An organization must compare the actual outcome of 'Do' stage with reference or expected results which are planned outcomes.*
- *Expected and actual results must be in numerical terms, and compared at some periodicity as defined in the plan.*

(d) Act:

- *If any deviations are observed in actual outcome with respect to planned results, the organization may need to decide actions to correct the situation.*
- *One may have to initiate corrective or preventive actions as per the outcome of 'Check' stage.*
- *When expected results and actual results match with given degree of variation, one can consider that the plan is going in the right direction.*

Question 2

Q2a. Explain the lifecycle of software testing.

Ans:

Software Testing Life Cycle (STLC) is defined as a sequence of activities conducted to perform Software Testing. Software Testing is not a just a single activity. It consists of a series of activities carried out methodologically to help certify your software product.

Test planning & control:

- *Determine the scope and risks & identify the objectives of testing.*
- *Determine the test approach.*
- *Implement the test policy test strategy.*
- *Determine the requirement test resources (people, test environment pc's)*
- *Determine the exit criteria i.e. task & checks should be completed.*

Test analysis & design:

- *Designing a black-box test before code exists.*
- *Requirement should help in designing test like s/w needs to responds in 5 seconds with 20 people logged on.*
- *Assigning priorities to test identifying test data used to execute test cases.*

Test implementation & execution:

- *In this phase test cases are executed either manually or automation.*
- *It compares expected results & actual results.*
- *It logs the result of test execution.*

Evaluating exit criteria & reporting:

- *Exit criteria like coverage criteria, acceptance, and criteria are evaluated.*
- *Test summary report is generated & it is provided to stake holders.*

Test closure activities:

- *Test closure activities are adding reuse component in repository.*
- *Check whether all bugs have been resolved or not.*
- *Document the result which can be reused in future releases.*

Q2b Write a note on requirement traceability matrix

Ans:

Requirement Traceability matrix provides complete mapping for the software . It is a Blueprint of an entire application. It helps in tracing if any requirement is not implemented. If any redundancy, the entire software development can be tracked. It can track failure of any test case. Application can become maintainable.

Disadvantages:

1 Difficult to create manually, Invest money, training
2 Maintaining relationship need huge effort
3 Requirements changes frequently
4 Developing team may not understand the importance
5 Customer may not find value in it

Horizontal Traceability:

1 When an application can be traced from requirement up to test results
2 On failure of test case, we must be able to find which requirement have not been met

3 When any requirement is not traceable to design, that requirement is not implemented at all.

Bidirectional traceability
1 It must be able to go in any direction from any point in traceability matrix

Vertical Traceability
1 Exist in individual column
2 Interdependencies, parent-child

Risk Traceability
1 It References about the risks or failure
2 It provides Control mechanism to reduce probability or improve detection ability.

Q2c. State and explain any 5 principles of software testing.

Ans:

The basic principles on which testing is based are given below:

1) Define the expected output or result for each test case executed, to understand if expected and actual output matches or not. Mismatches may indicate possible defects. Defects may be in
product or test cases or test plan.

2) Developers must not test their own programs. No defects would be found in such kind of testing as approach-related defects will we difficult to find.

3) Inspect the results of each test completely and carefully. It would help in root cause analysis and can be used to find weak processes. This will help in building processes rightly and improving their capability.

4) Include test cases for invalid or unexpected conditions which are feasible during production. Testers need to protect the users from any unreasonable failure.

5) Test the program to see if it does what is no supposed to do as well as what it is supposed to do.

6) Reusability of test case is important for regression. Test cases must be used repetitively so that they remain applicable. Test data may be changed in different iterations.

7) Do not plan tests assuming no errors will be found. There must be targeted number of defects for testing.

8) The probability of locating more errors in any module is directly proportional to the number of errors already found in that module.

Q2d. Explain the relationship between error, defect and failure with a proper example.

Ans:

ERROR: An error is a mistake, misconception, or misunderstanding on the part of a software developer. In the category of developer we include software engineers, programmers, analysts, and testers.

For example, a developer may misunderstand a de-sign notation, or a programmer might type a variable name incorrectly – leads to an Error. It is the one which is generated because of wrong login, loop or due to syntax. Error normally arises in software; it leads to change the functionality of the program.

DEFECT: It can be simply defined as a variance between expected and actual. Defect is an error found AFTER the application goes into production. It commonly refers to several troubles with the software products, with its external behavior or with its internal features.

In other words Defect is the difference between expected and actual result in the context of testing. It is the deviation of the customer requirement.

FAILURE: A failure is the inability of a software system or component to perform its required functions within specified performance requirements. When a defect reaches the end customer it is called a Failure. During development Failures are usually observed by testers. When a system or piece of software produces an incorrect result or does not perform the correct action, this is known as a failure. Failures are caused by faults in the software

Q2e. Discuss the challenges in software testing.

Ans:

Challenges in testing

- Requirements are not clear, complete, consistent, measurable and testable. These may create some problem in defining test cases.
- Requirements are wrongly documented and interpreted by business analyst and system analyst. These knowledgeable people are supposed to gather requirements of customers by understanding their business workflow. But sometimes they are prejudiced based on their experience.
- Code logic may be difficult to capture. Often testers are not able to understand the code due to lack of technical language knowledge. Sometimes they do not have access to files.
- Error handling may be difficult to capture. There may be combinations of errors and various error messages and controls are required such as detective controls , corrective controls, suggestive controls and preventive controls.

Other challenges in testing

- Badly written code introduces many defects
- Bad architecture of software cannot implement good requirement statement
- Testing is considered as a negative activity
- Testers find themselves in Lose - Lose situation

Q2f. Describe the structure of a testing team.

Ans: Location of test teams in organization Independent test team

Advantages of independent test team

- Test team is not under a delivery pressure
- Test team is not under pressure of `Not finding' a defect
- Independent view about a product is obtained as thought process of developers and testers may be completely different
- Expert guidance and mentoring required by test team for doing effective testing may be available in form of test manager

Disadvantages of independent test team

- There is always `Us' vs `Them' mentality
- Testers may not get a good understanding of development process
- Sometimes management is inclined excessively towards development team or test team and other team feels that they have no value in an organization

Test team reporting to development manager

Advantages of test team reporting to development manager

- There is a better cooperation between development team and test team as both are part of same team

- Test team can be involved in development and verification / validation activities from the start of the project
- Testers may get a good understanding of development process and can help in process improvement

Disadvantage of test team reporting to development manager

- Expert advice in form of test manager may not be available to testers
- Sometimes development managers are more inclined towards development team
- Many times testers start perceiving product from developer's angle and their defect finding ability is reduced

Matrix organization

Developers becoming testers Advantages of this approach

- Developers do not need another knowledge transfer while working as a tester
- Developers have better understanding of detail design, coding etc and can test application it easily
- For automation, some amount of development skill is required in writing the automation scripts
- It is less costly as there is no separate test team
- Psychological acceptance of defects is not a major issue as developers themselves find the defects

Disadvantages of this approach

- Developers may not find value in doing testing
- There may be blindfolds while understanding requirements or selection of approach
- Developers may concentrate more on development activities
- Development needs more of a creation skill while testing needs more of a destruction skill

Independent testing team Advantages of this approach

- Separate test team is supposed to concentrate more on test planning, test strategies and approach, creating test artifacts etc
- There is independent view about the work products derived from requirement statement
- Special skills required for doing special tests may be available in such independent teams
- `Testers working for customer', can be seen in such environment

Disadvantages of this approach

- Separate team means additional cost for organization
- Test team needs ramping up and knowledge transfer similar to development team
- Organization may have to check for rivalries between development team and test team

Domain experts doing a software testing Advantages of this approach

- `Fitness for use' can be tested
- Domain experts may provide facilitation

- Domain experts understand the scenario faced by actual users

Disadvantages of this approach

- Domain experts may have prejudices about the domain
- It may be very difficult to get domain experts in all areas
- It may mean huge cost for the organization

Question 3

Q3a. Explain boundary value testing and its guidelines.

Ans:

Boundary Value Analysis focuses on the boundary of the input space to identify test cases.

The rationale behind value testing is that errors tend to occur near the extreme values of an input variable.

Ex: Loop conditions may test for < when they should test for .

BVA uses input variable values at their:

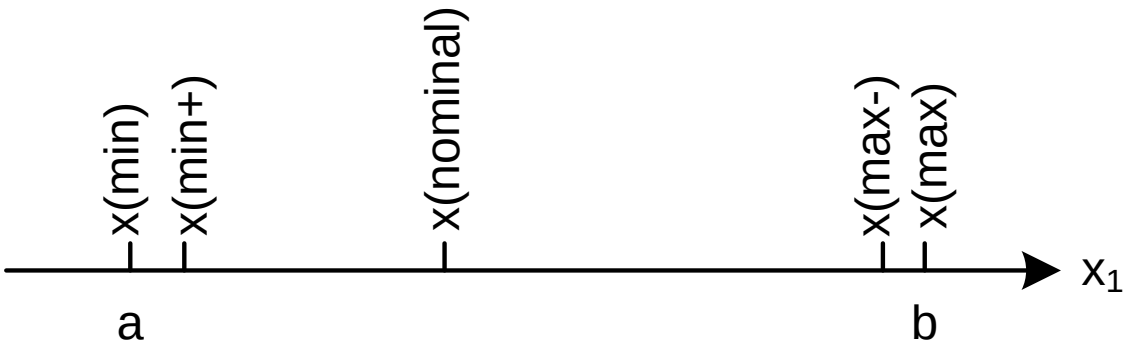
1)Minimum (min)

2) Just above the minimum (min+)

3) A nominal value (nom)

4) Just below their maximum (max-)

5) Maximum (max)

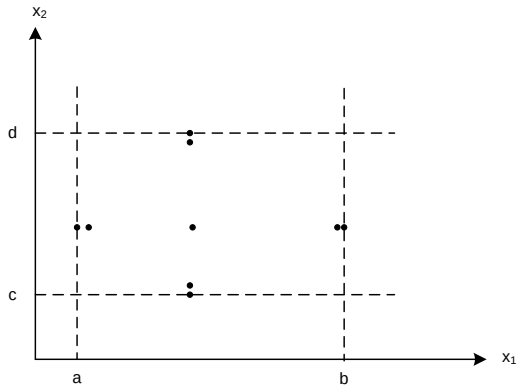


- *Test values for variable x where, $a \le x \le b$ and*

Four types of Boundary value testing

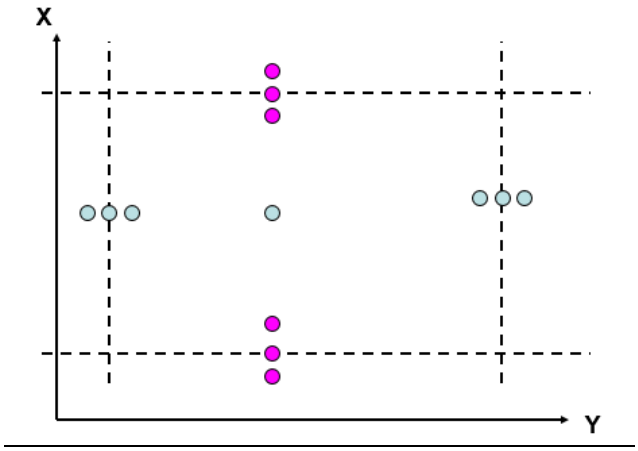
1) *Normal Boundary Value Testing*
2) *Robust Boundary Value Testing*
3) *Worst case Boundary Value Testing*
4) *Robust worst case Boundary Value Testing*

1) Normal Boundary Value Testing



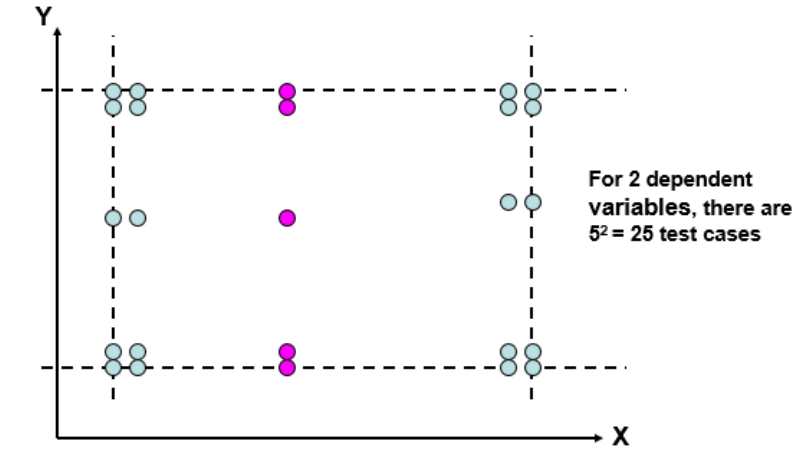
there will be (4n + 1) test cases for n <u>independent inputs</u>

2) Robust Boundary Value Testing



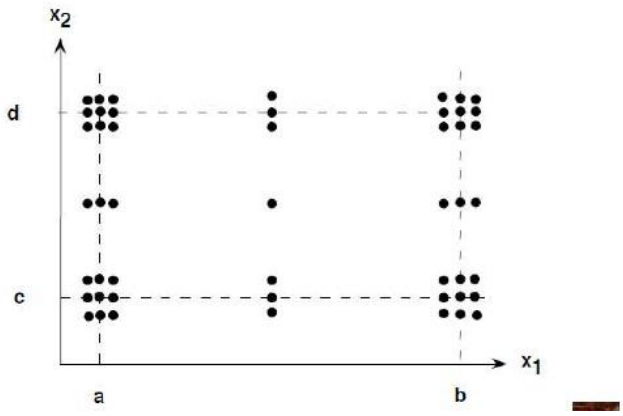
In general, there will be (6n+ 1) test cases for n independent inputs

3) Worst case Boundary Value Testing



- In general, there will be 5^n test cases for n <u>dependent inputs</u>.

4) Robust worst case Boundary Value Testing



Robust Worst-case testing for a function on n variables generates 7n test cases.

Q3b. Write a note on improved equivalence class testing.

Ans:

Equivalence class testing

1) The inputs and outputs are partitioned into mutually exclusive parts called as Equivalence classes.

2) The elements in the equivalence classes are such that, they are expected to exhibit similar behaviour.

3) Equivalence partitions/classes can be found for both valid data and invalid data.

4) Any one sample from a class, represents the entire class.

Steps to write test cases:

1. Identify equivalence classes by taking each input & output conditions and partitioning it into valid and invalid classes.

2. Generate test cases using equivalence classes.

Ex: If Age lies between 18 to 60, then equivalence classes will be

Invalid class	Valid class	Invalid class
<17	$18 \leq age \leq 60$	> 60

There are 4 types of equivalence class testing

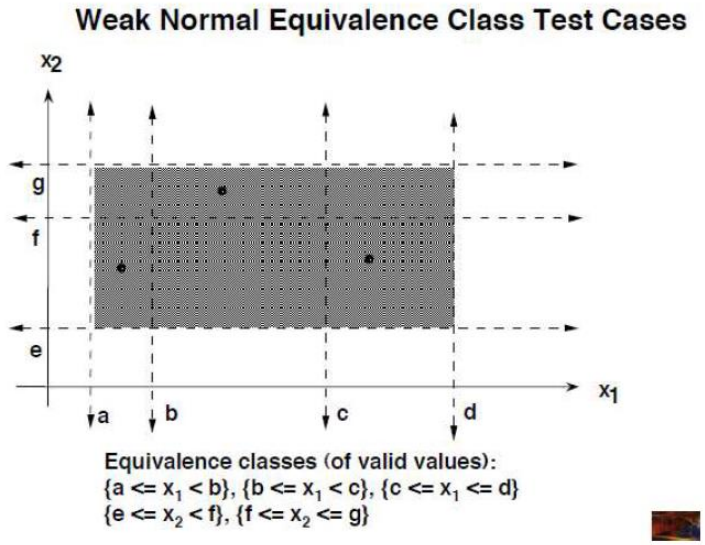
1) Weak Normal equivalence class testing

2) Strong Normal equivalence class testing

3) Weak Robust equivalence class testing

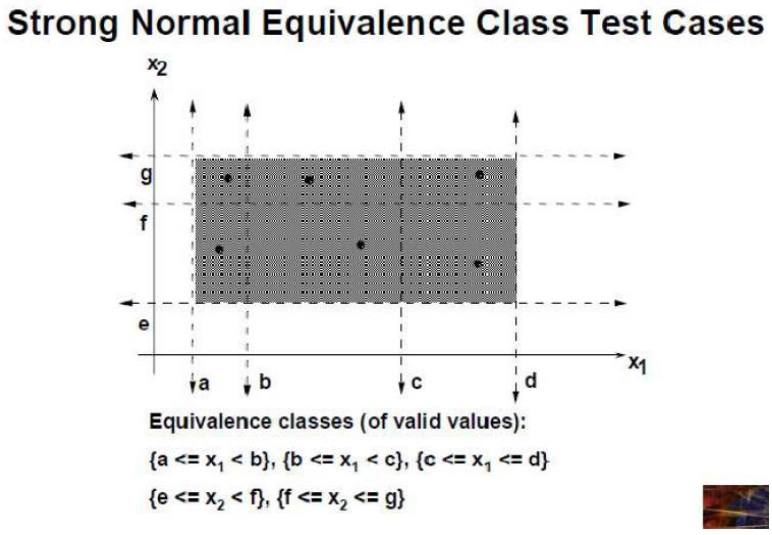
4) Strong Robust equivalence class testing

1) Weak normal equivalence class testing is accomplished by using one variable from each equivalence class in a test case. The word 'weak' means 'single fault assumption'. This type of testing is accomplished by using one variable from each equivalence class in a test case. We would, thus, end up with the weak equivalence class test cases as shown in the figure. Each dot in above graph indicates a test data. From each class we have one dot meaning that there is one representative element of each test case. In fact, we will have, always, the same number of weak equivalence class test cases as the classes in the partition.

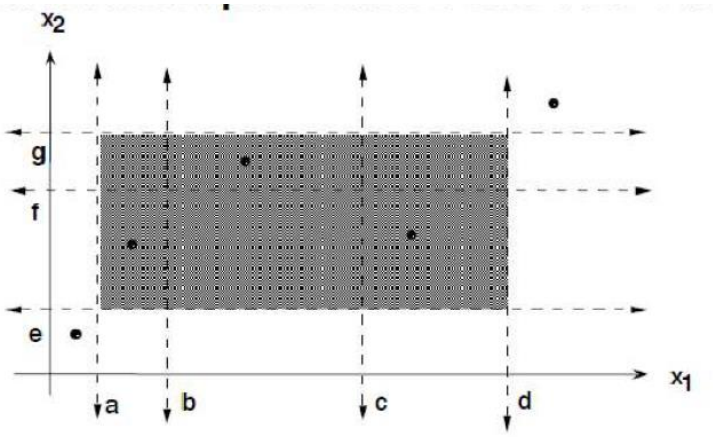


2) Strong Normal Equivalence Class Testing :
This type of testing is based on the multiple fault assumption theory. So, now we need test cases from each element of the Cartesian product of the equivalence classes, as shown in the figure. Just like we have truth tables in digital logic, we have similarities between these truth tables and our pattern of test cases. The Cartesian product guarantees that we have a notion of "completeness" in following two ways :

- *We cover all equivalence classes*
- *We have one of each possible combination of inputs.*

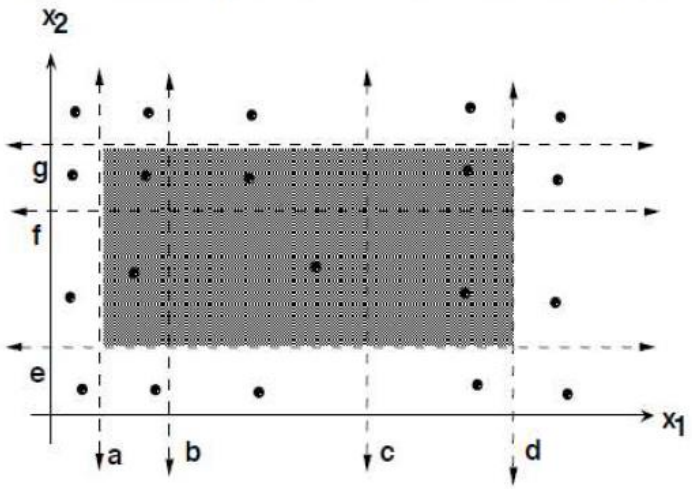


3) Weak Robust equivalence class testing
Identify equivalence classes of valid and invalid values.
Test cases have all valid values except one invalid value.
Detects faults due to calculations with valid values of a single variables.
Detects faults due to invalid values of a single variable.



Equivalence classes (of valid and invalid values):
{a <= x_1 < b}, {b <= x_1 < c}, {c <= x_1 <= d}, {e <= x_2 < f}, {f <= x_2 <= g}
Invalid classes: {x_1 < a}, {x_1 > d}, {x_2 < e}, {x_2 > g}

4) Strong Robust equivalence class testing
Robust part comes from consideration of invalid values. Strong part refers to the multiple fault assumption.



Equivalence classes (of valid and invalid values):
{a <= x_1 < b}, {b <= x_1 < c}, {c <= x_1 <= d}, {e <= x_2 < f}, {f <= x_2 <= g}
Invalid classes: {x_1 < a}, {x_1 > d}, {x_2 < e}, {x_2 > g}

Q3c. Describe the decision table testing technique in detail.
Ans: Decision table ideal for describing situations in which a number of combinations of actions are taken under varying sets of conditions.
1. A decision table has four portions the condition stub, the condition entries, the action stub, and the action entries.
2.A column in the entry portion is a rule. Rules indicate which actions, if any, are taken for the circumstances indicated in the condition portion of the rule. In the decision table below, when conditions c1, c2, and c3 are all true, actions a1 and a2 occur. When c1 and c2 are both true and c3 is false, then actions a1 and a3 occur.
3. The entry for c3 in the rule where c1 is true and c2 is false is called a "don't care" entry. The don't care entry has two major interpretations: the condition is irrelevant, or the

condition does not apply.
4.This structure guarantees that we consider every possible combination of condition values.
5. This completeness property of a decision table guarantees a form of complete testing.
6. Decision tables in which all the conditions are binary are called Limited Entry Decision Tables (LETDs).

Table : Decision table diagram.

Stub	Rule 1	Rule 2	Rules 3, 4	Rule 5	Rule 6	Rules 7, 8
c1	T	T	T	F	F	F
c2	T	T	F	T	T	F
c3	T	F	–	T	F	–
a1	X	X		X		
a2	X				X	
a3		X		X		
a4			X			X

Q3d. Write a note on DD path testing.

Ans:

The best-known form of code-based testing is based on a construct known as a decision-to – decision path (DD-path) (Miller, 1977). The name refers to a sequence of statements that, in Miller's words, begins with the "outway" of a decision statement and ends with the "inway" of the next decision statement. No internal branches occur in such a sequence, so the corresponding code is like a row of dominoes lined up so that when the first falls, all the rest in the sequence fall.

We will define DD-paths in terms of paths of nodes in a program graph. In graph theory, these paths are called chains, where a chain is a path in which the initial and terminal nodes are distinct, and every interior node has indegree = 1 and outdegree = 1.

Definition

A DD-path is a sequence of nodes in a program graph such that

Case 1: It consists of a single node with indeg = 0.

Case 2: It consists of a single node with outdeg = 0.

Case 3: It consists of a single node with indeg ≥ 2 or outdeg ≥ 2.

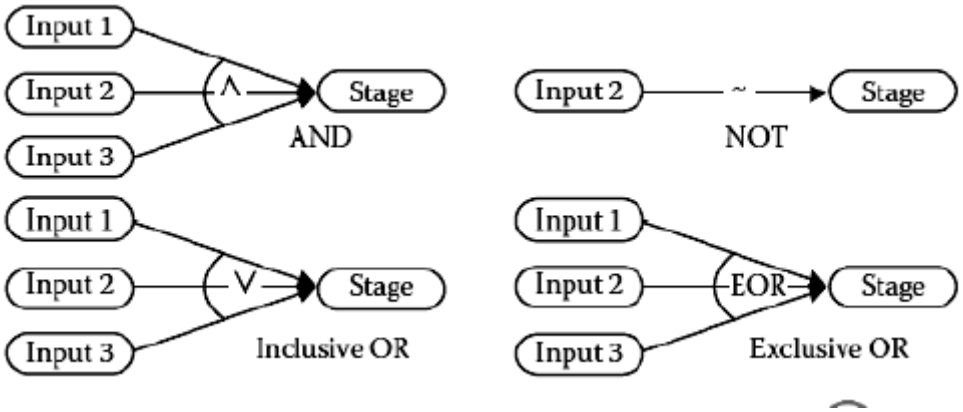
Case 4: It consists of a single node with indeg = 1 and outdeg = 1.

Case 5: It is a maximal chain of length ≥ 1.

Q3e. Explain the concept and significance of cause and effect graphing technique

Ans:

- Cause-Effect Graphing is a technique where causes are the input conditions and effects are the results of those input conditions.
- The Cause-Effect graph technique restates the requirements specification in terms of the logical relationship between the input and output conditions. Since it is logical, it is obvious to use Boolean operators like AND, OR and NOT.
- Cause-and-effect graphs shows unit inputs on the left side of a drawing, and using AND, OR, and NOT "gates" to express the flow of data across stages of unit.
- The following Figure shows the basic cause-and-effect graph structures :

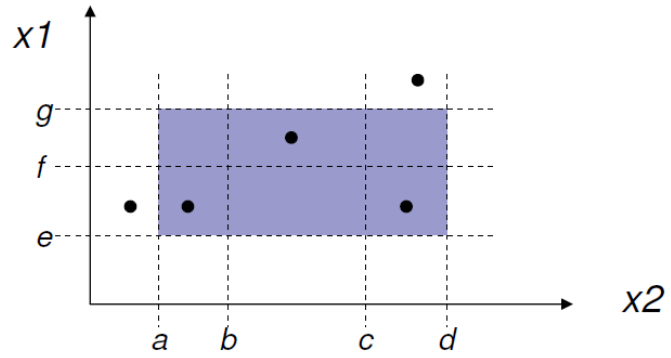


Q3f. Compare weak robust and strong robust equivalence class testing.

Ans: There are two main properties that underpin the methods used in functional testing. The single fault assumption and the multiple fault assumption. These two properties lead to two different types of equivalence class testing, weak and strong. However if we decide to test for invalid input/output as well as valid input/output we can produce another two different types of Equivalence Class Testing, normal and robust. Robust Equivalence Class testing takes into consideration the testing of invalid values, whereas normal does not. Therefore we now have four different types of Equivalence Class Testing, namely weak normal, strong normal, weak robust and strong robust.

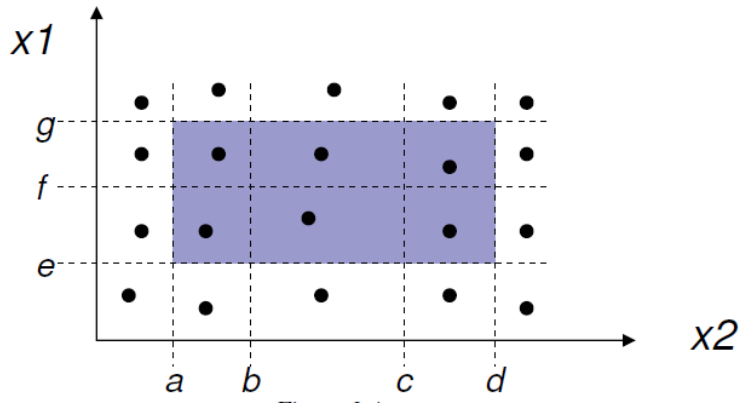
1) Weak Robust Equivalence Class Testing

As with weak normal Equivalence Class testing we only test for one variable from each Equivalence Class. However we now also test for invalid values as well. Since weak Equivalence Class Testing is based on the single fault assumption a test case will have one invalid value and the remaining values will all be valid.



2) Strong Robust Equivalence Class Testing

This form of Equivalence Class testing produces test cases for all valid and invalid elements of the Cartesian product of all the equivalence classes.


x1
g
f
e
x2
a
b
c
d

Question 4

Q4a. Explain different methods of verification.

Ans: Verification is the process, to ensure that whether we are building the product right i.e., to verify the requirements which we have and to verify whether we are developing the product accordingly or not.

Methods of verification:

- **Inspection** - This is the process of examining the product using one or several of the five senses: visual, auditory, olfactory, tactile, and taste. An example of inspection is the taste test of a cake you ordered. For software development, it may mean reading the source code and checking for syntax mistakes.
- **Self-Review-**Self review may not be referred as an official way of review in most of the software verification descriptions, as it is assumed that everybody does a self-check before giving work product for further verification.
- **Peer Review**-Peer review is the most informal type of review where an author and a peer are involved. It is review done by a peer and review records are maintained. A peer may be a fellow developer or tester as the case may be.
- **Walkthrough**- A walkthrough is conducted by the author of the 'document under review' who takes the participants through the document and his or her thought processes, to achieve a common understanding and to gather feedback.
- **Audits**-Audit is a formal review based on samples. Audits are conducted by auditors who may or may not be experts in the given work product.

Q4b. Explain the steps involved in management of verification and validation.

Ans:

Verification is a process of evaluating software at the development phase. It helps you to decide whether the product of a given application satisfies the specified requirements. Validation is the process of evaluating software at the after the development process and to check whether it meets the customer requirements.

- *Defining the processes for Verification and Validation*
    - *Software quality assurance process*
    - *Software quality control process*
    - *Software development process*
    - *Software life cycle definition*
- *Prepare plans for execution of process*
- *Initiate implementation plan*
- *Monitor execution plan*
- *Analyze problems discovered during execution*
- *Report progress of the processes*
- *Ensure product satisfies requirements*

Q4c Describe the benefits of review technique.

Ans:

A review is a systematic examination of a document by one or more people with the main aim of finding and removing errors early in the software development life cycle. Reviews are used to verify documents such as requirements, system designs, code, test plans and test cases.

- Verification can confirm that the work product has followed the processes correctly

- It can find defect in terms of deviations from standards easily
- Location of the defect can be found
- It can reduce the cost of finding and fixing the defects
- It can locate the defect easily as work product under review is yet to be integrated
- It can be used effectively for training people
- Productivity is improved and timescales reduced because the correction of defects in early stages and work-products will help to ensure that those work-products are clear and unambiguous.
- Testing costs and time is reduced as there is enough time spent during the initial phase.
- Reduction in costs because fewer defects in the final software.
- Verification helps in lowering down the count of the defect in the later stages of development.
- Verifying the product at the starting phase of the development will help in understanding the product in a better way.
- It reduces the chances of failures in the software application or product.
- It helps in building the product as per the customer specifications and needs.

Q4d. List and explain how the formal review is carried out.

Ans: A typical formal review process consists of six main steps:

1 Planning
2 Kick-off
3 Preparation
4 Review meeting
5 Rework
6 Follow-up.

(i) Planning

The review process for a particular review begins with a 'request for review' by the author to the moderator (or inspection leader). A moderator is often assigned to take care of the scheduling (dates, time, place and invitation) of the review. On a project level, the project planning needs to allow time for review and rework activities, thus providing engineers with time to thoroughly participate in reviews. For more formal reviews, e.g. inspections, the moderator always performs an entry check and defines at this stage formal exit criteria. The entry check is carried out to ensure that the reviewers' time is not wasted on a document that is not ready for review. A document containing too many obvious mistakes is clearly not ready to enter a formal review process and it could even be very harmful to the review process. It would possibly de-motivate both reviewers and the author. Also, the review is most likely not effective because the numerous obvious and minor defects will conceal the major defects.

(ii) Kick-off

An optional step in a review procedure is a kick-off meeting. The goal of this meeting is to get everybody on the same wavelength regarding the document under review and to commit to the time that will be spent on checking. Also the result of the entry check and defined exit criteria are discussed in case of a more formal review. In general a kick-off is highly recommended since there is a strong positive effect of a kick-off meeting on the motivation of reviewers and thus the effectiveness of the review process.

During the kick-off meeting the reviewers receive a short introduction on the objectives of the review and the documents. The relationships between the document under review and

the other documents (sources) are explained, especially if the number of related documents is high.

Role assignments, checking rate, the pages to be checked, process changes and possible other questions are also discussed during this meeting. Of course the distribution of the document under review, source documents and other related documentation, can also be done during the kick-off.

(iii) Preparation

The participants work individually on the document under review using the related documents, procedures, rules and checklists provided. The individual participants identify defects, questions and comments, according to their understanding of the document and role. All issues are recorded, preferably using a logging form. Spelling mistakes are recorded on the document under review but not mentioned during the meeting. The annotated document will be given to the author at the end of the logging meeting. Using checklists during this phase can make reviews more effective and efficient, for example a specific checklist based on perspectives such as user, maintainer, tester or operations, or a checklist for typical coding problems.

A critical success factor for a thorough preparation is the number of pages checked per hour. This is called the checking rate.

(iv) Review Meeting

The meeting typically consists of the following elements (partly depending on the review type): logging phase, discussion phase and decision phase.

During the logging phase the issues, e.g. defects, that have been identified during the preparation are mentioned page by page, reviewer by reviewer and are logged either by the author or by a scribe. A separate person to do the logging (a scribe) is especially useful for formal review types such as an inspection. To ensure progress and efficiency, no real discussion is allowed during the logging phase. If an issue needs discussion, the item is logged and then handled in the discussion phase. A detailed discussion on whether or not an issue is a defect is not very meaningful, as it is much more efficient to simply log it and proceed to the next one. Furthermore, in spite of the opinion of the team, a discussed and discarded defect may well turn out to be a real one during rework.

During the logging phase the focus is on logging as many defects as possible within a certain timeframe. To ensure this, the moderator tries to keep a good logging rate (number of defects logged per minute). In a well-led and disciplined formal review meeting, the logging rate should be between one and two defects logged per minute.

For a more formal review, the issues classified as discussion items will be handled during this meeting phase. Informal reviews will often not have a separate logging phase and will start immediately with discussion. Participants can take part in the discussion by bringing forward their comments and reasoning. As chairman of the discussion meeting, the moderator takes care of people issues. For example, the moderator prevents discussions from getting too personal, rephrases remarks if necessary and calls for a break to cool down 'heated' discussions and/or participants.

At the end of the meeting, a decision on the document under review has to be made by the participants, sometimes based on formal exit criteria. The most important exit criterion is the average number of critical and/or major defects found per page.

(v) Rework

Based on the defects detected, the author will improve the document under review step by step. Not every defect that is found leads to rework. It is the author's responsibility to judge if a defect has to be fixed. If nothing is done about an issue for a certain reason, it should be reported to at least indicate that the author has considered the issue.

Changes that are made to the document should be easy to identify during follow-up. Therefore the author has to indicate where changes are made.

(vi) Follow-up

The moderator is responsible for ensuring that satisfactory actions have been taken on all (logged) defects, process improvement suggestions and change requests. Although the moderator checks to make sure that the author has taken action on all known defects, it is not necessary for the moderator to check all the corrections in detail. If it is decided that all participants will check the updated document, the moderator takes care of the distribution and collects the feedback. For more formal review types the moderator checks for compliance to the exit criteria.

In order to control and optimize the review process, a number of measurements are collected by the moderator at each step of the process.

Q4e. Explain the VV model of testing.

Ans: The VV-model is an SDLC model where execution of processes happens in a sequential manner in a V-shape. It is also known as Verification and Validation model. The VV-Model is an extension of the waterfall model and is based on the association of a testing phase for each corresponding development stage.

- **Requirement Analysis**
  Requirements analysis is critical to the success or failure of a systems or software project. The requirements should be documented, actionable, measurable, testable, traceable, related to identified business needs or opportunities, and defined to a level of detail sufficient for system design.
- **System Design**
  Software design usually involves problem solving and planning a software solution. This includes both a low-level component and algorithm activities and a high-level, architecture design.
- **Program Design**
  Program Design is the process that an organization uses to develop a program. It is most. often an iterative process involving research, consultation, initial design, testing and. redesign. A program design is the plan of action that results from that process.
- **Acceptance testing**
  Acceptance testing is a level of software testing where a system is tested for acceptability. The purpose of this test is to evaluate the system's compliance with the business requirements and assess whether it is acceptable for delivery.
- **Interface testing**

Interface testing is defined as a software testing type which verifies whether the communication between two different software systems is done correctly. A connection that integrates two components is called interface.

- **Integration testing**

Integration testing is a level of software testing where individual units are combined and tested as a group? The purpose of this level of testing is to expose faults in the interaction between integrated units. Test drivers and test stubs are used to assist in Integration Testing

- **Coding, code review, unit testing**

Coding methodology includes a diagrammatic notation for documenting the results of the procedure. It also includes an objective set (ideally quantified) of criteria for determining whether the results of the procedure are of the desired quality.

Unit testingis a level of software testing where individual units/ components of a software are tested. The purpose is to validate that each unit of the software performs as designed. A unit is the smallest testable part of any software. It usually has one or a few inputs and usually a single output.

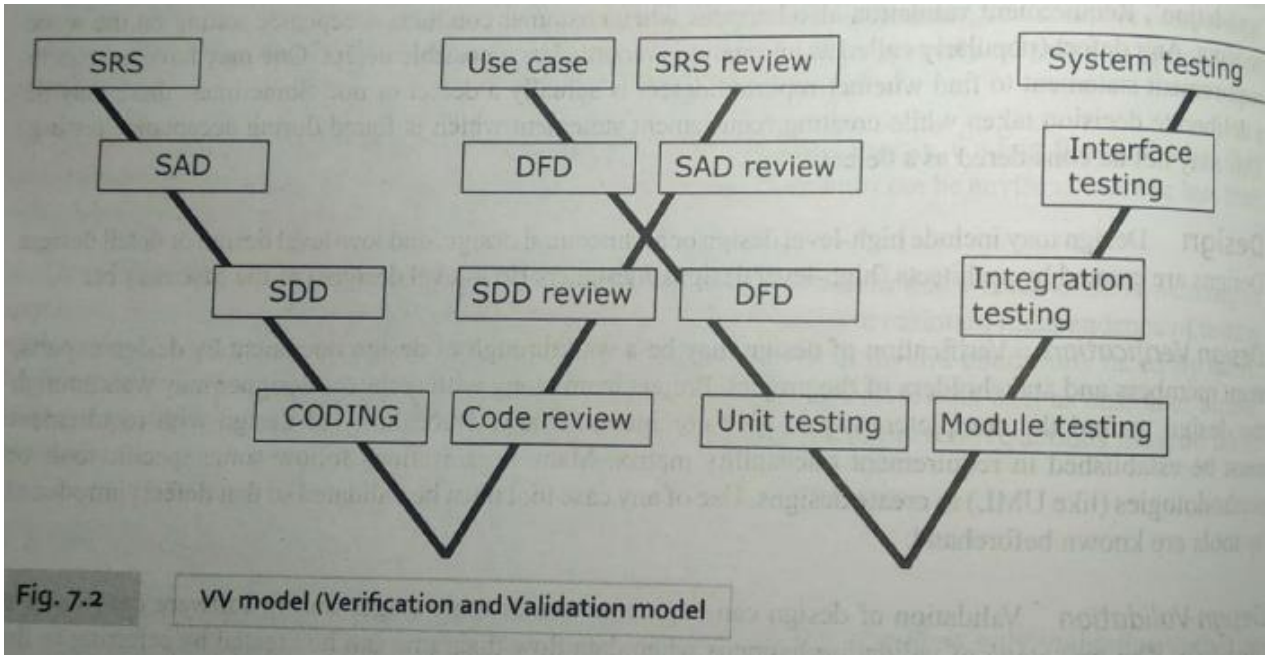


Fig. 7.2 VV model (Verification and Validation model

Q4f. What are the roles and responsibilities of a reviewer.

Ans:

1. Moderator: The Moderator is the key role in a code review. The moderator is responsible for selecting a team of reviewers, scheduling the code review meeting, conducting the meeting, and working with the author to ensure that necessary corrections are made to the reviewed document.

2. **Author**: The Author wrote the code that is being reviewed. The author is responsible for starting the code review process by finding a Moderator. The role of Author must be separated from that of Moderator, Reader, or Recorder to ensure the objectivity and effectiveness of the code review. However, the Author serves an essential role in answering questions and making clarifications during the review and making corrections after the review.

3. **Reader**: The Reader presents the code during the meeting by paraphrasing it in his own words. It is important to separate the role of Reader from Author, because it is too easy for an author to

explain what he meant the code to do instead of explaining what it actually does. The reader's interpretation of the code can reveal ambiguities, hidden assumptions, poor documentation and style, and other errors that the Author would not be likely to catch on his own.
4. **Scribe**: The Scribe records all issues raised during the code review. Separating the role of Scribe from the other roles allows the other reviewers to focus their entire attention on the code.

Question 5

Q5a. What is integration testing? Explain the Big bang approach.

Ans: .

Integration testing:

- Integration means combining. For Example, In this testing phase, different software modules are combined and tested as a group to make sure that integrated system is ready for system testing.
- Integrating testing checks the data flow from one module to other modules.
- This kind of testing is performed by testers.

Characteristics of Big bang approach

Big Bang Integration Testing is an integration testing strategy wherein all units are linked at once, resulting in a complete system. When this type of testing strategy is adopted, it is difficult to isolate any errors found, because attention is not paid to verifying the interfaces across individual units.

- Testing is a last phase of development life cycle when everything is finalized
- Characterized by huge rework, retesting, scrap, sorting
- Regression testing reveals many issues as correction may not be correct
- All requirements and designs cannot be covered in testing
- Major part of software build never gets tested

Disadvantages of Big-Bang approach

- Defects present at the interfaces of components are identified at very late stage as all components are integrated in one shot.
- It is very difficult to isolate the defects found.
- There is high probability of missing some critical defects, which might pop up in the production environment.
- It is very difficult to cover all the cases for integration testing without missing even a single scenario.

Q5b. What is the need of a Security Testing?

Ans:

Security Testing:

- Security testing is a special type of testing intended to check the level of security and protection offered by an application to the users against unfortunate incidences.
- The incidence could be loss of privacy, loss of data etc.
- There are some weak points in a system which is vulnerable to outside attacks/unauthorized entry in the system.
- Security testing need to follow validation activities to prove that system is protected enough against external attacks.
- No system is fully protected from all types of attacks.

Some definitions associated with security are given below:

Vulnerability:

- No system in the world is perfect. There is no system exist which does not have any vulnerability, therefore one must take precaution to not expose these weak points to outside.

Threats:

- A threat represents the possible attacks on the system from outsiders with malicious intentions.
- Threat is defined as an exploitation of vulnerabilities of the system.
  Perpetrators:
- Perpetrators are the entities who are unwelcomed guest in the system.
- The can create a problem in a system by doing something undesirable like loss of data and making changes in system.
- Perpetrator can be people, other system, viruses, etc.

***Q5c.* What is performance testing? List different types of performance testing.**

Ans:

Performance Testing:

- Performance testing is intended to find whether the system meets its performance requirements under normal level of activities.
- Generally, system performance requirements are identified in requirement statement defined by the customer and system design implements them. Performance criteria must be expressed in numerical terms.
- Design verification can help in determining whether required measures meet with performance requirements or not.
- This is the situation where verification does not work at that extent, and one need to test it by actually performing the operations on the system.
-

The focus of Performance Testing is checking a software program's
Speed - Determines whether the application responds quickly
Scalability - Determines maximum user load the software application can handle.
Stability - Determines if the application is stable under varying loads

Performance criteria must be measurable in quantitative terms such as time in 'milliseconds'. Some examples of performance testing can be given below:

- Adding a new record in a database must take maximum five milliseconds.
- Database containing one million records must not take more than one second while searching a record in a database.
- Sending information across the system size of 1 MB with a network of 512 KBPS must not take more than one minute.

Types of Performance Testing

- Load testing - checks the application's ability to perform under anticipated user loads. The objective is to identify performance bottlenecks before the software application goes live.
- Stress testing - involves testing an application under extreme workloads to see how it handles high traffic or data processing. The objective is to identify the breaking point of an application.
- Endurance testing - is done to make sure the software can handle the expected load over a long period of time.
- Spike testing - tests the software's reaction to sudden large spikes in the load generated by users.

- Volume testing - Under Volume Testing large no. of. Data is populated in a database and the overall software system's behavior is monitored. The objective is to check software application's performance under varying database volumes.
- Scalability testing - The objective of scalability testing is to determine the software application's effectiveness in "scaling up" to support an increase in user load. It helps plan capacity addition to your software system.

Q5d. Explain the concept of inter system testing and its Importance.

Ans: Many a times, an application is hosted across locations; however, all data needs to be deployed over a central location. The process of testing the integration points for single application hosted at different locations and then ensuring correct data flow across each location is known as inter system testing.

Objective

- Determine Proper parameters and data are correctly passed between the applications.
- Documentation for involved system is correct and accurate.
- Ensure Proper timing and coordination of functions exists between the application system.

Use

- When there is change in parameters in application system.
- The parameters, which are erroneous then risk associated to such parameters, would decide the extent of testing and type of testing.
- Intersystem parameters would be checked / verified after the change or new application is placed in the production.

Example

- Develop test transaction set in one application and passing to another system to verify the processing.
- Entering test transactions in live production environment and then using integrated test facility to check the processing from one system to another.
- Verifying new changes of the parameters in the system, which are being tested, are corrected in the document.

Q5e. Explain the significance of Usability testing.

Ans:

Usability testing

- *It is simple to understand application usage*
- *Help is available and user can use it effectively*
- *It is easy to execute an application Usability testing is done by,*
- *Direct observation of people using system*
- *Conducting usability surveys*
- *Beta testing or business pilot of application*

Usability testing checks for human factor problems such as,

- *Whether outputs from the system such as printouts, reports etc are meaningful or not*
- *Is error diagnostic straightforward or not*
- *Error messaging must help common users using the system*
- *Does User Interface have conformity to the syntax, format, style observations etc*
- *Is the application easy to use?*
- *Is there an exit option in all choices so that user can exit the system at any moment?*
- *System must not annoy intended user*

- *System taking control from user without indicating when it will be returned can be a problem*
- *System must provide online help or user manual*
- *Consistent in it's function and overall design*

Q5f. Explain Commercial off-the-shelf software testing.

Ans:

'COTS' stands for 'Commercially Off The Shelf', These software are readily available in the market and user can buy and use them directly.

There are many limitations for a development organisation that prevent it from making its own required software, and one has to make a decision to buy 'COTS'. Some of the reasons are given below:

- *Line of Business: An organisation developing software for banks, financial, institute etc. It may not be in a line of business to develop automation testing tool for its test requirements. In such case, the organisation buy software from outside and use it without investing much time and resources for making such software in-house.*
- *Cost-Benefits Analysis: Sometimes, it is very costly to develop software in-house due to various limitations like knowledge, skills, resources, etc. It would be easy to go to market and purchase that product. Because buying a product may be cost effective.*
- *Expertise/Domain Knowledge: An organisation may have knowledge about how to use the software that it needs. But it may not have development knowledge to build such software in-house. Such software can be bought from market and use without going in details of how it is built.*
- *Delivery Schedule: 'COTS' are available across the table by paying a price while developing such software in-house may take a very long time and huge effects.*

Features of COTS Testing:

- *'COTS' are developed with general requirements collected from the market. It may not exactly match with organisation's needs and expectations.*
- *Some 'COTS' may need changing business process to suite the 'COTS' implementation in organisation.*
- *This is another way of Business Process Reengineering(BPR) for an organisation where intentionally proven practices can be implemented by using 'COTS'.*
- *Sometimes, 'COTS' may need configuration of software or system to suite business model.*

Challenges in Testing 'COTS':

- *Requirement statement and design may not be available to testers as product manufacturer never share with any customer.*
- *Verification and validation records prepared during SDLC are very important for system testing and acceptance testing.*