Q. Write about s/w testing techniques

A

- Software Testing Techniques help you design better test cases. Since exhaustive testing is not possible; Manual Testing Techniques help reduce the number of test cases to be executed while increasing test coverage. They help identify test conditions that are otherwise difficult to recognize

- Boundary value analysis is based on testing at the boundaries between partitions. It includes maximum, minimum, inside or outside boundaries, typical values and error values.
- It is generally seen that a large number of errors occur at the boundaries of the defined input values rather than the center. It is also known as BVA and gives a selection of test cases which exercise bounding values.
- This black box testing technique complements equivalence partitioning. This software testing technique base on the principle that, if a system works well for these particular values then it will work perfectly well for all values which comes between the two boundary values.

- Boundary value analysis is based on testing at the boundaries between partitions. It includes maximum, minimum, inside or outside boundaries, typical values and error values.
- It is generally seen that a large number of errors occur at the boundaries of the defined input values rather than the center. It is also known as BVA and gives a selection of test cases which exercise bounding values.
- This black box testing technique complements equivalence partitioning. This software testing technique base on the principle that, if a system works well for these particular values then it will work perfectly well for all values which comes between the two boundary values.

- **Guidelines for Boundary Value analysis**
- If an input condition is restricted between values x and y, then the test cases should be designed with values x and y as well as values which are above and below x and y.
- If an input condition is a large number of values, the test case should be developed which need to exercise the minimum and maximum numbers. Here, values above and below the minimum and maximum values are also tested.
- Apply guidelines 1 and 2 to output conditions. It gives an output which reflects the minimum and the maximum values expected. It also tests the below or above values.
- **Example:**
- Input condition is valid between 1 to 10 Boundary values 0,1,2 and 9,10,11

- **Equivalence Class Partitioning**
- Equivalent Class Partitioning allows you to divide set of test condition into a partition which should be considered the same. This software testing method divides the input domain of a program into classes of data from which test cases should be designed.
- The concept behind this technique is that test case of a representative value of each class is equal to a test of any other value of the same class. It allows you to Identify valid as well as invalid equivalence classes.
- **Example:**
- Input conditions are valid between
- 1 to 10 and 20 to 30Hence there are three equivalence classes

--- to 0 (invalid) 1 to 10 (valid) 11 to 19 (invalid) 20 to 30 (valid) 31 to --- (invalid)

- **Decision Table Based Testing.**
- A decision table is also known as to Cause-Effect table. This software testing technique is used for functions which respond to a combination of inputs or events. For example, a submit button should be enabled if the user has entered all required fields.
- The first task is to identify functionalities where the output depends on a combination of inputs. If there are large input set of combinations, then divide it into smaller subsets which are helpful for managing a decision table.
- For every function, you need to create a table and list down all types of combinations of inputs and its respective outputs. This helps to identify a condition that is overlooked by the tester.

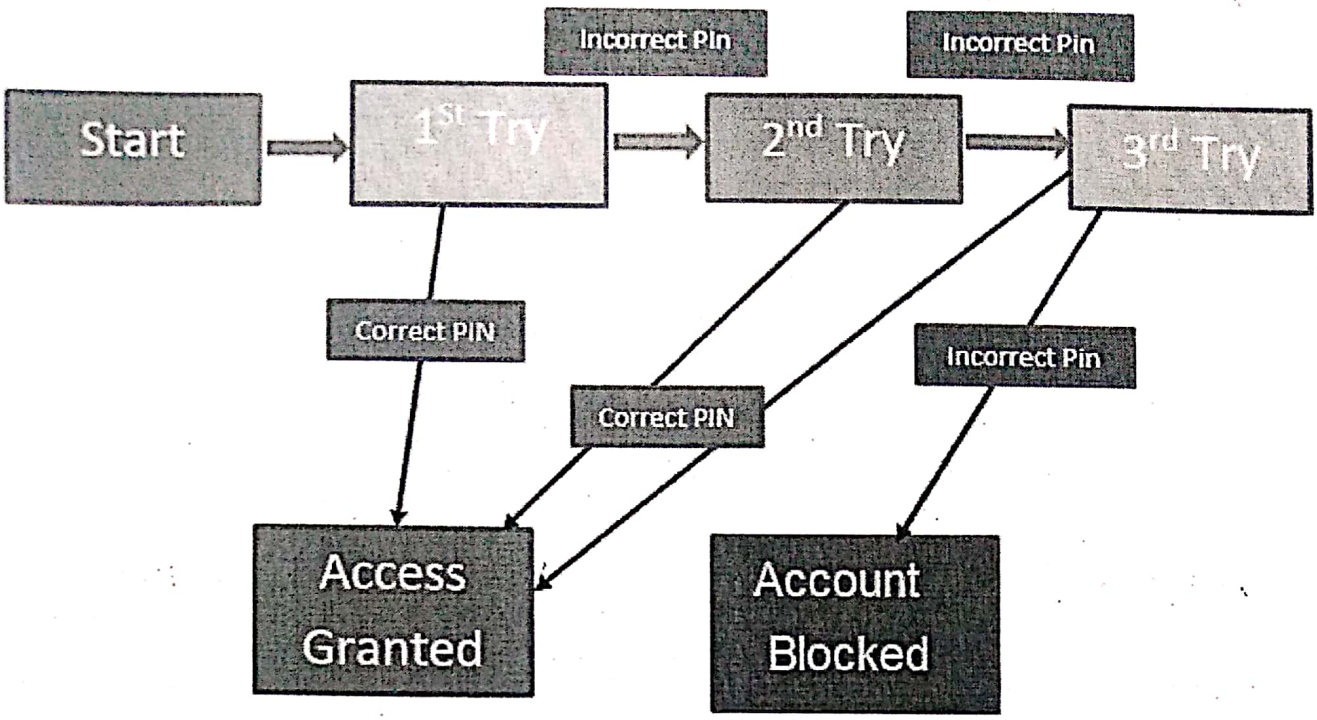
- **Following are steps to create a decision table:**
- Enlist the inputs in rows
- Enter all the rules in the column
- Fill the table with the different combination of inputs
- In the last row, note down the output against the input combination.
- **Example:** A submit button in a contact form is enabled only when all the inputs are entered by the end user.

	Rule 1	Rule 2	Rule 3	Rule 4	Rule 5	Rule 6	Rule 7	Rule 8
Input								
Name	F	T	F	T	F	T	F	T
Email	F	F	T	T	F	F	T	T
Message	F	F	F	F	T	T	T	T
Output								
Submit	F	F	F	F	F	F	F	T

- **State Transition**
- In State Transition technique changes in input conditions change the state of the Application Under Test (AUT). This testing technique allows the tester to test the behavior of an AUT. The tester can perform this action by entering various input conditions in a sequence. In State transition technique, the testing team provides positive as well as negative input test values for evaluating the system behavior.
- **Guideline for State Transition:**
- State transition should be used when a testing team is testing the application for a limited set of input values.
- The technique should be used when the testing team wants to test sequence of events which happen in the application under test.
- **Example:**

In the following example, if the user enters a valid password in any of the first three attempts the user will be able to log in successfully. If the user enters the invalid password in the first or second try, the user will be prompted to re-enter the password. When the user enters password incorrectly 3rd time, the action has taken, and the account will be blocked.

In this diagram when the user gives the correct PIN number, he or she is moved to Access granted state. Following Table is created based on the diagram above-



		Correct PIN	Incorrect PIN	
S1	START	S5	S2	
S2	1st Attempt	S5	S3	
S3	2nd Attempt	S5	S4	
S4	3rd Attempt	S5	S4	
S5	Access Granted	S5	S6	
S5	Access Granted	---	---	
S6	Account Blocked	---	---	

Error Guessing :

- Error guessing is a software testing technique which is based on guessing the error which can prevail in the code. It is an experience-based technique where the test analyst uses his/her or experience to guess the problematic part of the testing application.
- The technique counts a list of possible errors or error-prone situations. Then tester writes a test case to expose those errors. To design test cases based on this software testing technique, the analyst can use the past experiences to identify the conditions.
- The test should use the previous experience of testing similar applications
- Understanding of the system under test
- Knowledge of typical implementation errors
- Remember previously troubled areas
- Evaluate Historical data & Test results
- **Conclusion**
- Software testing Techniques allow you to design better cases. There are five primarily used techniques.
- Boundary value analysis is testing at the boundaries between partitions.
- Equivalent Class Partitioning allows you to divide set of test condition into a partition which should be considered the same.
- Decision Table software testing technique is used for functions which respond to a combination of inputs or events.
- In State Transition technique changes in input conditions change the state of the Application Under Test (AUT)
- Error guessing is a software testing technique which is based on guessing the error which can prevail in the code.

What is boundary value Analysis ?

Boundary value Analysis

A boundary value is the value on a boundary of an equivalence class.

- 1. A boundary value analysis is hence strongly related to equivalence class partitioning.
- Analysis Test Design Template

Test design item number :	Traces :
Based on : Input / Output	Assumptions
Type Description	Tag BT

The fields in the table are :

1.Test design item number -> unique identifier of the test design item.

- 2. Traces -> Refrences to the requirement or other descriptions covered by this test design.
- 3. Based on Input / Output -> Indication of which type of domain the design is based on.
- 4. Assumptions -> assumption must be documented.
- 5. Type -> VC -> VALID CLASS
- 6. IC -> INVALID CLASS
- 7. VB -> VALID BOUNDARY VALUE
- IB -> INVALID BOUNDARY value
- Desc -> The specification of the test condition.
- Tag -> Unique identification of the test condition.
- BT ->Belongs to indicates the class a boundary value belongs to .

Analysis Test Design Examples

1. Test conditions and test cases for the testing of this user requirement.
2. Test conditions and test cases for the testing of this user requirement.
3. Test design item number : 11

Identifying the valid class

- 1. consider any invalid classes.

2. Identify two obvious and two special invalid equivalence classes. New rows are indicated

Equivalence partitioning and Boundary value Analysis Hints

- 1. Reduce the number of test cases, because we can argue that one value from each equivalence partition is enough.
- 2.Find more faults because we concentrate our focus on the boundaries where the density of defects according to all experience is highest.
- 3.Input/ Output domains we are testing do not have one dimensional boundaries.
- 4. If it is not possible or feasible to partition our input or output domain in one dimension we have to use the technique called domain analysis instead.

Domain Analysis

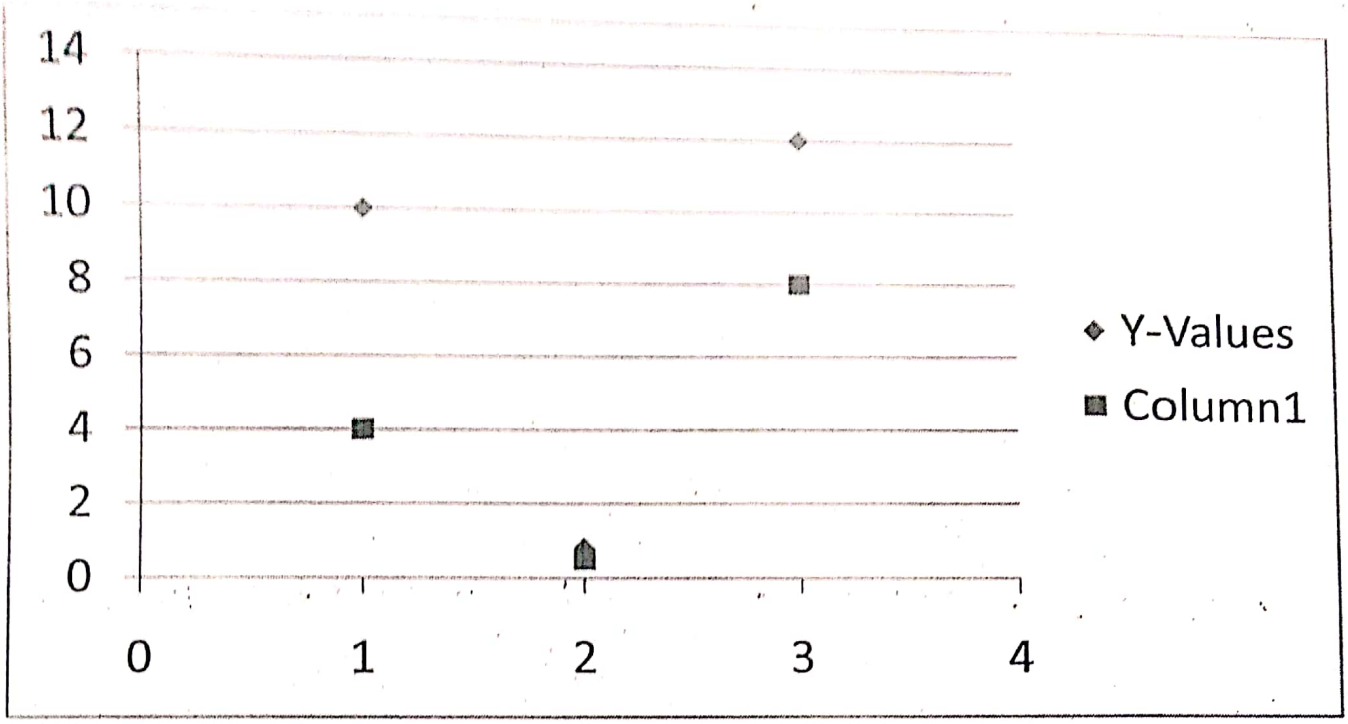
- 1. In equivalence partitioning of intervals where the boundaries are given by simple numbers.
- 2. We have one dimensional partitions.
- 3. The domain analysis test case design technique is used when our input partitions are multidimensional.
- 4. When a border for an equivalence partition depends on combinations of aspects or variables
- Test Progress Monitoring and Control
- . Need to collect information about facts, compare these with the estimates and analyze the findings
- 1. continuous monitoring of how the test is progressing compared to plan
- 2. collect information about facts.
- Compare these with the estimates and analyze the findings.
- Needed to minimize divergence from the test plan.

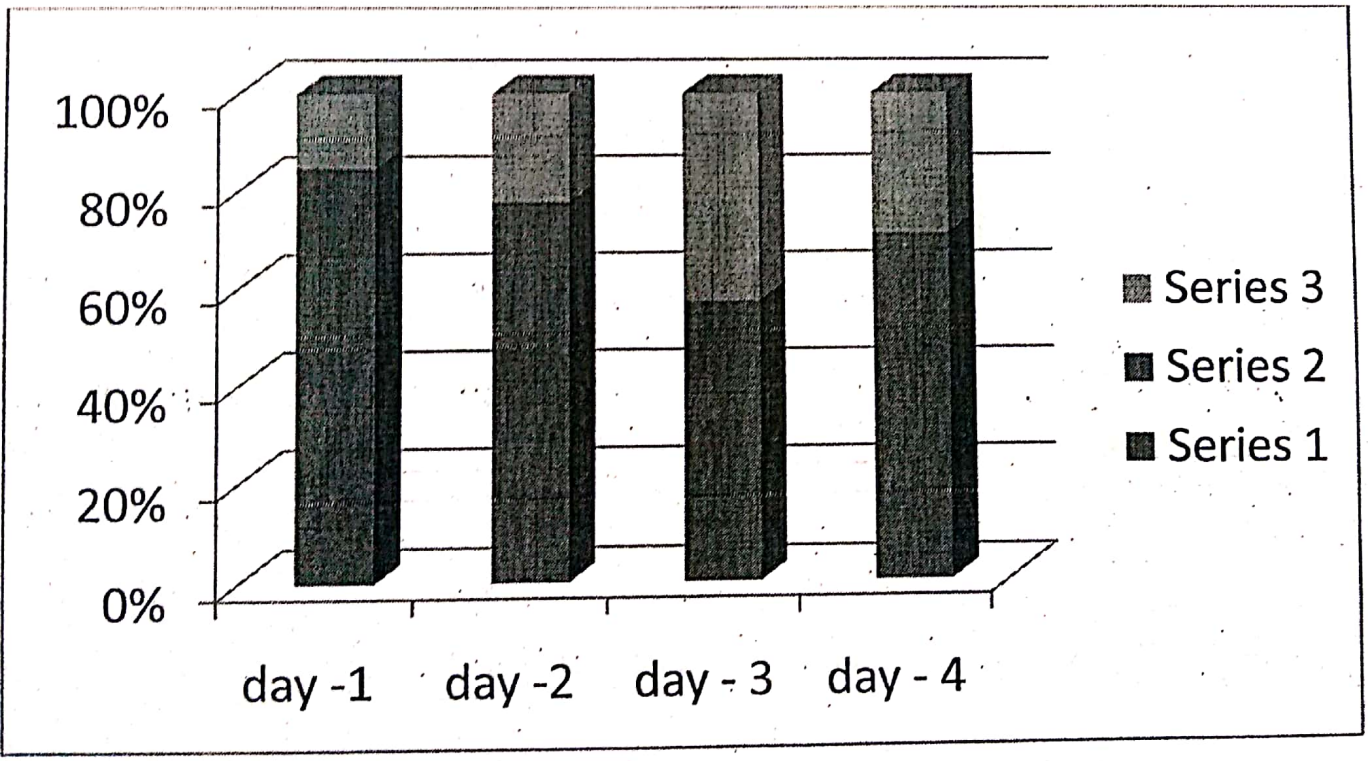
- Inform to stakeholders.
- Actual activities
- Honesty
- Visibility
- Action
- Collect information about reality.
- To make the information visible to all stakeholders.
- Information about progress and findings must be made readily available to the stakeholders in appropriate forms.

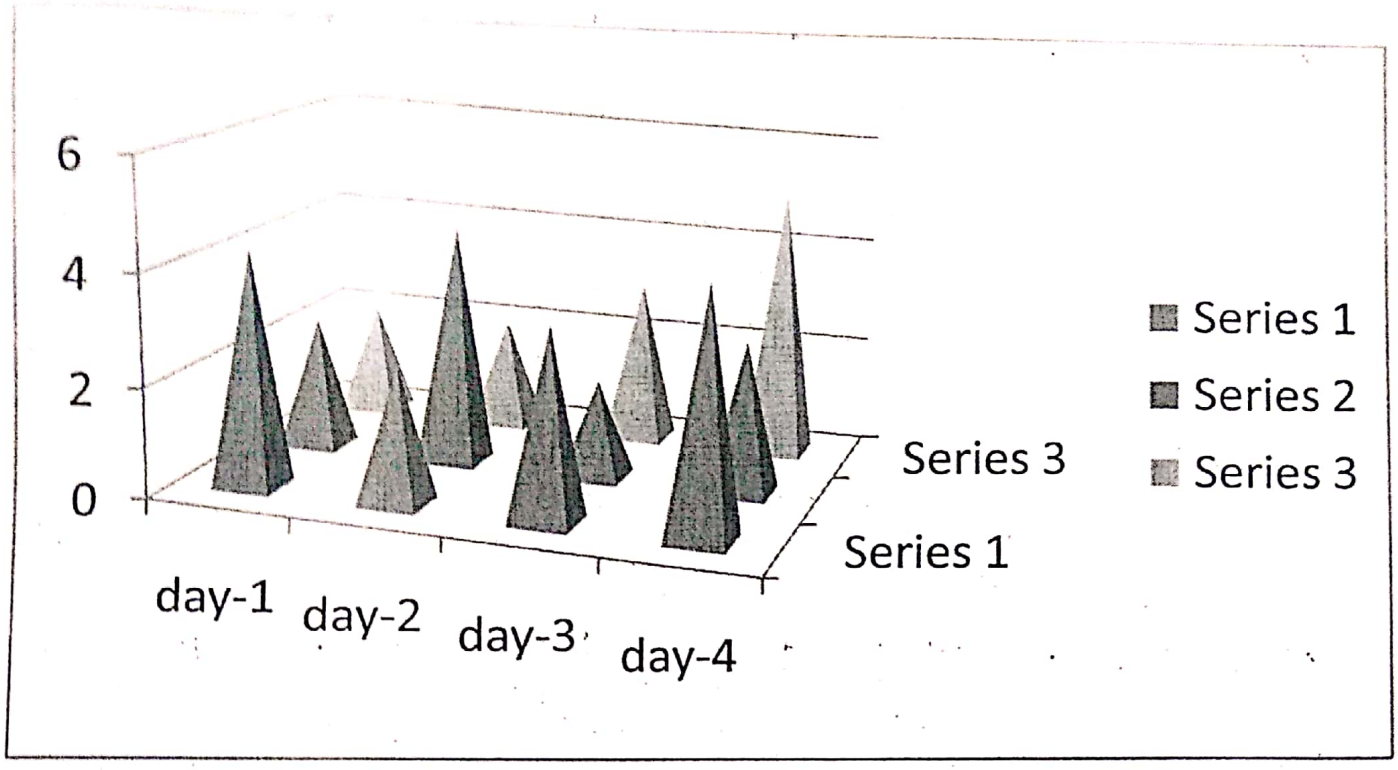
Collecting data

- The data to collect during testing should be specified in the approach section in the test plan, based on the requirements outlined in the policy and the strategy.
- No matter which data is collected .
- Just collect it.
- It must be presented and analysed to be of real value.
- Test reports are used to communicate test progress.
- These reports must be tailored to the different recipients or stakeholders.
- The test monitoring information are the customers, the project management and product management.
- The customer and the test management needs the report.
- The test mangement needs information on a continuous basis to keep in control.
- The picture speaks thousand words.
- The best way to present progress information for testing is by using graphics.

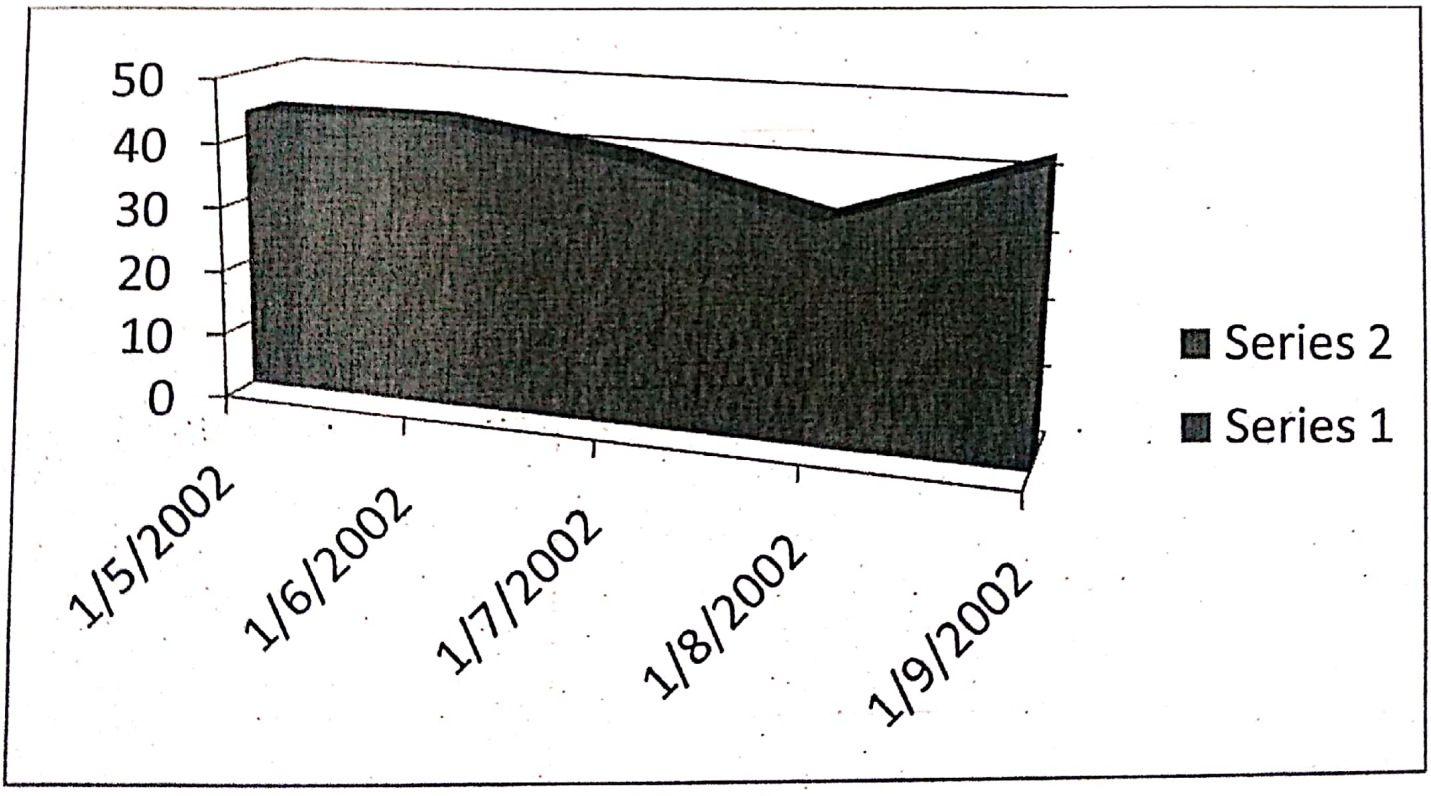
One-dimensional information







s-curves



- Used, simple useful way of presenting progress information and controlling .
- Many different metrics
- Test cases (run, attempted, passed, complted)

- Incidents (encountered, fixed, retested)
- S-curves us early warnings of something being wrong.
- 3 – phases
- 1. slow start 15-25%
- 2. productive phase 55-65%
- 3. difficult part – 10-25%

S-curves

1.Simple

2. Useful way of presenting progress information and controlling

- Because of the shape of the wanted curve.
- Used for many different metrics.
- Test cases
- Incidents
- Principle in s-curves is that our work falls in 3 phases.
- Phase-1 :slow start 15-25%
- Phase -2 : productive phase 55-65%
- Phase -3 : difficult part 10-25%
- Example: 300 test cases passed after 21 days of testing.
- Pie chart
- Pie chart are used to give an overview of the proportions of different aspects relative to each other.
- Most used in world
- Nice impression of the testing going well
- Easy and simple to prepare

Can be followed by everyone

Short answers

(1) Define check sheets?

Check sheets

A)

- Check sheets are a good way to give a quick overview of status
- Show progress compared to plan.
- Planned test cases, planned test areas.
- Presented as colourful graphics.
- Easy to produce
- Easy to understand.
- Wall sized check sheets are prepared and stick them in the corridor.
- James Bach's recommendations for the presentation are draw it on the wall, make it huge, and update it everyday.

(2) Define test techniques?

A)

Test techniques:

- Specification – based Techniques
- 1. specification based case design techniques are used to design test cases based on an analysis of the description of the product without references to its internal workings.
- 2. the techniques are also known a black-box tests.
- 3. these techniques focus on the functionality.
- 4. These should be in the form of requirements specifications, but may also be in the form of user manuals and process description.
- 5. these test case design techniques can be used in all stages and levels of testing.
- 6. The techniques can be used as a starting point in low-level tests.
- 7. Component testing and integration testing where test cases can be designed based on the design and on the requirements.
- 8. These test cases can be supplied with structural or white-box tests to obtain adequate coverage.
- 9. The techniques are also very useful in high level tests like acceptance testing and system testing where test cases are designed from the requirements.

Functional test case design techniques:

- 1.Equivalence partitioning and boundary value analysis.
- 2. Decision tables
- 3. Cause – effect graph
- 4. State transition testing
- 5. Classification tree method
- 6. Pairwise testing
- 7. Use-case testing
- 1.Equivalence partitioning and boundary value analysis
- A. designing test cases is about finding the input to test.
- B. It can be very difficult to figure out which input to choose for our test cases
- C. Equivalence partitioning test technique can help us handle situations with many input possibilities.
- Class :
- 1. A class is a portion of the domain.
- 2. The domain is said to be partitioned into classes if all members of the domain belong to exactly one class.
- 3. No member belongs to more than one class and no member falls outside the classes.
- 4. The reason for the equivalence partitioning is that all members in an equivalence class will either fail or pass the same test.
- 5. When we parition a domain into equivalence classes both valid and invalid classes.
- Test techniques :
- Test case design techniques are the heart of testing.
- There are many advantages of using techniques to design test cases.
- They are also extremely good for finding possible faults.
- They support systematic and meticulous work.
- Make the testing specification effective and efficient.
- The test case design techniques are based on models of the system.

UNIT-4

1. What is use-case Testing?

USE-CASE TESTING:

- 1.The concept of the use cases was first developed by the swedish Ivar Jacobsen in 1992.
- 2. A use-case or scenario as it is also called shows how the product interacts with one or more actors.
- 3. It includes a number of actions performed by the product as results of triggers from the actor.
- 4.An actor may be a user or another product with which the product in question has an interface.
- 5. Use cases are much used to express user requirements at an early state in the development.
- 6. Use cases should be presented in a structured textual form with a number of headings for which the relevant information must be supplied.

Example----preconditions

Use-Case	The name of the use case				Preconditions: Any prerequisite that must be fulfilled before the use case may be performed.				
	The name should be as desciptive as possible								
Purpose:	The goal of the use case.(What is achieved when it is completed).								
Actor :	Who(in terms of a predefined role) is interacting with the product.								

Guidelines:

- 1.A good use case provides a lot of useful information for testing purposes.
- 2. Identification of the use-case for tracebility purposes and the necessary preconditions directly from the form.
- 3.The high-level test cases can be extracted directly from the steps in the description.

Syntax testing:

- 1.Syntax is a set of rules.
- 2.Each defining the possible ways of producing a string in terms of sequences of iterations of or selections among other strings.
- 3. Syntax is defined for input to eliminate "garbage in".
- 4. For ex: web addresses :www.aaaa.aa
- 5. We can set up a list of rules, defining strings as building blocks and defining a notation to express the rules applied to the building blocks in a precise and compressed way.
- The building blocks are usually called the elements of the entire string.
- 2. The syntax rule for the string are defining must be given a name.
- 3. The most commonly used notation form is the Backus-Naur form. This form defines the following notations.
- | alternative separator "A" | "B"
- [ ] Optional item [" "]
- { } iterated item
- These notations can be used to form elements and the entire string.
- Ex:- pno = 2d[" "]2d[" "] 2d[" "] 2d
- Pno string must consists of foursets of two digits. Digit can range 0 – 9.The set of two digits can be separated by blanks.
- Valid string –39 62 36 48
- | alternative separator "A" | "B"
- [ ] Optional item [" "]
- { } iterated item
- These notations can be used to form elements and the entire string.
- Ex:- pno = 2d[" "]2d[" "] 2d[" "] 2d
- Pno string must consists of foursets of two digits. Digit can range 0 – 9.The set of two digits can be separated by blanks.

- Valid string –39 62 36 48

Test design item number :		Traces :		
Based on : Input / Output		Assumptions :		
Tag		Description		

2q. What is Defect Based Testing Technique????

Defect Based Testing Technique....

- A **Defect Based** Testing **Technique** is a **technique** where test cases are derived on the basis of **defects**. Instead of using the traditional requirements documents or the use cases (Specification-**based techniques**), this strategy uses the **defects** to base their test cases. The categorized list of **Defects** (called Defect Taxonomy) is being used.
- This technique can complement your test deriving conditions and can be taken as one of the options to increase the testing coverage. Or in some other sense – This technique can be applied when all the test conditions and test cases are identified and we need some extra coverage or insight into testing.
- Defect based technique can be used at any level of testing, but it is best suited in Systems Testing. We should base our test cases from the available defect taxonomies as well. These defects can be the production ones or historical ones. Root cause analysis can also be used to baseline your test cases.
- **The 5 step plan to write the test cases through Defect based testing techniques can be:**
- Identify and prioritize the Requirements
- Identify and collect all the Defects
- Brainstorm on the Defect
- Link the Defect to the Requirement

Write the Test conditions/Test cases based on the linked Defects

- 1.The defect-based techniques covered here are :
- 1. Taxonomies.
- 2. Fault injection and mutation.
- Taxonomy:-
- A **taxonomy** for **software testing** projects Abstract: This article establishes **taxonomy** for **software testing** projects, allowing the development team or **testing** personnel to identify the **tests** to which the project must be subjected for validation. The **taxonomy** is focused on identifying **software** projects according to their technology.
- · Bug **taxonomies** help in providing fast and effective feedback so that they can easily identify possible reasons for failure of the **software**. Using bug **taxonomy**, a large number of potential bugs can be grouped into few categories. Whenever a new bug...
- 1. A taxonomy is an ordered hierarchy of names for something.
- 2. It is an ordered hierarchy of possible defect types.
- 3. Many people have worked on defect taxonomies starting with Beizer's bug taxonomy defined in the late 1980s.

Levels are:

1. Requirements ->Requirements incorrect , Requirement logic, Requirements completeness, verifiability presentation, documentation, Requirements changes
2. 1.Requirements
3. 2.Features and functionality
4. 3. Structural Bugs
5. 4. Data
6. 5. Implementation and coding
7. 6. Integration
8. 7. System and s/w Architecture
9. 8. Test Definition and Execution

3q. What is Fault Injection and Mutation?????

- 1. Fault (or defect) injection and mutation are a type of technique where the product under test is changed in controlled ways and then tested.
- 2. This technique type is used to assess the effectiveness of the prepared test cases, rather than actually looking for defects.

- 3. Fault injection is also known as fault **seeding** or **debugging.**
- 4. Defects are deliberately inserted into the source code, either by hand or by the use of tools.
- 5. The defects inserted may be inspired by a checklist or a defect taxonomy.
- 6. The product is tested and it is determined how many of the injected defects are found and how many other defects are found.
- 7. Technique helps us identify more defects of the inserted types.
- 8. Fault injection may also be applied to data in the sense that data may be edited to be wrong compared to the expected data.
- 9. It must be clear what the correct code is and what code has been deliberately changed.

4q. What is experience-based Testing Technique????

- 1. Error Guessing
- 2. Checklist – based
- 3. Exploratory testing
- 4. Attacks.

Exploratory testing.....

- . philosophy when people are looking for oil or mines.
- 2. It is the philsophy in exploratory testing.
- 3. ET is the testing where the tester actively controls the design of the tests as those tests are performed and uses information gained while testing to design new and better tests.
- 4. Learning
- 5. Test Design
- 6. Test Execution.
- 1. ET is an important supplement to structured testing.

- 2. As with all the nonsystematic techniques it may be used before the structured test has stopped.
- 3. The exploratory tester needs to be able to analyze, reason and make decisions.
- 4. Systematic approach and be creative.
- **What is Static analysis tools in software testing?**
- Static analysis tools are generally used by developers as part of the development and component testing process. ...
- These tools are mostly used by developers.
- Static analysis tools are an extension of compiler technology – in fact some compilers do offer static analysis features. ...
- Other than software code, static analysis can also be carried out on things like, static analysis of requirements or static analysis of websites (for example, to assess for proper use ...
- **Static analysis** involves no dynamic execution of the **software** under **test** and can detect possible defects in an early stage, before running the program. **Static analysis** is done after coding and before executing unit tests. **Static analysis** can be done by a machine to automatically "walk through" the source code .
- **Dynamic analysis in Software Testing**
- § The testing of SW is seen as the execution of the test object § The test object is provided with test data § At the beginning, the test ...
- Three types of systematic technique Static (non-execution) • examination of documentation, source code listings, etc. ...

- Black Box Testing Technique § Here the test object is seen as a black box § It is the specification-based testing techniques § Also known as input/output-driven testing techniques ...

- **4q.What is Dynamic Testing or Dynamic Analysis ????**

- Dynamic testing (or dynamic analysis) is a term used in software engineering to describe the testing of the dynamic behavior of code. That is, dynamic analysis refers to the examination of the physical response from the system to variables that are not constant and change with time. In dynamic testing the software must actually be compiled and run. It in... New content will be added above the current area of focus upon selection

It involves working with the software, giving input values and checking if the output is as expected by executing specific test cases which can be done manually or with the use of an automated process. This is in contrast to static testing. Unit tests, integration tests, system tests and acceptance tests utilize dynamic testing. Usability tests involving a mock version made in paper or cardboard can be classified as static tests when taking into account that no program has been executed; or, as dynamic ones when considering the interaction between users and such mock version is effectively the most basic form of a prototype