Join Telegram Group :- @AKTU_Group
www.aktutor.in

Quantum Series
ENGINEERING

QUANTUM *Series*

Semester - 7 CS & IT

Software Testing

- Topic-wise coverage of entire syllabus in Question-Answer form.
- Short Questions (2 Marks)

Session 2021-22 Odd Semester

Join Telegram Group :- @AKTU_Group
Scanned by TapScanner

Join Telegram Group :- @AKTU_Group www.aktutor.in

CONTENTS

KCS 076 : Software Testing

UNIT-1: REVIEW OF SOFTWARE ENGINEERING (4 E—53 E)

Overview of software evolution, SDLC, Testing Process, Terminologies in Testing: Error, Fault, Failure, Verification, Validation, Difference between Verification and Validation, Test Cases, Testing Suite, Test Oracles, Impracticality of Testing All data; Impracticality of testing AllPaths.
Verification: Verification methods, SRS verification, Source code reviews, User documentation verification, Software project audit, Tailoring Software Quality Assurance Program by Reviews, Walkthrough, Inspection, and Configuration Audits.

UNIT-2 : FUNCTIONAL & STRUCTURAL TESTING (54 E—88 E)

Functional Testing: Boundary Value Analysis, Equivalence Class Testing, Decision Table Based Testing, Cause Effect Graphing Technique.
Structural Testing: Control flow testing, Path testing, Independent paths, Generation of graph from program, Identification of independent paths, Cyclomatic Complexity, Data Flow Testing, Mutation Testing.

UNIT-3: REGRESSION TESTING (89 E—106 E)

Regression Testing: What is Regression Testing? Regression Test cases selection, Reducing the number of test cases, Code coverage prioritization technique.
Reducing the number of test cases: Prioritization guidelines, Priority category, Scheme, Risk Analysis.

UNIT-4 : SOFTWARE TESTING ACTIVITIES (107 E—145 E)

Software Testing Activities: Levels of Testing, Debugging, Testing techniques and theirApplicability, Exploratory Testing.
Automated Test Data Generation: Test Data, Approaches to test data generation, test data generation using genetic algorithm, Test Data Generation Tools, Software Testing Tools, and Software test Plan.

UNIT-5 : OBJECT ORIENTED TESTING (146 E—180 E)

Object oriented Testing: Definition, Issues, Class Testing, Object Oriented Integration and System Testing.
Testing Web Applications: Web testing, User interface Testing, Usability Testing, Security Testing, Performance Testing, Database testing, Post Deployment Testing.

VERY SHORT ANSWER TYPE QUESTIONS (181 E—208 E)

SOLVED PAPERS (2016-17 TO 2018-19) (1 E—20 E)

Join Telegram Group :- @AKTU_Group Scanned by TapScanner

Join Telegram Group :- @AKTU_Group
www.aktutor.in

1 UNIT

Review of Software Engineering and Verification

Part-1 ..... (5E - 29E)

- *Overview of Software Evolution*
- *SDLC*
- *Terminologies in Testing : Error, Fault, Failure*
- *Verification*
- *Difference between Verification and Validation*
- *Testing Process*
- *Validation*

A. Concept Outline : Part-1 ..... 5E
B. Long and Medium Answer Type Questions 5E

Part-2 ..... (29E - 35E)

- *Test Cases*
- *Test Oracles*
- *Impracticality of Testing all Data*
- *Impracticality of Testing all Paths*
- *Testing Suites*

A. Concept Outline : Part-2 ..... 29E
B. Long and Medium Answer Type Questions 29E

Part-3 ..... (35E - 53E)

- *Verification Methods*
- *Source Code Review*
- *Software Project Audit*
- *Tailoring Software Quality Assurance Programme by Review*
- *Walkthrough*
- *Inspection and Configuration Audits*
- *SRS Verification*
- *User Documentation Verification*

A. Concept Outline : Part-3 ..... 35E
B. Long and Medium Answer Type Questions 35E

PART-1

Overview of Software Evolution, SDLC, Testing Process, Terminologies in Testing : Error, Fault, Failure, Verification, Validation, Difference between Verification and Validation.

CONCEPT OUTLINE : PART-1

- Software refers to a program or a set of instructions and applications used to manage and control various functions of a device.
- Software evolution refers to the process of developing software initially, then repeatedly updating it for various reasons.
- SDLC is a process followed for a software project, within a software organization.
- Verification is the system of confirming that software meets its specification.
- Validation is the system of confirming that software meets the user's requirement.

Questions-Answers

Long Answer Type and Medium Answer Type Questions

Que 1.1. **What is software ? What the types of software ?**

OR

What do you understand by the term software ?

Answer

1. Computer software, or simply software, is that part of a computer system that consists of encoded information or computer instructions, in contrast to the physical hardware from which the system is built.
2. Software refers to a program or a set of instructions and applications used to manage and control various functions of a device such as a computer.
3. Unlike hardware, which represents a physical part of a device, software is virtual.

Types of software :

Based on the goal, computer software can be divided into :

1. **Application software :** It is the software that uses the computer system to perform special functions or provide entertainment functions beyond

Join Telegram Group :- @AKTU_Group Scanned by TapScanner

Join Telegram Group :- @AKTU_Group www.aktutor.in

the basic operation of the computer itself. There are many different types of application software, because the range of tasks that can be performed with a modern computer is so large.

2. **System software :** It is the software that directly operates the computer hardware, to provide basic functionality needed by users and other software, and to provide a platform for running application software. System software includes :

   a. **Operating systems :**

      i. They are essential collections of software that manage resources and provides common services for other software that runs "on top" of them.

      ii. Supervisory programs, boot loaders, shells and window systems are core parts of operating systems.

      iii. In practice, an operating system comes bundled with additional software (including application software) so that a user can potentially do some work with a computer that only has an operating system.

   b. **Device drivers :**

      i. They operate or control a particular type of device that is attached to a computer.

      ii. Each device needs at least one corresponding device driver because a computer typically has at minimum at least one input device and at least one output device, a computer typically needs more than one device driver.

   c. **Utilities :** They are computer programs designed to assist users in the maintenance and care of their computers.

Que 1.2. **Explain the term software engineering.**

Answer

1. Before the emergence of software engineering, the software development was very risky and costly. At that time, software development process does not follow any systematic and scientific approach.
2. As a result, size of software and chances of errors increases, which causes the software crisis.
3. Software engineering follows the engineering approach for development of software that allows to overcome from the software crisis.
4. An engineering approach calls for need analysis, conceptualization of a product with specification, product design, process of manufacturing, testing, quality assurance, installation and implementation.
5. In engineering or scientific approach, the complex requirements of customer can be solved by using prototype approach.

   a. System engineering is concerned with all aspects of computer-based development including hardware, software and process engineering. Software engineering is part of this process.

   b. System engineers are involved in system specification, architectural design, integration and deployment.

6. Software, being a commercial product, calls for engineering approach to ensure that it is designed with correct choice of technology and architecture to :

   a. Achieve customer satisfaction.

   b. Ensure on-time delivery.

   c. Be developed within budgeted cost.

   d Provide ease of maintenance to meet changing requirements.

7. According to Boehm (1979), "Software engineering is the practical application of scientific knowledge in the design and construction of computer programs and associated documentation required to develop, operate and maintain them".

Que 1.3. **Give an overview of software evolution.**

Answer

Software evolution :

1. Software evolution is the term used in software engineering (specifically software maintenance) to refer to the process of developing software initially, then repeatedly updating it for various reasons.
2. The process of developing a software product using software engineering principles and methods is referred to as software evolution.
3. This includes the initial development of software and its maintenance and updates, till desired software product is developed, which satisfies the expected requirements.

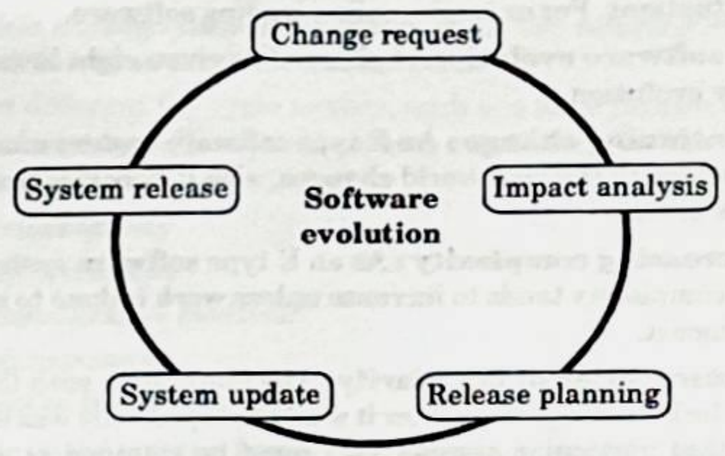


Fig. 1.2.1.

Join Telegram Group :- @AKTU_Group Scanned by TapScanner

Join Telegram Group :- @AKTU_Group www.aktutor.in

4. Evolution starts from the requirement gathering process after which developers create a prototype of the intended software and show it to the users to get their feedback at the early stage of software product development.
5. The users suggest changes, on which several consecutive updates and maintenance keep on changing too.
6. This process changes to the original software, till the desired software is accomplished.
7. Even after the user has desired software in hand, the advancing technology and the changing requirements force the software product to change accordingly.
8. Re-creating software from scratch and to go one-on-one with requirement is not feasible.
9. The only feasible and economical solution is to update the existing software so that it matches the latest requirements.

Software evolution laws :

Lehman has given laws for software evolution. He divided the software into three different categories :

1. **S-type (static-type) :** This is a software which works strictly according to defined specifications and solutions. The solution and the method to achieve it, both are immediately understood before coding. The s-type software is least subjected to changes hence this is the simplest of all. For example, calculator program for mathematical computation.
2. **P-type (practical-type) :** This is a software with a collection of procedures. This is defined by exactly what procedures can do. In this software, the specifications can be described but the solution is not obvious instantly. For example, gaming software.
3. **E-type (embedded-type) :** This software works closely as the requirement of real-world environment. This software has a high degree of evolution as there are various changes in laws, taxes etc. in the real world situations. For example, online trading software.

E-type software evolution : Lehman has given eight laws for E-type software evolution :

a. **Continuing change :** An E-type software system must continue to adapt to the real world changes, else it becomes progressively less useful.

b. **Increasing complexity :** As an E-type software system evolves, its complexity tends to increase unless work is done to maintain or reduce it.

c. **Conservation of familiarity :** The familiarity with the software or the knowledge about how it was developed, why was it developed in that particular manner etc., must be retained at any cost, to implement the changes in the system.

d. **Continuing growth :** In order for an E-type system intended to resolve some business problem, its size of implementing the changes grows according to the lifestyle changes of the business.

e. **Reducing quality :** An E-type software system declines in quality unless rigorously maintained and adapted to a changing operational environment.

f. **Feedback systems :** The E-type software systems constitute multi-loop, multi-level feedback systems and must be treated as such to be successfully modified or improved.

g. **Self-regulation :** E-type system evolution processes are self-regulating with the distribution of product and process measures close to normal.

h. **Organizational stability :** The average effective global activity rate in an evolving E-type system is invariant over the lifetime of the product.

Que 1.4. **What is SDLC ? Explain the various stages of a typical SDLC.**

OR

Write advantages and disadvantages of SDLC.

Answer

Software Development Life Cycle (SDLC) :

1. Software life cycle represents number of identifiable stages under which software goes during its life.
2. It's a diagrammatic representation which also provides description of various phases and their sequence in life cycle of software product.
3. Software undergoes some basic stages during its life cycle *i.e.*, requirement analysis and specification, design, coding and maintenance.
4. There are many software models which are used as per requirement of software product.
5. All models undergo these basic stages while the mapping of the stages may be different as per model requirement.
6. We have different life cycle models, each one have its own advantages and disadvantages. We can choose any one of them on the basis of :
   a. Development speed
   b. Product quality
   c. Project visibility
   d. Administrative overhead
   e. Risk exposure

Requirement of life cycle model :

1. Life cycle model create a frame which help in systematic and disciplined development of software.

Join Telegram Group :- @AKTU_Group Scanned by TapScanner

Join Telegram Group :- @AKTU_Group www.aktutor.in

2. Software development is not a 'one man show' that the person can decide by his own that which work (stage) is to do when, it is a team work which cannot be done by one man.
3. Hence, it is must that there should be a sequence how the work to be carried out in systematic way otherwise it will create problem. That's why there is need of life cycle model.
4. When the life cycle model is selected its documentation should be done simultaneously, as documentation help in developing quality software product through identifying inconsistencies and redundancies and removing them.
5. A System/Software life cycle model is a description of sequence of activities in a software engineering project and iterative order of their activities.
6. We must have a defined software life cycle model for our project to :
   a. Define the work to be performed.
   b. Divide up the work into manageable pieces.
   c. Determine project milestones at which project performance can be evaluated.
   d. Define the sequence of work units.
   e. Provide a framework for definition and storage of the deliverables produced by the project.
   f. Communicate your development strategy to project stakeholders.

Phases of software development life cycle models :

1. Requirement definition (system analysis and system specification)
2. System and component (software) design
3. Implementation and unit testing
4. Integration and system testing
5. Operation and maintenance

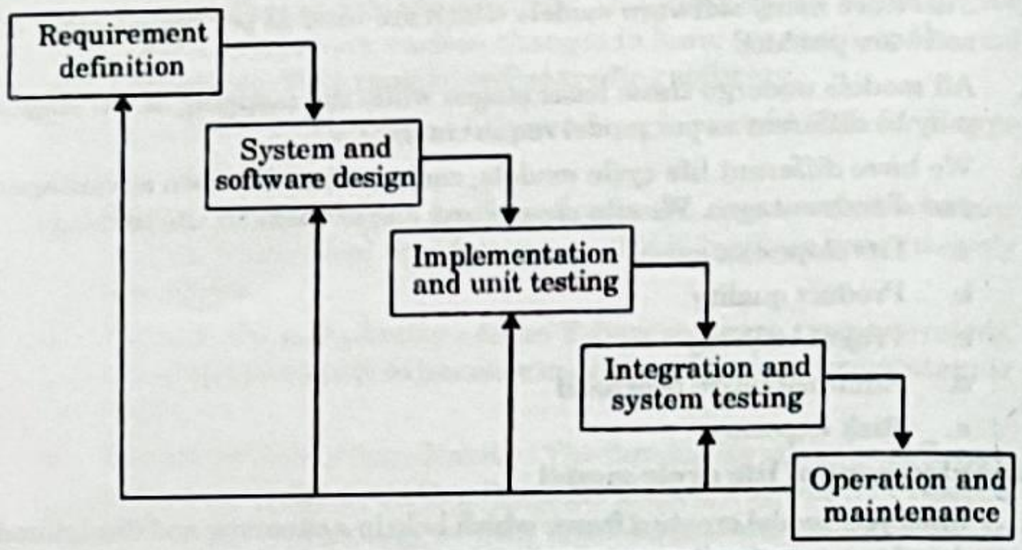


Fig. 1.4.1. Software development life cycle.

End product of each step :

1. The output of requirement definition is SRS document.
2. The output of system and software design is ER, DFD, flow chart etc.
3. The output of implementation and unit testing is coded program.
4. The output of integration and system testing is code without error or bugs.
5. The output of operation and maintenance phase is implemented system that is running well at client end.

Advantages of SDLC :

1. Formal review is created at the end of each stage allowing maximum management control.
2. This approach creates considerable system documentation.
3. This documentation ensures that system requirements can be traced back to stated business requirements.
4. It produces many intermediate products that can be reviewed to see whether they meet the user's needs and conform to standards. These can be further worked on if they require tweaks to be made, ensuring that the business gets exactly what it needs.

Disadvantages of SDLC :

1. What may be seen as a major problem for some, end-user does not see the solution until the system is almost complete.
2. Users get a system that meets the need as understood by the developers; this may not be what was really needed for them. There may be a loss in translation.
3. Documentation is expensive and time-consuming to create. It is also difficult to keep current. What may be current this month may not be the same this time next year!
4. Users cannot easily review intermediate products and evaluate whether a particular product (for example, data flow diagram) meets their business requirements.
5. Another disadvantage of a program or software that follows the SDLC program is, it encourages stiff implementation instead of creativity. There are requirements that must be met and that is all that developers complete.

Que 1.5. List the various software development life cycle models.

Answer

1. There are various software development life cycle models defined and designed which are followed during software development process.

Join Telegram Group :- @AKTU_Group Scanned by TapScanner

Join Telegram Group :- @AKTU_Group www.aktutor.in

2. Each process model follows a series of steps unique to its type, in order to ensure success in process of software development.
3. Following are the most important and popular SDLC models followed in the industry :
   a. Waterfall model
   b. Iterative model
   c. Spiral model
   d. V-model
4. The other related methodologies are Big Bang model, Agile model, RAD model - Rapid Application Development and Prototyping models.

Que 1.6. Explain waterfall model in detail. What are the advantages and disadvantages of it ?

OR

Write the various stages of waterfall model. What is the need waterfall model ?

Answer

1. Waterfall model is a theoretical software development model which was used in 70's. It is also known as classical, traditional, conventional or linear segment model.
2. There are different stages to the development and the output of first stage flow to the next (second) stage and output of second flows to third stage and so on.
3. It force on sequential phase development in which no phase can overlap another phase and so the developer must complete each phase before starting next phase.
4. Each phase of this model has a well defined starting and ending criteria which is to be documented by which the standard outputs (deliverables) to be produced by each phase can formulate.
5. This model does not allow to go back to the previous stage from one stage, "one way street with no turning back" like waterfall that's why it is called waterfall model.
6. The different phases of this model are :
   a. **Feasibility study :**
      i. This phase is used to check whether the new proposed system is economically, technically and operationally feasible or not.
      ii. In this, information is gathered about what outputs to be produce, input required and process can be used and then different solution strategies are formulated.
      iii. Finally, analysis of all solutions done on the basis of their cost and benefits and accordingly the best solution is selected.

      iv. It may be possible that none of the solution is feasible either due to technical or economical reason, in this case the project is dropped.
      v. The output of this phase is feasibility report.
   b. **Requirement analysis and specification :**
      i. This phase give specification about what is the system for.
      ii. This phase analyze and specifies the requirement of user/ customer and document them properly.
      iii. It is a very critical phase of software development life cycle as whole project is planned and scheduled here for development.
      iv. In requirement analysis, the data are gathered from users using different methods as interviews, questionnaires, on site observation and through written document of the organization, and then it is to be assured that the requirements are understood properly and no inconsistency, incompleteness and ambiguity are left.
      v. Finally, the requirements organized systematically in the form of document called Software Requirement Specification (SRS) document.
      vi. The main element of this document is functional requirement and non-functional requirement.
      vii. Functional requirement specifies required input data, required processing on data and output data to be produced.
      viii. Non-functional requirement specifies standard of software.
      ix. Then, SRS document is given to the user for approval.
      x. The output of this phase is SRS document.
   c. **System and software designing phase :** In design phase, overall structure or architecture is developed which is transformation of requirement specified in SRS. There are two main types of design approach :
      i. **Traditional design approach :** In this approach, first structural analysis of requirement is done which is obtained from SRS. For which data flow diagrams (DFD) are used which show flow of information among functions. In this function, SRS document is decomposed into small sub functions and information flow among these sub functions are also checked. After structural analysis, structural design starts which is done in two parts :
         - **Architectural design :** In this design, whole system is divided into modules and relationship is developed among these modules.

Join Telegram Group :- @AKTU_Group Scanned by TapScanner

Join Telegram Group :- @AKTU_Group www.aktutor.in

- **Detailed design :** In this phase, previous stages (modules) are expanded for detail design *i.e.*, data structure and documentation for modules are done here.

ii. **Object oriented approach :** This is new software design approach in which problem and their solution are represented in the form of object and then relationship is developed among these objects. This approach require less time and efforts for development and give better result as it is easy to maintain. The output of this phase is module or object design.

d. **Coding and module testing :**

i. In this phase, system design is translated into source code also called program code.

ii. Here, programming for different module is done in selected programming language.

iii. End product of coding phase is module testing in which each module is tested individually to check whether they are working properly or not, this is also called unit testing.

iv. The output of this phase is programmed module.

e. **Integration and system testing :**

i. Individually tested modules are integrated here according to planned system to develop the system.

ii. Generally, in this integration, all the modules are not joined together to form the system rather than it is done in various steps, and during these steps the partially integrated system is tested and then the next module is added to it and again the testing is done.

iii. This help to find out whether the developed system conforms the requirement or not.

iv. There are two main testing : ALPHA testing which is done at developer end and BETA testing which is done at user end.

v. The output of this phase is testing and integration report.

f. **Implementation / Installation and maintenance :**

i. In this phase, system is installed at the user end and it is checked if there is any upgradation required in hardware or software element at user end as per our software, so that it is made available.

ii. Training is given to user staff for using the system.

iii. Once the software is properly installed there is need of maintaining the software.

iv. This ensures that software is working properly at user site.

v. The maintenance requires much more efforts than software development.

vi. It is of three types :

a. **Corrective maintenance :** It remove the uncovered errors of the software which were not noticed during software development.

b. **Perfective maintenance :** Adding new functions to software as per user requirement for enhancing the capacity of software.

c. **Adaptive maintenance :** Making changes in the software so that it can adopt new working environment.

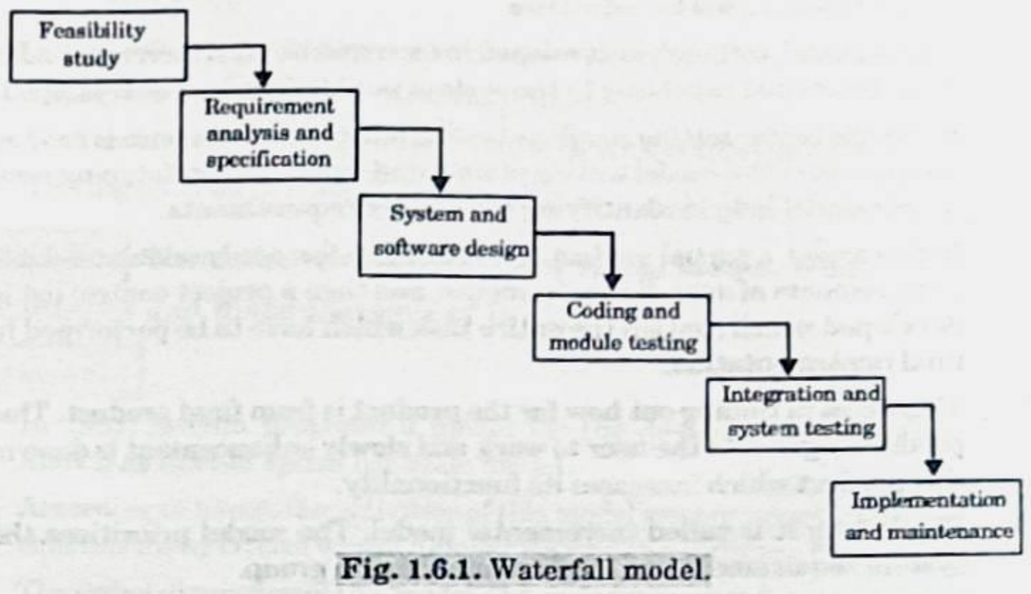


Fig. 1.6.1. Waterfall model.

Advantages of waterfall model :

1. Easy to understand.
2. Each stage has defined input and output.
3. Helps in project planning.

Disadvantage of waterfall model :

1. Iteration not possible as it is one way street.
2. Requirements freezing at starting stage.
3. Sequencing – no stage can start until the previous stage is completed.
4. A rigid model.
5. Difficulty in accommodating changes after project development.
6. Customer gets opportunity very late to review the project so less user involvement during development process.

Que 1.7. **Explain iterative model in detail. What are the advantages and disadvantages of it ?**

OR

Join Telegram Group :- @AKTU_Group Scanned by TapScanner

Join Telegram Group :- @AKTU_Group www.aktutor.in

Write the various stages of iterative enhancement model.

Answer

1. The classical waterfall model work on the concept that once the requirements specified, no further change will require in any phase of life cycle of product.
2. However, it is not practically possible. Iterative model is developed to overcome this drawback of waterfall model.
3. It is a combination of benefits of waterfall and prototype model. It is very popular model used by industries.
4. In this model, software is developed in increments, each increment adds some functional capability to the system until full system is developed.
5. It provide better testing result as testing after each increment is easy as compare to entire model testing of waterfall model and prototyping used in this model help in identifying the system requirements.
6. In this model, a partial product is developed on few easily understandable requirements of overall requirements, and then a project control list is developed which contain the entire task which have to be performed in final implementation.
7. This helps in finding out how far the product is from final product. This product is given to the user to work and slowly enhancement is done in this product which increases its functionality.
8. That's why it is called incremental model. The model prioritizes the system requirement and implements them in group.
9. With each step, next task is removed from control list. Designing, coding and implementation of selected task is done and analysis of partial product is done for checking the performance after this increment and the list is updated.
10. The process is iterated until the control list is empty and final implementation of system is done.
11. In this model, developer themselves provide specification so they have good control over system development.

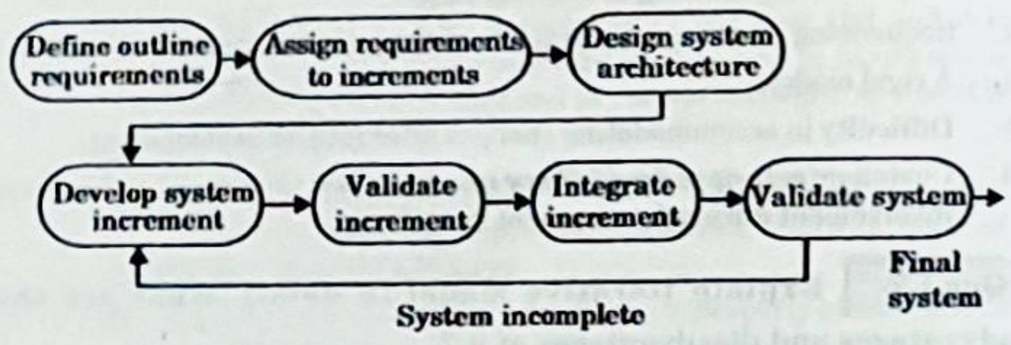


Fig. 1.7.1. Iterative enhancement model.

Advantages of iterative enhancement model :

1. Product delivered in parts so the cost of product is also distributed.
2. Easy testing as testing of the system is done after each increment.
3. Less human resource is required as product is delivered in parts.
4. Development of next phase and evolution of partial product can be done simultaneously.
5. User can see working of software in early stage.
6. User requirement become clearer.
7. Low failure risk.

Disadvantages of iterative enhancement model :

1. Total cost of product development is high.
2. Well defined project planning is required for distribution of work.
3. Testing each model causes extra cost.

Que 1.8. **Write the various stages of spiral model. What are the advantages and disadvantages of it ?**

Answer

1. In 1987, Boehm proposed a model for the development of software known as Boehm spiral life cycle model.
2. According to name, the activities of this model are organized like a spiral that has many circles whose number depends on software requirement.
3. The radial dimension of this model, the cumulative cost for accomplishing different stages (phases) and angular dimension show the progress in completing each cycle of the spiral.
4. The main objective of this model is to minimize the risk through the use of prototype. This model is mainly used for large projects.
5. The spiral model can said to be made up of waterfall model in which each stage is preceded by risk analysis.
6. Its main feature is risk avoidance rather than documentation or coding.
7. This model is more flexible than any other model as number of phases through which the product will be developed is not fixed, it depends on software requirement.
8. The two basic step of this model are :
   a. Identify the sub-problem which is having highest risk.
   b. Find solution for that particular problem (risk).
9. Generally, there are four spirals in Boehm spiral life cycle model.
10. The inner (first) spiral is concept development cycle, the second spiral indicates new product development cycle; the third spiral represents

Join Telegram Group :- @AKTU_Group Scanned by TapScanner

Join Telegram Group :- @AKTU_Group www.aktutor.in

product enhancement cycle and fourth spiral is known as product maintenance cycle.

11. Each phase of this model is split into four quadrant (sections) having specific functions :

a. In the first quadrant, we do identification of objectives; find out different alternative for achieving the objective and present constraints.

b. In the second quadrant, we evaluate these alternatives on the basis of objective and constraints. The main focus in this step is given on evolution of alternative on the basis of risk as risk causes the chances of unmet objectives. Risk is basically adverse conditions which may create abstraction in successful development of project. This model helps in coping up many project risks.

c. In the third quadrant, project development and validation is carried out.

d. In the fourth quadrant, the project is reviewed and decision is made up whether to continue with a further loop of spiral or not. If it is decided to continue, the project plans are drawn up for the next phase of project.

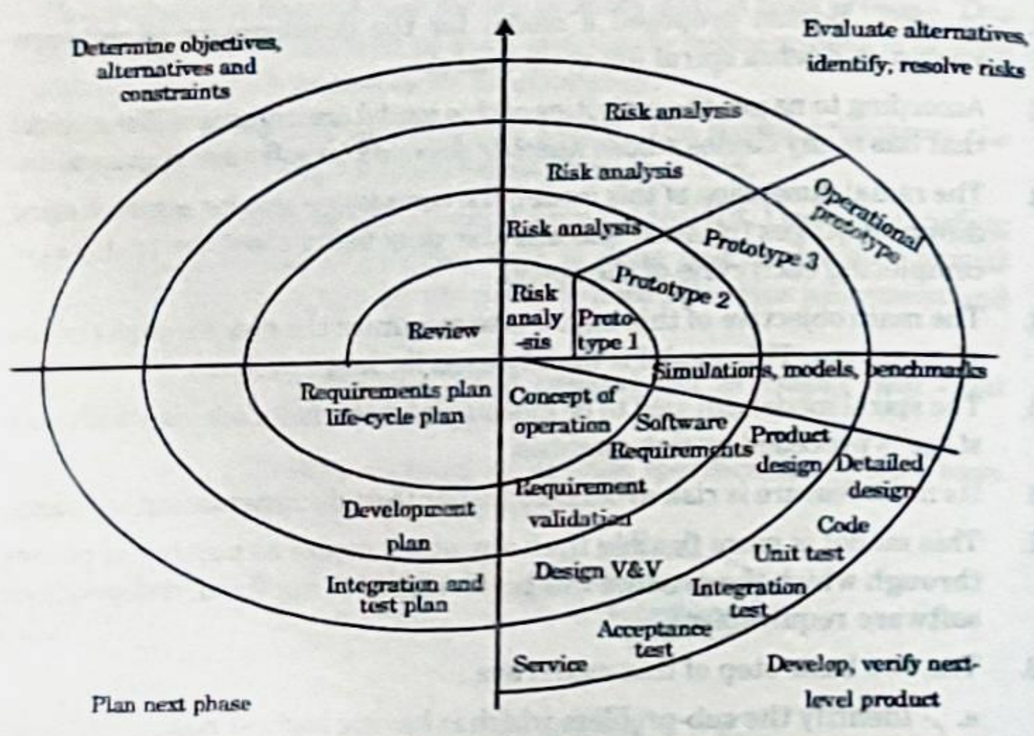


Fig. 1.8.1.

Advantages of Boehm spiral life cycle model :

1. This model tries to resolve all possible risks involved in the project starting with the highest risk.

2. User can see the product early in the life cycle.
3. In each phase, product is refined on the basis of customer feedbacks which ensure good quality.
4. Little documentation is required as compared to waterfall model.
5. Efficient use of prototyping and component based design.
6. It is very flexible model.
7. It can cope with changing user requirements.

Disadvantages of Boehm spiral life cycle model :

1. The model requires experts for risk management.
2. This model is not suitable for small projects.
3. This is time consuming model.
4. The cost of risk analysis is high which makes the model costly.
5. Different persons involve in the project may find it complex to use.
6. This model is not widely used because it is relatively new.

Advantages of spiral model over traditional iterative process model :

1. The risk analysis and validation steps eliminate errors in the early phase of development.
2. The model makes use of techniques like reuse, prototyping and component based design.
3. It becomes equivalent to another life cycle model in appropriate situations. The model is not suitable for small projects as cost of risk analysis may exceed the actual cost of the project.

Que 1.9. **Explain the verification and validation model with its different phases.**

OR

Discuss V- model in detail with their advantages and disadvantages.

Answer

1. The V - model is SDLC model where execution of processes happens in a sequential manner in V-shape.
2. It is also known as verification and validation model.
3. V-model is an extension of the waterfall model and is based on association of a testing phase for each corresponding development stage.
4. This means that for every single phase in the development cycle, there is a directly associated testing phase.
5. This is a highly disciplined model and next phase starts only after completion of the previous phase.

Join Telegram Group :- @AKTU_Group Scanned by TapScanner

Join Telegram Group :- @AKTU_Group www.aktutor.in

V- model design :

1. Under V-model, the corresponding testing phase of the development phase is planned in parallel.
2. So, there are verification phases on one side and validation phases on the other side.
3. Coding phase joins the two sides of the V-model.
4. Fig. 1.9.1 illustrates the different phases in V-model of SDLC.

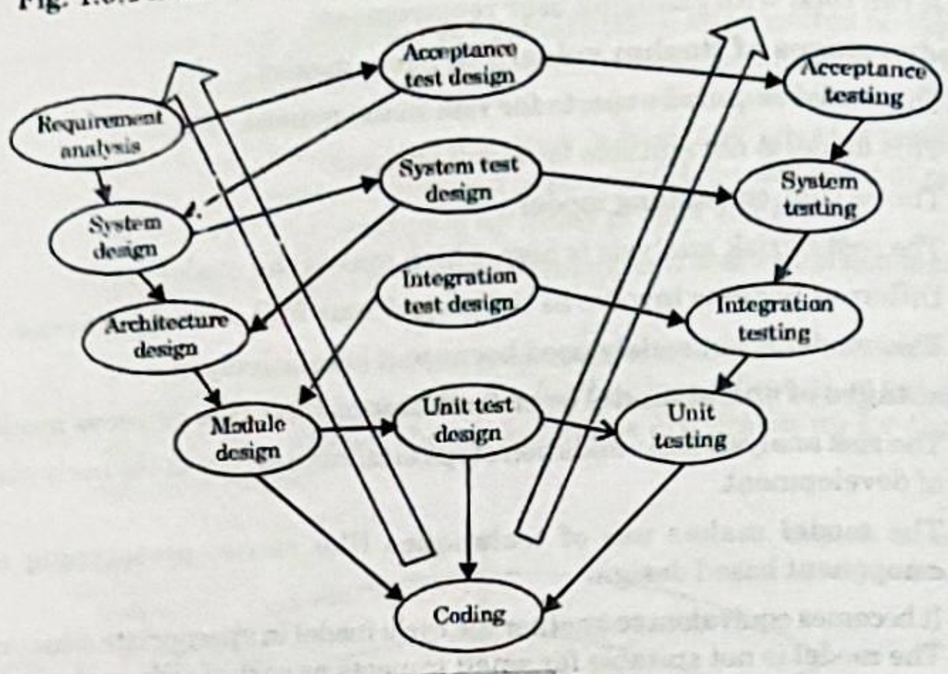


Fig. 1.9.1.

Verification phases : Following are the verification phases in V-model :

1. **Business requirement analysis :**
   a. This is the first phase in the development cycle where the product requirements are understood from the customer perspective.
   b. This phase involves detailed communication with the customer to understand his expectations and exact requirement.
   c. This is a very important activity and need to be managed well, as most of the customers are not sure about what exactly they need.
   d. The acceptance test design planning is done at this stage as business requirements can be used as an input for acceptance testing.
2. **System design :**
   a. Once you have the clear and detailed product requirements, it's the time to design the complete system.
   b. System design would comprise of understanding and detailing the complete hardware and communication setup for the product under development.

   c. System plan is developed based on the system design. Doing this at an earlier stage leaves more time for actual test execution later.
3. **Architectural design :**
   a. Architectural specifications are understood and designed in this phase.
   b. Usually more than one technical approach is proposed and based on the technical and financial feasibility the final decision is taken.
   c. System design is broken down further into modules taking up different functionality.
   d. This is also referred to as High Level Design (HLD).
4. **Module design :**
   a. In this phase, the detailed internal design for all the system modules is specified, referred to as Low Level Design (LLD).
   b. It is important that the design is compatible with the other modules in the system architecture and the other external systems.
   c. Unit tests are an essential part of any development process and helps eliminate the maximum faults and errors at a very early stage.
   d. Unit tests can be designed at this stage based on the internal module designs.
5. **Coding phase :**
   a. The actual coding of the system modules designed in the design phase is taken up in the coding phase.
   b. The best suitable programming language is decided based on the system and architectural requirements.
   c. The coding is performed based on the coding guidelines and standards.
   d. The code goes through numerous code reviews and is optimized for best performance before the final build is checked into the repository.

Validation phases : Following are the validation phases in V-model :

1. **Unit testing :**
   a. Unit tests designed in the module design phase are executed on the code during this validation phase.
   b. Unit testing is the testing at code level and helps eliminate bugs at an early stage, though all defects cannot be uncovered by unit testing.
2. **Integration testing :**
   a. Integration testing is associated with the architectural design phase.
   b. Integration tests are performed to test the coexistence and communication of the internal modules within the system.

Join Telegram Group :- @AKTU_Group Scanned by TapScanner

Join Telegram Group :- @AKTU_Group www.aktutor.in

3. **System testing :**
   a. System testing is directly associated with the system design phase.
   b. System tests check the entire system functionality and the communication of the system under development with external systems.
   c. Most of the software and hardware compatibility issues can be uncovered during system test execution.
4. **Acceptance testing :**
   a. Acceptance testing is associated with the business requirement analysis phase and involves testing the product in user environment.
   b. Acceptance tests uncover the compatibility issues with the other systems available in the user environment.
   c. It also discovers the non functional issues such as load and performance defects in the actual user environment.

Advantages of V- model :

1. This is a highly disciplined model and phases are completed one at a time.
2. It works well for smaller projects where requirements are very well understood.
3. Simple and easy to understand and use.
4. Easy to manage due to the rigidity of the model. Each phase has specific deliverables and a review process.

Disadvantages of V- model :

1. High risk and uncertainty.
2. Not a good model for complex and object-oriented projects.
3. Poor model for long and ongoing projects.
4. Not suitable for the projects where requirements are moderate to high risk of changing.
5. Once an application is in the testing stage, it is difficult to go back and change its functionality.

When to use V- model :

1. Requirement is well defined and not ambiguous.
2. Acceptance criteria are well defined.
3. Project is short to medium in size.
4. Technology and tools used are not dynamic.

Que 1.10. **What do you understand by testing process ? What are the basic steps of the testing process ?**

Answer

Testing is a process rather than a single activity. This process starts from test planning then designing test cases, preparing for execution and evaluating status till the test closure. So, we can divide the activities within the fundamental test process into the following basic steps :

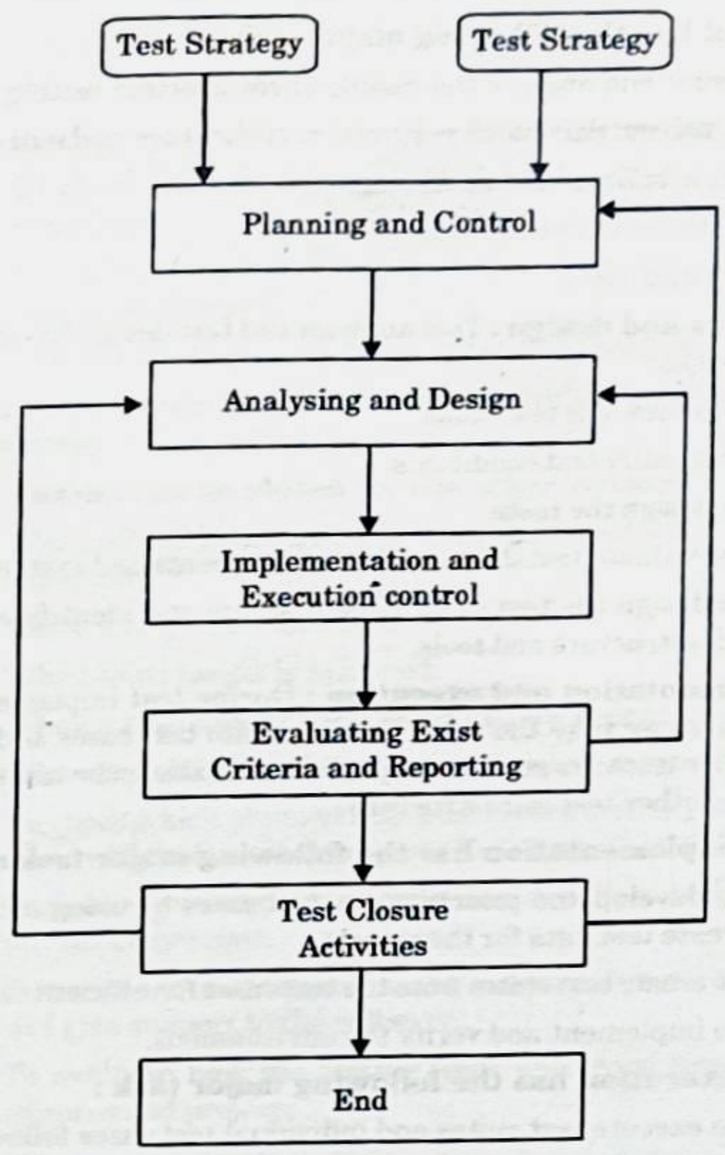


Fig. 1.10.1.

1. **Planning and control :** Test planning has following major tasks :
   a. To determine the scope and risks and identify the objectives of testing.
   b. To determine the test approach.
   c. To implement the test policy and/or the test strategy.

Join Telegram Group :- @AKTU_Group Scanned by TapScanner

Join Telegram Group :- @AKTU_Group www.aktutor.in

d. To determine the required test resources like people, test environments, PCs, etc.

e. To schedule test analysis and design tasks, test implementation, execution and evaluation.

f. To determine the exit criteria, we need to set criteria such as coverage criteria.

Test control has the following major tasks :

a. To measure and analyze the results of reviews and testing.

b. To monitor and document progress, test coverage and exit criteria.

c. To provide information on testing.

d. To initiate corrective actions.

e. To make decisions.

2. **Analysis and design :** Test analysis and test design has the following major tasks :

a. To review the test basis.

b. To identify test conditions.

c. To design the tests.

d. To evaluate testability of the requirements and system.

f. To design the test environment set-up and identify and required infrastructure and tools.

3. **Implementation and execution :** During test implementation and execution, we take the test conditions into test cases and procedures and other testware such as scripts for automation, the test environment and any other test infrastructure.

Test implementation has the following major task :

a. To develop and prioritize our test cases by using techniques and create test data for those tests

b. To create test suites from the test cases for efficient test execution.

c. To implement and verify the environment.

Test execution has the following major task :

a. To execute test suites and individual test cases following the test procedures.

b. To re-execute the tests that previously failed in order to confirm a fix. This is known as confirmation testing or re-testing.

c. To log the outcome of the test execution and record the identities and versions of the software under tests.

d. To compare actual results with expected results.

e. Whenever there are differences between actual and expected results, it report discrepancies as incidents.

4. **Evaluating exit criteria and reporting :** Based on the risk assessment of the project, we will set the criteria for each test level against which we will measure the "enough testing". These criteria vary from project to project and are known as exit criteria.

Exit criteria come into picture when :

a. Maximum test cases are executed with certain pass percentage.

b. Bug rate falls below certain level.

c. When achieved the deadlines.

Evaluating exit criteria has the following major tasks :

a. To check the test logs against the exit criteria specified in test planning.

b. To assess if more test are needed or if the exit criteria specified should be changed.

c. To write a test summary report for stakeholders.

5. **Test closure activities :** Test closure activities are done when software is delivered.

The testing can be closed for the other reasons :

a. When all the information has been gathered which are needed for the testing ?

b. When a project is cancelled.

c. When some target is achieved.

d. When a maintenance release or update is done.

Test closure activities have the following major tasks :

a. To check which planned deliverables are actually delivered and to ensure that all incident reports have been resolved.

b. To finalize and archive testware such as scripts, test environments, etc. for later reuse.

c. To handover the testware to the maintenance organization. They will give support to the software.

d. To evaluate how the testing went and learn lessons for future releases and projects.

Que 1.11. **Discuss the terms fault, error and failure.**

OR

Are the fault, error, and failure terms synonymous to each other ? If not, then discuss the difference among them.

OR

What is bug ?

Join Telegram Group :- @AKTU_Group Scanned by TapScanner

Join Telegram Group :- @AKTU_Group www.aktutor.in

Answer

The terms like error, fault and failure are not synonymous and hence these should not be used interchangeably. All these terms are defined as :

Failure :

1. When the software is tested, failure is the first term being used.
2. It means the inability of a system or component to perform a required function according to its specification.
3. In other words, when results or behaviour of the system under test are different as compared to specified expectations, then failure exists.
4. Failure is the term which is used to describe the problems in a system or software on the output side or result side.

Fault/Defect/Bug :

1. Fault is a condition, that in actual, causes a system to produce failures.
2. Fault is synonymous with the words defect or bug. Therefore, fault is the reason embedded in any phase of SDLC (software development life cycle) and results in failure.

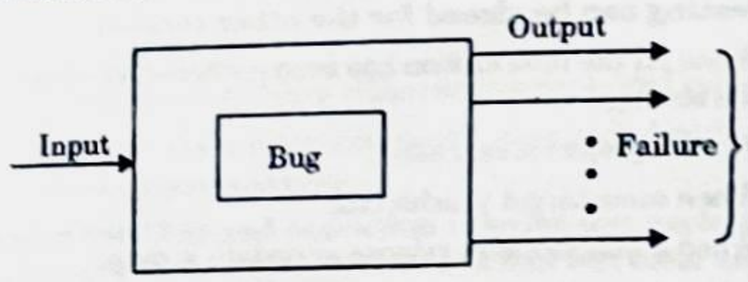


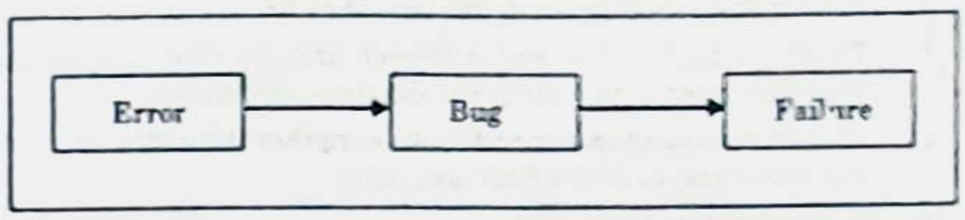
Fig. 1.11.1.

3. It can be said that failures are manifestation of bugs.
4. One failure may be due to one or more bugs or one bug may cause one or more failures.
5. Thus, when the bug is executed, then failures are generated.
6. But this is not always true.
7. Some bugs are hidden, in sense that these are not executed as they do not get the required conditions in the system.
8. So, hidden bugs may not always produce failures.
9. They may execute only in certain rare conditions.

Error :

1. Whenever a development team makes a mistake in any phase of SDLC, errors are produced.
2. It might be a typographical error, a misleading of a specification, a misunderstanding of what a subroutine does, and so on.
3. Error is very general term used for human mistakes.

4. Thus, an error causes a bug and the bug in turn causes failures as shown below :



Flow of faults

Fig. 1.11.2.

For example :

Let take a module in software as :

```
module A()
{
-----;
-----;
while(a>n + 1);
{
-----
printf("The value of x is", x);
}
-----
}
```

Suppose the module shown above is expected to print the value of x which is critical for the use of software. But when this module will be executed, the value of x will not be printed. It is a failure of the program.

When we try to look for the reason of this failure, we find that in module A(), the while loop is not being executed. A condition is preventing the body of while loop to be executed. This is known as bug/defect/fault.

On close observation, we find a semicolon (;) being misplaced after the while loop which is not its correct syntax and it is not allowing the loop to execute. This mistake is known as an error.

Que 1.12. **Explain the verification and validation.**

Answer

Verification :

1. Verification = "Are we building the project right".
2. Verification includes :
   a. Verifying process includes checking documents, design, code and program.
   b. It does not involve executing the code.
   c. Verification uses methods like reviews, walkthroughs, inspections and desk-checking etc.

Join Telegram Group :- @AKTU_Group Scanned by TapScanner

Join Telegram Group :- @AKTU_Group www.aktutor.in

d. Whether the software conforms to specification is checked.

e. It finds bugs early in the development cycle.

f. Target is application and software architecture, specification, complete design, high level and database design etc.

g. A team does verification and make sure that the software is as per the requirement in the SRS document.

h. It comes before validation.

Validation :

1. Validation = "Are we building the right project/product".
2. Validation includes :
   a. It is a dynamic mechanism of testing and validating the actual product.
   b. It always involves executing the code.
   c. It uses methods like black box testing, white box testing and non-functional testing.
   d. It checks whether software meets the requirements and expectations of customer.
   e. It can find bugs that the verification process can not catch.
   f. Target is actual product.
   g. With the involvement of testing team, validation is executed on software code.
   h. It comes after verification.

Que 1.13. **Compare validation and verification.**

Answer

S. No.	Validation	Verification
1.	Am I building the right product ?	Am I building the product right ?
2.	Determining if the system complies with the requirements and performs functions for which it is intended and meets the organization's goals and user needs. It is traditional and is performed at the end of the project.	The review of interim work steps and interim deliverables during a project to ensure they are acceptable. To determine if the system is consistent, adheres to standards, uses reliable techniques and prudent practices, and performs the selected functions in the correct manner.

3.	Am I accessing the right data (in terms of the data required to satisfy the requirements).	Am I accessing the data right (in the right place; in the right way).
4.	High level activity.	Low level activity.
5.	Performed after a work product is produced against established criteria ensuring that the product integrates correctly into the environment.	Performed during development on key artifacts, like walkthroughs, reviews and inspections, mentor feedback, training, checklists and standards.
6.	Determination of correctness of the final software product by a development project with respect to the user needs and requirements.	Demonstration of consistency, completeness, and correctness of the software at each stage and between each stage of the development life cycle.

PART-2

Test Cases, Testing Suites, Test Oracles, Impracticality of Testing All Data, Impracticality of Testing All Paths.

CONCEPT OUTLINE : PART-2

- A test case is a document, which has a set of test data, preconditions, expected results, and post conditions developed for a particular test scenario to verify compliance against a specific requirement.
- Test suite is a container that has a set of tests which help testers in executing and reporting the test execution status.
- The mechanism for determining whether a software program or system has passed or failed such a test is known as test oracle.

Questions-Answers

Long Answer Type and Medium Answer Type Questions

Que 1.14. **What is test case ? List the typical test case parameters.**

Join Telegram Group :- @AKTU_Group Scanned by TapScanner

Join Telegram Group :- @AKTU_Group www.aktutor.in

Answer

1. A test case is a document, which has a set of test data, preconditions, expected results and post conditions, developed for a particular test scenario in order to verify compliance against a specific requirement.
2. Test case acts as the starting point for the test execution, and after applying a set of input values; the application has a definitive outcome and leaves the system at some end point or also known as execution post condition.
3. While drafting a test case do include the following information :
   a. **Test steps :** List all test execution steps in detail. Write test steps in the order in which these should be executed. Make sure to provide as much details as you can. Tip - to efficiently manage test case with lesser number of fields use this field to describe test conditions, test data and user roles for running test.
   b. **Test data :** Use of test data as an input for this test case. You can provide different data sets with exact values to be used as an input.
   c. **Expected result :** What should be the system output after test execution ? Describe the expected result in detail including message/ error that should be displayed on screen.
   d. **Post-condition :** What should be the state of the system after executing this test case ?
   e. **Actual result :** Actual test result should be filled after test execution. It describes system behaviour after test execution.
   f. **Status (Pass/Fail) :** If actual result is not as per the expected result, mark this test as failed. Otherwise, update as passed.
   g. **Notes/Comments/Questions :** To support above fields if there are some special conditions which can't be described in any of the fields or there are questions related to expected or actual results mention those here.

Typical test case parameters :

1. Test case ID
2. Test scenario
3. Test case description
4. Test steps prerequisite
5. Test data
6. Expected result
7. Test parameters
8. Actual result
9. Environment information
10. Comments

Que 1.15. **How to write best test cases ?**

Answer

Best practice for writing good test cases are :

1. **Test cases need to be simple and transparent :**
   a. The description of what requirement is being tested.
   b. The explanation of how the system will be tested.
   c. The test setup like version of application under test, software, data files, operating system, hardware, security access, physical or logical date, time of day, prerequisites such as other tests and any other setup information pertinent to the requirements being tested.
   d. Inputs and outputs or actions and expected results.
   e. Any proofs or attachments use active case language.
   f. Automated script is commented with inputs, purpose and expected results.
   g. Setup offers alternative to prerequisite tests.
2. **Create test case with end user in mind :** Ultimate goal of any software project is to create test cases that meets customer requirements and is easy to use and operate. A tester must create test cases keeping in mind the end user perspective.
3. **Avoid test case repetition :** Do not repeat test cases. If a test case is needed for executing some other test case, call the test case by its test case ID in the precondition column.
4. **Do not assume :** Do not assume functionality and features of your software application while preparing test case. Stick to the specification documents.
5. **Ensure 100% coverage :** Make sure you write test cases to check all software requirements mentioned in the specification document. Use traceability matrix to ensure that no functions/conditions is left untested.
6. **Test cases must be identifiable :** Name the test case ID such that they are identified easily while tracking defects or identifying a software requirement at a later stage.
7. **Implement testing techniques :** It is not possible to check every possible condition in your software application. Testing techniques help you select a few test cases with the maximum possibility of finding a defect :
   a. **Boundary Value Analysis (BVA) :** As the name suggests, it is the technique that defines the testing of boundaries for specified range of values.
   b. **Equivalence Partition (EP) :** This technique partitions the range into equal parts/groups that tend to have the same behaviour.

Join Telegram Group :- @AKTU_Group Scanned by TapScanner

Join Telegram Group :- @AKTU_Group www.aktutor.in

c. **State transition technique :** This method is used when software behaviour changes from one state to another following particular action.

d. **Error guessing technique :** This is guessing/anticipating the error that may arise while testing. This is not a formal method and takes advantages of a tester's experience with the application.

8. **Self cleaning :** The test case you create must return the test environment to the pre-test state and should not render the test environment unusable. This is especially true for configuration testing.

9. **Repeatable and self-standing :** The test case should generate the same results every time, no matter who tests it.

10. **Peer review :** After creating test cases, get them reviewed by your colleagues. Your peers can uncover defects, test case design, which you may easily miss.

Que 1.16. **Write standard fields of sample test case.**

Answer

Standard fields of sample test case template :

1. **Test case ID :** Unique ID for each test case. Follow some convention to indicate types of test. For example, 'TC_UI_1' indicating 'user interfaces test case #1'.
2. **Test priority (low/medium/high) :** This is useful while test execution. Test priority for business rules and functional test cases can be medium or higher whereas minor user interface cases can be low priority. Test priority should be set by reviewer.
3. **Module name :** Mention name of main module or sub module.
4. **Test designed by :** Name of tester.
5. **Test designed date :** Date when wrote.
6. **Test executed by :** Name of tester who executed this test. To be filled after test execution.
7. **Test execution date :** Date when test executed.
8. **Test title/name :** Test case title. For example, verify login page with valid username and password.
9. **Test summary/description :** It describe test objective in brief.
10. **Precondition :** Any prerequisite that must be fulfilled before execution of this test case. List all preconditions in order to successfully execute this test case.
11. **Dependencies :** Mention any dependencies on other test cases or test requirement.

Que 1.17. **What is test suite or validation suite ?**

OR

Write the difference between test case, test suite & test oracle.

OR

Explain the following :

i. **Test suite**

ii. **Test oracle**

Answer

Test suite :

1. In software development, a test suite, less commonly known as a 'validation suite', is a collection of test cases that are intended to be used to test a software program to show that it has some specified set of behaviours.
2. A test suite often contains detailed instructions or goals for each collection of test cases and information on the system configuration to be used during testing.
3. A group of test cases may also contain prerequisite states or steps, and descriptions of the following tests.
4. Test suite is a container that has a set of tests which helps testers in executing and reporting the test execution status.
5. It can take any of the three states namely active, in progress and completed.
6. A test case can be added to multiple test suites and test plans.
7. After creating a test plan, test suites are created which in turn can have any number of tests.
8. Test suites are created based on the cycle or based on the scope.
9. It can contain any type of tests, functional or non-functional.

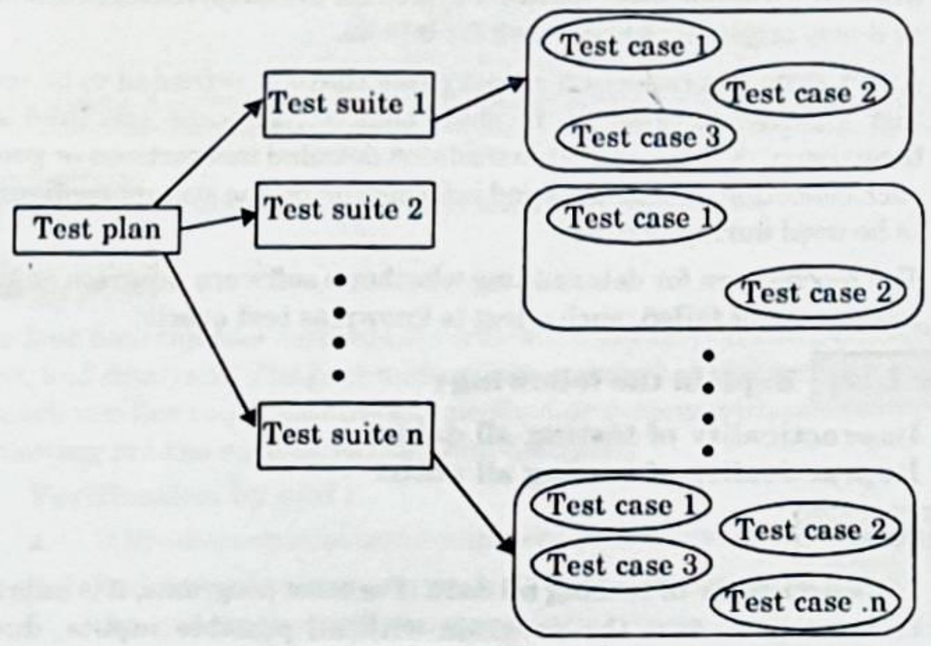


Fig. 1.17.1. Test suite.

Join Telegram Group :- @AKTU_Group Scanned by TapScanner

Join Telegram Group :- @AKTU_Group www.aktutor.in

Test oracle :

1. An oracle is a mechanism for determining whether the program has passed or failed a test.
2. A complete oracle would have three capabilities and would carry them out perfectly :
   a. A generator, to provide predicted or expected results for each test.
   b. A comparator, to compare predicted and obtained results.
   c. An evaluator, to determine whether the comparison results are sufficiently close to be a pass.
3. Common oracles include :
   a. Specifications and documentation.
   b. Other products (for example, an oracle for a software program might be a second program that uses a different algorithm to evaluate the same mathematical expression as the product under test).
   c. A heuristic oracle that provides approximate results or exact results for a set of a few test inputs.
   d. A statistical oracle that uses statistical characteristics.
   e. A consistency oracle that compares the results of one test execution to another for similarity.
   f. A model-based oracle that uses the same model to generate and verify system behaviour.
   g. A human oracle (*i.e.*, the correctness of the system under test is determined by manual analysis).

Difference between test case, test suite and test oracle :

1. A test case is a set of conditions under which a tester will determine whether an application, software system or one of its features is working as it was originally established for it to do.
2. A test suite is a collection of test cases that are intended to be used to test a software program to show that it has some specified set of behaviours. A test suite often contains detailed instructions or goals for each collection of test cases and information on the system configuration to be used during testing.
3. The mechanism for determining whether a software program or system has passed or failed, such a test is known as test oracle.

Que 1.18. **Explain the following :**

a. Impracticality of testing all data
b. Impracticality of testing all paths

Answer

a. **Impracticality of testing all data :** For most programs, it is impractical to attempt to test the program with all possible inputs, due to a combinational explosion. For those inputs selected, a testing oracle is

needed to determine the correctness of the output for a particular test input.

b. **Impracticality of testing all paths :** For most programs, it is impractical to attempt to test all execution paths through the project due to a combinational explosion. It is also not possible to develop an algorithm for generating test data for paths in an arbitrary product, due to the mobility to determine path feasibility.

PART-3

Verification Methods, SRS Verification, Source Code Review, User Documentation Verification, Software Project Audit, Tailoring Software Quality Assurance Programme by Review, Walkthrough, Inspection and Configuration Audits.

CONCEPT OUTLINE : PART-3

- SRS is a detailed description of a software system to be developed with its functional and non-functional requirements.
- Code review is systematic examination of computer source code.
- Walkthrough is a method of conducting informal group/individual review.
- An inspection is a formal, rigorous, in-depth group review designed to identify problems as close to their point of origin as possible.
- Configuration audits provide a mechanism for determining the degree to which the current state of the system is consistent with the latest baseline and documentation.

Questions-Answers

Long Answer Type and Medium Answer Type Questions

Que 1.19. **Discuss various verification methods.**

Answer

The four fundamental methods of verification are Inspection, Demonstration, Test, and Analysis. The four methods are somewhat hierarchical in nature, as each verifies requirements of a product or system with increasing rigor. Following are the various verification methods :

1. **Verification by test :**
   a. It involves special test equipment and/or instrumentation.
   b. It also involves running multiple "tests" (activities) to collect data on the system that will show proof the requirement has been met.

Join Telegram Group :- @AKTU_Group Scanned by TapScanner

Join Telegram Group :- @AKTU_Group www.aktutor.in

c. Because there are multiple sets of data collected, we then have to analyze (activity) the data to make a determination whether or not the verification success criteria has been met to say the verification by Test (method) was successful.

d. Prior to running these tests (activities) we may inspect (activity) the verification (method) results of the children requirements to this requirement.

e. In this case, the method of verification is Test.

2. **Verification by demonstration :**

a. It involves running at least one "test" (activity) (usually more than once) without special test equipment or instrumentation to collect the sets of data that we analyze (activity) to prove the system meets a requirement.

b. Again, prior to running these tests (activities) we may inspect (activity) the verification (method) results of the children requirements to this requirement.

c. In this case, the method of verification is demonstration.

3. **Verification by analysis :**

a. It is used when one of the other methods is not appropriate or you can't afford to use verification by test (method) for every requirement or you can't do an end-to-end test (activity) but only on parts of the system.

b. In this case you collect data on parts of the system via test (activity).

c. Then based on this data and your knowledge of the system design, you make an engineering judgment whether or not the defined success criteria has been met in order to conclude (prove) that verification by analysis (method) was successful.

d. In this case, even though I may have run some tests (activities) to collect the data, the method of verification is analysis.

4. **Verification by inspection :**

a. It is when you can use one of your senses to prove if the system meets a requirement.

b. You don't have to exercise the system, you can observe the results of your observations or inspections (activities) to prove the system meets a requirement.

c. The word inspection is often misused (activity vs method) when there is a need to examine paperwork from other verification activities as a prerequisite to completing one of the other verification methods.

Example :

1. I may be verifying the system meets a requirement that involves the interaction of two of the subsystems.

2. Before running a test (activity) as part of verification by test (method), I may want to examine the verification paperwork for each of the subsystem children requirements to make sure each subsystem was proven to meet its respective interface requirement prior to integrating the two subsystems together and then verifying that the integrated system meets the parent requirement.

3. In this case, even though I inspected (examined) (activity) the lower level verification paperwork, the verification method is test (or one of the other methods of verification).

Que 1.20. **Define SRS. What are the key points of SRS verification ?**

Answer

SRS :

1. SRS is a detailed description of a software system to be developed with its functional and non-functional requirements.
2. The SRS is developed based on the agreement between customer and contractors.
3. It may include the use cases of how user is going to interact with software system.
4. The software requirement specification document consistent of all necessary requirements required for project development.
5. To develop the software system we should have clear understanding of software system.
6. To achieve this we need a continuous communication with customers to gather all requirements.
7. A good SRS defines how software system will interact with all internal modules, hardware, communication with other programs and human user interactions with wide range of real life scenarios.
8. Using the software requirements specification (SRS) document on QA lead, managers creates test plan.

Key points of SRS verification are :

1. **Correctness of SRS should be checked :**

a. Since the whole testing phase is dependent on SRS, it is very important to check its correctness.

b. There are some standards with which we can compare and verify.

2. **Ambiguity should be avoided :**

a. Sometimes in SRS, some words have more than one meaning and this might confuse testers making it difficult to get the exact reference.

b. It is advisable to check for such ambiguous words and make the meaning clear for better understanding.

Join Telegram Group :- @AKTU_Group Scanned by TapScanner

Join Telegram Group :- @AKTU_Group www.aktutor.in

3. **Requirements should be complete :**
   a. When tester writes test cases, what exactly is required from the application, is the first thing which needs to be clear.
   b. For example, if application needs to send the specific data of some specific size then it should be clearly mentioned in SRS that how much data and what is the size limit to send.
4. **Consistent requirements :**
   a. The SRS should be consistent within itself and consistent to its reference documents.
   b. If you call an input "Start and Stop" in one place, don't call it "Start/Stop" in another.
   c. This sets the standard and should be followed throughout the testing phase.
5. **Verification of expected result :** SRS should not have statements like "Work as expected", it should be clearly stated that what is expected since different testers would have different thinking aspects and may draw different results from this statement.
6. **Testing environment :**
   a. Some applications need specific conditions to test and also a particular environment for accurate result.
   b. SRS should have clear documentation on what type of environment is needed to set up.
7. **Pre-conditions defined clearly :**
   a. One of the most important parts of test cases is pre-conditions.
   b. If they are not met properly then actual result will always be different expected result.
   c. Verify that in SRS, all the pre-conditions are mentioned clearly.
8. **Requirements ID :**
   a. These are the base of test case template. Based on requirement ids, test case ids are written.
   b. Also, requirements ids make it easy to categorize modules so just by looking at them, tester will know which module to refer.
   c. SRS must have them such as id defines a particular module.
9. **Security and performance criteria :**
   a. Security is priority when software is tested especially when it is built in such a way that it contains some crucial information when leaked can cause harm to business.
   b. Tester should check that all the security related requirements are properly defined and are clear to him.

   c. Also, when we talk about performance of a software, it plays a very important role in business so all the requirements related to performance must be clear to the tester and he must also know when and how much stress or load testing should be done to test the performance.
10. **Assumption should be avoided :**
    a. Sometimes when requirement is not cleared to tester, he tends to make some assumptions related to it, which is not a right way to do testing as assumptions could go wrong and hence, test results may vary.
    b. It is better to avoid assumptions and ask clients about all the "missing requirements" to have a better understanding of expected results.
11. **Deletion of irrelevant requirements :**
    a. There are more than one team who work on SRS so it might be possible that some irrelevant requirements are included in SRS.
    b. Based on the understanding of the software, tester can find out which are these requirements and remove them to avoid confusions and reduce work load.
12. **Freeze requirements :**
    a. When an ambiguous or incomplete requirement is sent to client to analyze and tester gets a reply, that requirement result will be updated in the next SRS version and client will freeze that requirement.
    b. Freezing here means that result will not change again until and unless some major addition or modification is introduced in the software.

Que 1.21. **What is source code review ? Why it is important ?**

OR

Explain formal code review and lightweight code review.

Answer

Code review :

1. Code review is a phase in the software development process in which the authors of code, peer reviewers, and perhaps quality assurance (QA) testers get together to review code.
2. Finding and correcting errors at this stage is relatively inexpensive and tends to reduce the more expensive process of handling, locating, anti-fixing bugs during later stages of development or after programs are delivered to users.
3. Code review is systematic examination (sometimes referred to as peer review) of computer source code.

Join Telegram Group :- @AKTU_Group Scanned by TapScanner

Join Telegram Group :- @AKTU_Group www.aktutor.in

4. It is intended to find mistakes overlooked in the initial development phase, improving the overall quality of software.
5. Code reviews can often find and remove common vulnerabilities such as format string exploits, race conditions, memory leaks and buffer overflows.
6. Reviewers read the code line by line to check for :
   a. Flaws or potential flaws.
   b. Consistency with the overall program design.
   c. The quality of comments.
   d. Adherence to coding standards.
7. Code review may be especially productive for identifying security vulnerabilities.
8. Automated code reviewing facilitates systematic testing of source code for potential trouble such as buffer overflows, race conditions, memory leakage, size violations, and duplicate statements.
9. Code review is also commonly done to test the quality of patches.

Types of code review :

1. **Formal code review :**
   a. Just like a Fagan inspection, it involves a careful and detailed process with multiple participants and multiple phases.
   b. Formal code reviews are the traditional method of review, in which software developers attend a series of meetings and review code line by line, usually using printed copies of the material.
   c. Formal inspections are extremely thorough and have been proven effective at finding defects in the code under review.
2. **Lightweight code review :**
   a. Typically requires less overhead than formal code inspections, though it can be equally effective when done properly.
   b. Lightweight reviews are often conducted as part of the normal development process :
      i. **Over-the-shoulder :** One developer looks over the author's shoulder as the latter walks through the code.
      ii. **Email pass-around :** Source code management system emails code to reviewers automatically after check is made.
      iii. **Pair programming :** Two authors develop code together at the same workstation, as it is common in extreme programming.
      iv. **Tool-assisted code review :** Authors and reviewers use software tools, informal ones such as pastebins and IRC, or specialized tools designed for peer code review.

Que 1.22. **What is user documentation verification ?**

Answer

1. Any written or pictorial information describing, defining, specifying, reporting, or certifying activities, requirements, procedures, or results.
2. Documentation is as important to a product's success as the product itself.
3. If the documentation is poor, non-existent, or wrong, it reflects on the quality of the product and the vendor.
4. As per the IEEE documentation describing plans for, or results of the testing of a system or component, types include test case specification, test incident report, test log, test plan, test procedure, test report.
5. Hence, the testing of all the above mentioned documents is known as documentation testing.
6. This is one of the most cost effective approaches to testing.
7. If the documentation is not right, there will be major and costly problems.
8. The documentation can be tested in a number of different ways to many different degrees of complexity.
9. These range from running the documents through a spelling and grammar checking device to manually reviewing the documentation to remove any ambiguity or inconsistency.
10. Documentation testing can start at the very beginning of the software process and hence save large amounts of money, since earlier a defect is found, the less it will cost to be fixed.
11. Documentation testing means verifying the technical accuracy and readability of the user manuals, including the tutorials and the online help.
12. Documentation testing is performed at three levels as explained :
    a. **Read test :** In this test, documentation is reviewed for clarity, organization, flow, and accuracy without executing the documented instructions on the system.
    b. **Hands-on test :** The online help is exercised and the error messages are verified to evaluate their accuracy and usefulness.
    c. **Functional test :** The instructions embodied in the documentation are followed to verify that the system works as it has been documented.

Following concrete tests are recommended for documentation testing :

1. Read all the documentations to verify :
   a. Correct use of grammar.

Join Telegram Group :- @AKTU_Group Scanned by TapScanner

Join Telegram Group :- @AKTU_Group www.aktutor.in

b. Consistent use of the terminology.

c. Appropriate use of graphics where possible.

2. Verify that the glossary accompanying the documentation uses a standard, commonly accepted terminology and that the glossary correctly defines the terms.

3. Verify that there exists an index for each of the documents and the index block is reasonably rich and complete. Verify that the index section points to the correct pages.

4. Verify that there is no internal inconsistency within the documentation.

5. Verify that the online and printed versions of the documentation are same.

6. Verify the installation procedure by executing the steps described in the manual in a real environment.

7. Verify the troubleshooting guide by inserting error and then using the guide to troubleshoot the error.

8. Verify the software release notes to ensure that these accurately describe the changes in features and functionalities between the current release and the previous ones and the set of known defects and their impact on the customer.

9. Verify the online help for its :

   a. Usability

   b. Integrity

   c. Usefulness of the hyperlinks and cross-references to related topics

   d. Effectiveness of table look-up

   e. Accuracy and usefulness of indices

10. Verify the configuration section of the user guide by configuring the system as described in the documentation.

11. Finally, use the document while executing the system test cases. Walkthrough the planned or existing user work activities and procedures using the documentation to ensure that the documentation is consistent with the user work.

Que 1.23. **What is the objective of software project audit ?**

Answer

A software audit review, or software audit, is a type of software review in which one or more auditors who are not members of the software development organization conduct an independent examination of a software product, software process, or set of software processes to assess compliance with specifications, standards, contractual agreements, or other criteria.

Objectives of software project audit : The purpose of a software audit is to provide an independent evaluation of conformance of software products and processes to applicable regulations, standards, guidelines, plans, and procedures.

The following roles are recommended :

1. **The initiator** (who might be a manager in the audited organization, a customer or user representative of the audited organization, or a third party), decides upon the need for an audit, establishes its purpose and scope, specifies the evaluation criteria, identifies the audit personnel, decides what follow-up actions will be required, and distributes the audit report.

2. **The lead auditor** (who must be someone "free from bias and influence that could reduce his ability to make independent, objective evaluations") is responsible for administrative tasks such as preparing the audit plan and assembling and managing the audit team, and for ensuring that the audit meets its objectives.

3. **The recorder** documents anomalies, action items, decisions, and recommendations made by the audit team.

4. **The auditors** (who must be like the lead auditor, free from bias) examine products defined in the audit plan, document their observations, and recommend corrective actions (there may be only a single auditor.)

5. **The audited organization** provides a liaison to the auditors, and provides all information requested by the auditors. When the audit is completed, the audited organization should implement corrective actions and recommendations.

Principles of software audit :

1. **Timeliness :** Only when the processes and programming is continuous inspected in regard to their potential susceptibility to faults and weaknesses, but as well with regard to the continuation of the analysis of the found strengths, or by comparative functional analysis with similar applications, an updated frame can be continued.

2. **Source openness :** It requires an explicit reference in the audit of encrypted programs, how the handling of open source has to be understood.

3. **Elaborateness :** Audit processes should be oriented to certain minimum standard.

4. **The financial context :** Further, transparency is needed to clarify whether the software has been developed commercially and whether the audit was funded commercially (paid audit). It makes a difference whether it is a private hobby / community project or whether a commercial company is behind it.

5. **Scientific referencing of learning perspectives :** Each audit should describe the findings in detail within the context and also highlight progress and development needs constructively.

Join Telegram Group :- @AKTU_Group Scanned by TapScanner

Join Telegram Group :- @AKTU_Group
www.aktutor.in

6. **Literature-inclusion :** A list of references should be accompanied in each case of an audit.
7. **Inclusion of user manuals and documentation :** Further, a check should be done, whether there are manuals and technical documentations, and, if these are expanded.
8. **Identify references to innovations :** Applications that allow both messaging to offline and online contacts, so considering chat and e-mail in one application as it is also the case with GoldBug-should be tested with high priority (criterion of presence chats in addition to the e-mail function). The auditor should also highlight the references to innovations and underpin further research and development needs.

Que 1.24. **Explain the following in terms of software testing :**

a. **Tailoring**
b. **Tailoring software quality assurance program by reviews**

Answer

a. **Tailoring :**
   i. Tailoring is the responsibility of the development team in consultation with the process champion or software quality assurance who actually owns the development process.
   ii. Purpose of tailoring is to eliminate tasks that add unnecessary costs, cycle time and data that do not add value to the process or the definition of the product.
   iii. Tailoring of a standard takes the form of deletion of wasteful steps, alteration of activities to more explicitly reflect the application to a particular context, or addition of activities to satisfy program requirements.
   iv. Sound engineering judgment is necessary to ensure that tailoring does not contribute any additional risks.

Objective of tailoring :

1. To document the necessary preliminary activities that shall be executed before calling for a review.
2. To provide the review process mechanism.
3. To provide details on the review meeting.
4. To document the activities that shall govern the review disposition.
5. To provide the post-review meeting activities.
6. Templates for facilitating the review process.

Purpose of tailoring :

1. To conduct the business specification reviews, technical specification reviews and test readiness reviews using a structured approach.

2. To document a formal, well-defined process to conduct reviews of all deliverables/policies baselines at the site.
3. To establish a quality checkpoint to ensure the creation of defect-free deliverables using systematic screening mechanisms.
4. To encourage the development process to embark on an internal process improvement initiative by continuously improving the review process mechanism itself.

Preliminary activities :

1. The document shall be correct, complete and consistent in all respects before the author calls for a review.
2. The author and a review leader/review facilitator shall conduct such a preliminary investigation.
3. Any information, which the author does not already possess to make the deliverable correct, complete and consistent, shall be solicited by the author from people competent to provide such information. This shall happen much before actually documenting the deliverable and would ideally happen at the point of entry to each phase.
4. The author shall choose a review leader/review facilitator well in advance and appraise this person about the nature of review and provide the person the document/deliverable that will be reviewed using which a baseline will be formed.
5. If the review leader/review facilitator decides that the document is fit for a review, then this person will put together a review panel that will comprise of at least 4 people but not more than 9, who will review the document/deliverable keeping their perspective in mind.
6. If the review leader/review facilitator decides that the document is NOT fit for a review, then this person will suggest that the document/deliverable be modified, re-written by the author before a formal review is conducted. This decision will have to be communicated to all concerned people. For example, project manager.
7. The review panel put together to conduct the review shall comprise of people having a good exposure to the domain, that is, being captured in the document/deliverable and in the whole review process mechanism itself.
8. After conducting this readiness check, a representative from the software quality assurance (SQA) department shall authorize the conduct of the review. The review leader/review facilitator shall appraise the concerned SQA representative on the worthiness of the document/deliverable for a review.
9. The SQA representative shall then allow the review leader/review facilitator to distribute the deliverable/document in a formal manner.
10. The document/deliverable, which shall be reviewed, will be in the possession of the review leader/review facilitator and no requests from

Join Telegram Group :- @AKTU_Group Scanned by TapScanner

Join Telegram Group :- @AKTU_Group www.aktutor.in

the author to modify or change or append new material to the review material shall be entertained.

11. The review leader/review facilitator shall conduct a preliminary estimate on the time required for completely reviewing the document and make sure that the estimated time is made available for the review panel.
12. The review leader/review facilitator shall then raise a duly completed review meeting notice form and distribute the review material.

b. **Tailoring software quality assurance program by reviews :**

i. **Continuous improvement :** All the standard process in SQA must be improved frequently and made official so that the other can follow. This process should be certified by popular organization such as ISO, CMMI etc.

ii. **Documentation :** All the QA policies and methods, which are defined by QA team, should be documented for training and reuse for future projects.

iii. **Experience :** Choosing the members who are seasoned SQA auditors is a good way to ensure the quality of management review.

iv. **Tool usage :** Utilizing tool such as the tracking tool, management tool for SQA process reduces SQA effort and project cost.

v. **Metrics :** Developing and creating metrics to track the software quality in its current state, as well as to compare the improvement with previous versions, will help in increasing the value and maturity of the testing process.

vi. **Responsibility :** The SQA process is not the SQA member's task, but everyone's task. Everybody in the team is responsible for quality of product, not just the test lead or manager.

Que 1.25. **What is walkthrough ? What are the goals of walkthrough ?**

Answer

1. In a walkthrough, author describes and explains the work product in an informal meeting to his peers or supervisor to get feedback.
2. Here, validity of the proposed solution for work product is checked.
3. Structured walkthrough technique is very useful technique to analyze a product for its effectiveness.
4. In design phase of the product, the purpose of walkthrough is to find out as many possible problems in product design while the design is on paper.
5. It is cheaper to make changes when the design is on paper rather than at the time of conversion.

6. Generally, walkthrough can be done at any stage of product development as given :
   a. At the time of deciding schedule for different phases.
   b. At the time of problem specification.
   c. Designing data structure.
   d. Program designing.
   e. Preparing documentation and user manual.
   f. Coding.
   g. Test plan, data and result.
   h. Maintenance changes.
7. So, the walkthrough can start in early stage of software development as design planning, long before the testing begins.
8. It is a static method of quality assurance. Walkthrough are informal meetings but with purpose.

Objectives of walkthrough :

1. It is one of the methods of review whose purpose is to ensure high quality.
2. Its main objectives are to find :
   a. Bugs, misinterpretation, errors, inconsistencies, and anything that is unclear.
   b. Anything that is complex and difficult to modify.
   c. Any deviation from standard.
3. The purpose of walkthrough is to only find out the problem not to correct them, the correction is the field of developer.
4. Different authorities give different recommendation for optimum number of participant in walkthrough.
5. Generally, four participants are taken for walkthrough. This is because the more people there are, the more passivity of waste of time due to difference of options.
6. Now who will attend the walkthrough ? Again there is difference in opinion but generally this is decided on the basis of work product being walkthrough.
7. There is general agreement that author of the product, a maintenance expert and a member of quality assurance group should attend the walkthrough.
8. Users can attend specification walkthrough, testing walkthrough, design walkthrough but not the code walkthrough.
9. User involvement in walkthrough may help system developer.

Join Telegram Group :- @AKTU_Group Scanned by TapScanner

Join Telegram Group :- @AKTU_Group www.aktutor.in

Que 1.26. **What is the objectives of inspection ? Write the goals of inspection ?**

OR

What is inspection ? List and explain the stages of the inspection process.

Answer

1. Inspections improve the reliability, availability, and maintainability of a software product.
2. Anything readable that is produced during software development can be inspected.
3. Inspections can be combined with structured, systematic testing to provide a powerful tool for creating defect-free programs.
4. The inspection activity follows a specified process and the participants play well-defined roles.
5. An inspection team consists of three to eight members who play the roles of moderator, author, reader, recorder, and inspector.
6. It also helps to have a client representative participate in requirements specification inspections.
7. Group inspections enable team members to exchange knowledge and ideas during an inspection session.
8. Moderator leads the inspection, schedules meetings, controls meetings, reports inspection results, and follows up on rework issues.
9. Author creates or maintains the work product being inspected.
10. Reader describes the sections of the work product to the team as they proceed through inspection.
11. Recorder classifies and records defects and issues raised during the inspection.
12. All participants play the role of inspectors. However, good inspectors are those who have created the specification for the work product being inspected.
13. For example, the designer can act as an inspector during code inspection while a quality assurance representative can act as standard enforcer.

An error checklist for inspections :

1. An important part of the inspection process is the use of a checklist to examine the program for common errors.

2. The checklist is largely language independent, meaning that most of the errors can occur with any programming language.
3. We may wish to supplement this list with errors peculiar to our programming language.
4. The errors may be :
   a. Data reference errors
   b. Data-declaration errors
   c. Computation errors
   d. Comparison errors
   e. Control-flow errors
   f. Interface errors
   g. Input-output errors

Goals of inspection :

1. It helps the author to improve the quality of the document under inspection.
2. It removes defects efficiently and as early as possible.
3. It improves product quality.
4. It creates common understanding by exchanging information.
5. It learn from defects found and prevent the occurrence of similar defects.

Stages in the inspections process :

1. **Planning :** The inspection is planned by the moderator.
2. **Overview meeting :** The author describes the background of the work product.
3. **Preparation :** Each inspector examines the work product to identify possible defects.
4. **Inspection meeting :** During this meeting, the reader reads through the work product, part by part and the inspectors point out the defects for every part.
5. **Rework :** The author makes changes to the work product according to the action plans from the inspection meeting.
6. **Follow-up :** The changes by the author are checked to make sure everything is correct.

Que 1.27. **Write the difference between inspection and walkthrough.**

Join Telegram Group :- @AKTU_Group Scanned by TapScanner

Join Telegram Group :- @AKTU_Group www.aktutor.in

Answer

Difference between inspection and walkthrough :

S. No.	Inspection	Walkthrough
1.	It is formal.	It is informal.
2.	Initiated by the project team.	Initiated by the author.
3.	A group of relevant persons from different departments participate in the inspection.	Usually team members of the same project take participation in the walkthrough. Author himself acts the walkthrough leader.
4.	Checklist is used to find faults.	No checklist is used in walkthroughs.
5.	Inspection processes includes overview, preparation, inspection, and rework and follow up.	Walkthrough process includes overview, little or no preparation, examination (actual walkthrough meeting), and rework and follow up.
6.	Formalized procedure in each step.	No formalized procedure in the steps.
7.	Inspection takes longer time as the list of items in the checklist is tracked to completion.	Shorter time is spent on walkthroughs as there is no formal checklist used to evaluate the program.
8.	Planned meeting with fixed roles assigned to all the members involved.	Unplanned.
9.	Reader reads the product code. Everyone inspects it and comes up with defects.	Author reads the product code and his team mate comes up with defects or suggestions.
10.	Recorder records the defects.	Author makes a note of defects and suggestions offered by team mate.
11.	Moderator has a role in making sure that the discussions proceed on the productive lines.	Informal, so there is no moderator.

Que 1.28. **Why configuration audits are required ? Explain its types and process of each.**

OR

Explain functional and physical configuration audit.

Answer

Configuration audits :

1. Configuration audits provide a mechanism for determining the degree to which the current state of the system is consistent with the latest baseline and documentation.
2. They provide greater visibility into the status of a project by evaluating the status of the items.
3. Configuration audits provide such a review.
4. They are not the same as quality audits or product tests.
5. However, they use the information available as an outcome of the quality audits and testing along with the configuration status accounting information, to provide assurance that what was required has been build.
6. Configuration audits are typically performed at the time of delivery and major upgrades to the software.
7. Configuration audits are conducted at the end of each life cycle phase.
8. They verify that :
   a. All required configuration items have been produced.
   b. All configuration items produced comply with the specified requirements.
   c. Technical documentation completely and accurately describes the configuration items.
   d. The configuration item register accurately describes designated baselines.
   e. All approved change requests have been resolved.
   f. At the completion of development, the software or systems product is ready for delivery.
9. Configuration audits may be conducted by the software quality assurance, the configuration management or the verification and validation functions.

There are two types of configuration audits :

1. **Functional configuration audit (FCA) :**
   a. The objective of the functional audit is to provide an independent evaluation of a software product, verifying that its configuration items actual functionality and performance is consistent with the relevant requirement specification.

Join Telegram Group :- @AKTU_Group Scanned by TapScanner

Join Telegram Group :- @AKTU_Group www.aktutor.in

b. This audit is held prior to software delivery to verify that all requirements specified in the software requirements specification have been met.

Inputs (data) for FCA :

1. Primary inputs for the FCA are the functional requirements for the device and test or operational data showing how the device operates.
2. Functional requirement information should include verification method (test, demonstration, analysis, etc.) and the test method used (if applicable).
3. FCAs may use but need not be limited to data from the following processes and tests :
   a. Environmental testing
   b. Reliability, availability and maintainability tests and trials
   c. User trials
   d. Interface checks and tests
   e. Software testing including independent verification and validation if safety critical software is involved.

FCA process :

Step 1 : If not already provided, construct a matrix (spreadsheet) showing the requirements, verification method, and testing procedure name. Ensure that all requirements have a verification method (and procedure) defined.

Step 2 : Add columns to the matrix for test status (pass, fail, outstanding action items). Add columns to record other details of interest such as the date on which the test was conducted and the quality assurance person who witnessed the test. Add a column for information on open action items.

Step 3 : Review the as-run test documentation (or inspection/analysis reports) that are called out as verifications for each requirement. Record the appropriate information in the matrix. When reviewing ensure that the test was adequate to verify the requirement.

Step 4 : Identify any requirements that are open (failed or outstanding action item).

Step 5 : Write a report documenting the audit and findings.

Step 6 : Resolve any findings and other issues with the project.

Output : Functional configuration audit report.

An audit template / checklist are provided to help you conduct your audit. The template covers all assurance levels and development phases. Tailor the template to fit the needs of the audit.

2. Physical configuration audits (PCA) :

a. The objective of the physical audit is to provide an independent evaluation of a software product's configuration items to confirm that all components in the as-built version map to their specifications.

b. Specifically, this audit is held to verify that the software and its documentation are internally consistent.

Inputs (data) for PCA :

FCA report :

1. Complex electronics requirements (specification).
2. Synthesis of log files.
3. Testing and verification reports.
4. Programming process plan.
5. Configuration Management records.
6. Deviations and waivers.
7. Problem reports.

PCA process :

Step 1 : Gather data and documents.

Step 2 : Review of FCA report and verify incorporation (or other appropriate disposition) of action items and findings.

Step 3 : Review the requirements (specification) for the complex electronics to ensure that the requirements are implemented in the design.

Step 4 : Review any log files or other documents from the synthesis process or from developer testing or verification activities. Ensure that no errors were detected by these processes that were not fixed in the device design.

Step 5 : Review the process plan for device programming. Ensure that the plan has been followed, and is programming the chip with the correct code. Review configuration management records to ensure that the correct design (code) was used.

Step 6 : Review problem reports, deviations, and waivers to ensure that there are no open issues with the complex electronics design or device.

Step 7 : Write the PCA report documenting the process and findings of the audit.

Step 8 : Resolve any issues and findings with the project.

Output : Physical configuration audit report.

An audit template / checklist are provided to help you conduct your audit. The template covers all assurance levels and development phases. Tailor the template to fit the needs of the audit.

☺☺☺

Join Telegram Group :- @AKTU_Group Scanned by TapScanner

Join Telegram Group :- @AKTU_Group www.aktutor.in

2 UNIT

Functional Testing and Structural Testing

Part-1 ..... (55E - 67E)

- *Functional Testing : Boundary Value Analysis*
- *Equivalence Class Testing*
- *Decision Table Based Testing*
- *Cause-Effect Graphing Technique*

A. Concept Outline : Part-1 ..... 55E
B. Long and Medium Answer Type Questions 55E

Part-2 ..... (67E - 88E)

- *Structural Testing : Control Flow Testing*
- *Path Testing*
- *Independent Paths*
- *Generation of Graph From Program*
- *Identification of Independent Paths*
- *Cyclomatic Complexity*
- *Data Flow Testing*
- *Mutation Testing*

A. Concept Outline : Part-2 ..... 67E
B. Long and Medium Answer Type Questions 67E

PART-1

Functional Testing : Boundary Value Analysis, Equivalence Class Testing, Decision Table Based Testing, Cause-Effect Graphing Technique.

CONCEPT OUTLINE : PART-1

- Functional testing is a quality assurance process and a type of black box testing.
- Decision tables provide a systematic way of stating complex business rules, which are useful for developers as well as testers.
- A cause-effect graph is a directed graph that maps a set of causes to a set of effects.

Questions-Answers

Long Answer Type and Medium Answer Type Questions

Que 2.1. **What is functional testing ? Write the difference between functional testing and non-functional testing.**

OR

Explain functional testing. List some functional testing technique.

Answer

1. Functional testing is a quality assurance (QA) process and a type of black box testing that bases its test cases on the specifications of the software component under test.
2. Functions are tested by feeding them input and examining the output, and internal program structure is rarely considered (not like in white box testing).
3. Functional testing usually describes what the system does.
4. Functional testing is a testing technique, that is, used to test the features/ functionality of the system or software, should cover all the scenarios including failure paths and boundary classes.
5. Functional testing verifies that each function of the software application operates in conformance with the requirement specification.
6. This testing mainly involves black box testing and it is not concerned about the source code of the application.
7. Each and every functionality of the system is tested by providing appropriate input, verifying the output and comparing the actual results with the expected results.

Join Telegram Group :- @AKTU_Group Scanned by TapScanner

Join Telegram Group :- @AKTU_Group www.aktutor.in

8. This testing involves checking of user interface, APIs, database, security, client/ server applications and functionality of the application under test.
9. The testing can be done either manually or using automation.
10. Functional testing does not imply that you are testing a function (method) of your module or class.
11. Functional testing tests a slice of functionality of the whole system.
12. Functional testing typically involves six steps :
    a. The identification of functions that the software is expected to perform.
    b. The creation of input data based on the function's specifications.
    c. The determination of output based on the function's specifications.
    d. The execution of the test case.
    e. The comparison of actual and expected outputs.
    f. To check whether the application works as per the customer need.

Functional v/s non-functional testing

S. No.	Functional testing	Non-functional testing
1.	Functional testing is performed using the functional specification provided by the client and verifies the system against the functional requirements.	Non-functional testing checks the performance, reliability, scalability and other non-functional aspects of the software system.
2.	Functional testing is executed first.	Non-functional testing should be performed after functional testing.
3.	Manual testing or automation tools can be used for functional testing.	Using tools will be effective for this testing.
4.	Business requirements are the inputs to functional testing.	Performance parameters like speed, scalability are inputs to non-functional testing.
5.	Functional testing describes what the product does.	Non-functional testing describes how good the product works.
6.	Easy to do manual testing.	Tough to do manual testing.
7.	Types of functional testing are : a. Unit testing b. Smoke testing c. Sanity testing d. Integration testing e. Black box testing f. User acceptance testing	Types of non-functional testing are : a. Performance testing b. Load testing c. Volume testing d. Stress testing e. Security testing f. Installation testing g. Penetration testing h. Compatibility testing i. Migration testing

Que 2.2. **Describe the boundary value analysis testing.**

OR

Explain boundary value analysis testing.

Answer

Boundary value analysis :

1. Boundary value analysis is a method useful for arriving at tests that are effective in catching defects that happen at boundaries.
2. Boundary value analysis believes and extends the concept that the density of defects is more towards the boundaries.
3. Generally, it has been found that most defects in situations such as this happen around the boundaries.
4. While the reasons for this phenomenon is not entirely clear, some possible reasons are as follows :
    a. Programmers tentativeness in using the right comparison operator, for example : whether to use the < = operator or < operator when trying to make comparisons.
    b. Confusion caused by the availability of multiple ways to implement loops and condition checking.
    c. The requirements themselves may not be clearly understood, especially around the boundaries.
5. Boundary value analysis is extremely useful in uncovering defects when there are internal limits placed on certain resources, variables or data structures.
6. Consider a database management system which caches the recently used data blocks in a shared memory area.
7. Usually such a cached area is limited by a parameter that the user specifies at the time of starting up the system.
8. When these buffers are full and next block needs to be cached, the least recently buffer – the first buffer – needs to be released, after storing it in secondary memory.
9. Both the operations, inserting the new buffer as well as freeing up the first buffer, happen at the "boundaries".

Join Telegram Group :- @AKTU_Group Scanned by TapScanner

Join Telegram Group :- @AKTU_Group www.aktutor.in

To summarize boundary value testing :

1. Look for any kind of gradation or discontinuity in data values which affect computation, the discontinuities are the boundary values, which require thorough testing.
2. Look for any internal limits such as limits on resources. The behaviour of the product at these limits should also be the subject of boundary value testing.
3. It also includes the list of boundary values, documented limits on hardware resources.

   The internal data structures like arrays, stacks, and queues need to be checked for boundary or limit conditions.

Que 2.3. **Write advantages and disadvantages of boundary value analysis.**

Answer

Advantages of boundary value analysis :

1. Small set of test cases is prepared.
2. It reduces the number of test cases for a tester.
3. Very good at exposing potential user interface/user input problems.
4. Easy to test the boundary values.
5. Provide clear guidelines to prepare the test cases.
6. It is used for finding the bugs that occur at boundaries rather than in the center.
7. It is the technique in which the input domain is divided into different class and selects each class for testing so that we can easily find the bugs.

Disadvantages of boundary value analysis :

1. Difficult to prepare the test cases if the boundary values are not known correctly.
2. Does not test dependencies between combinations of inputs.
3. Not useful for large amount of values.
4. Not all possible inputs are tested.

Que 2.4. **Explain equivalence class testing.**

OR

What is the equivalence partitioning technique ?

Answer

Equivalence class testing / Equivalence partitioning technique :

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

2. This reduces the number of permutations and combinations of input, output values used for testing, thereby, increasing the coverage and reducing the effort involved in testing.
3. The set of input values that generate single expected output is called a partition.
4. When the behaviour of the software is the same for a set of values, then the set is termed as an equivalence class or a partition.
5. In this case, one representative sample from each partition (also called the member of the equivalence class) is picked up for testing.
6. One sample from the partition is enough for testing as the result of picking up some more values from the set will be same and will not yield any additional defects.
7. Since all the values produce equal and same output, they are termed as equivalence partition.
8. Testing by this technique involves :
   a. identifying all partitions for the complete set of input, output values for a product and;
   b. picking up one member value from each partition for testing to maximize complete coverage.
9. From the results obtained for a member of an equivalence class or partition, this technique extrapolates the expected results for all the values in that partition.
10. The advantage of using this technique is that we gain good coverage with a small number of test cases.
11. By using this technique, redundancy of tests is minimized by not repeating the same tests for multiple values in the same partition.

The steps to prepare an equivalence partition table are as follows :

1. Choose criteria for doing the equivalence partitioning (range, list of values and so on).
2. Identify the valid equivalence classes based on the above criteria (number of ranges, allowed values).
3. Select a sample data from that partition.
4. Write the expected result based on the requirements given.
5. Identify special values, if any, and include them in a table.
6. Check to have expected results for all the cases prepared.
7. If the expected result is not clear for any particular test case, mark appropriately and escalate for corrective actions.

An example of equivalence classes for the life insurance premium is :

Join Telegram Group :- @AKTU_Group Scanned by TapScanner

Join Telegram Group :- @AKTU_Group www.aktutor.in

S.No.	Equivalence partitions	Type of Input	Test data	Expected results
1.	Age below 35	Valid	26,12	Monthly premium = Rs.(0.50 + 1.65) = Rs. 2.15
2.	Age 35- 59	Valid	37	Monthly premium = Rs.(0.50 + 2.87) = Rs.3.37
3.	Age above 60	Valid	65, 90	Monthly premium = Rs.(0.50 + 6.0) = Rs.6.5
4.	Negative age	Invalid	–23	Warning message – Invalid input
5.	Age as 0	Invalid	0	Warning message –

Que 2.5. **Write the advantages and disadvantages of equivalence class partitioning.**

OR

What are the guidelines used to define equivalence class partitioning ?

Answer

Following guidelines are used to define equivalence class partitioning :

1. For specified range of input condition, select one valid equivalence class that will cover valid range and second will contain invalid range.
2. For specific value as input, select one valid equivalence class which will have that value and other will have other values than valid one.
3. For specific condition, select the set of valid input values then select the one valid equivalence class that contains valid set values and other should contains values other than the valid set values.
4. If the input conditions are broken then select one valid and one invalid value.

Advantages of equivalence class partitioning (ECP) :

1. ECP number of test cases is reduced.
2. Classes helps tester to focus on smaller data sets by which probability to uncover the defects are more.
3. It eliminates the exhaustive testing for entire input domain which is not feasible.
4. Allows covering of large domain of inputs.

Limitations of equivalence class partitioning (ECP) :

1. ECP usually does not consider the boundary conditions.
2. Sometimes issues at boundaries are ignored.
3. Testers tend to assume that result will be correct for all input data sets.

Que 2.6. **What is decision table based testing ?**

Answer

1. A decision table is a good way to deal with combinations of things (for example, inputs).
2. This technique is sometimes also referred to as a 'cause-effect' table.
3. The reason for this is that there is an associated logic diagramming technique called 'cause-effect graphing' which was sometimes used to help in deriving the decision table.
4. It helps in reducing test effort in verifying each and every combination of test data, at the same time ensuring complete coverage.
5. Decision tables provide a systematic way of stating complex business rules, which is useful for developers as well as for testers.
6. Decision tables can be used in test design in order to determine whether or not they are used in specifications or not.
7. A decision table is the method used to build a complete set of test cases without using the internal structure of the program in question.
8. In order to create test cases, we use a table to contain the input and output values of a program.
9. Such a table is split up into four sections as shown in Fig. 2.6.1.

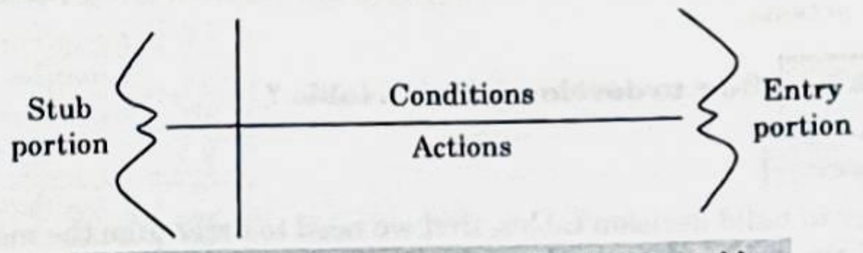


Fig. 2.6.1. The basic structure of a decision table.

10. In Fig. 2.6.1, there are two lines which divide the table into its main structure.
11. The solid vertical line separates the stub and entry portions of the table, and the solid horizontal line is the boundary between the conditions and actions.
12. So, these lines separate the table into four portions : condition stub, action stub, condition entries and action entries.
13. A column in the entry portion of the table is known as a rule.
14. Values which are in the condition entry columns are known as inputs and values inside the action entry portions are known as outputs.
15. Outputs are calculated depending on the inputs and specification of the program.
16. In Fig. 2.6.2, there is an example of a typical decision table.
17. The inputs in this table derive the outputs depending on what conditions these inputs met.

Join Telegram Group :- @AKTU_Group Scanned by TapScanner

Join Telegram Group :- @AKTU_Group www.aktutor.in

18. Notice the use of input in the table below, these are known as don't care entries.
19. Don't care entries are normally viewed as being false values which do not require the value to define the output.

Stub	Rule 1	Rule 2	Rule 3	Rule 4	Rule 5	Rule 6
c1	T	T	T	F	-	T
c2	F	T	T	T	-	T
c3	T	T	-	T	T	T
c4	T	F	F	T	T	
a1	X	X	-	X	X	X
a2	X	X		X	X	X

Fig. 2.6.2. Typical structure of a decision table.

20. Fig. 2.6.2, shows its values from the inputs as true(T) or false(F) values which are binary conditions, tables which use binary conditions are known as limited entry decision tables.
21. Tables which use multiple conditions are known as extended entry decision tables.
22. One important aspect to notice about decision tables is that they aren't imperative as they do not apply any particular order over the conditions or actions.

Que 2.7. **How to develop decision table ?**

Answer

In order to build decision tables, first we need to determine the maximum size of the table, then we have to eliminate any impossible situations, inconsistencies, or redundancies, and finally we need to simplify the table as much as possible.

Following are the ways to develop decision tables :

1. Study the given specification and determine the number of conditions that may affect the decision. Remove any conditions that repeat. Once we arrive at this list of non-repeatable conditions (that is, conditions that are mutually exclusive), we have to record all these conditions as rows in the top left half of the decision table.
2. Determine the number of possible actions that can be taken. These become the number of rows in the lower left half of the decision table.
3. Determine the number of condition alternatives for each condition. In the simplest form of decision table, there would be two alternatives (Y or N) for each condition. In an extended-entry table, there may be many alternatives for each condition.
4. Then, we have to calculate the maximum number of columns in the decision table by calculating number of conditions raise to the number

of alternatives. If there were three conditions and two alternatives (Y or N) for each of the conditions, there would be eight alternatives.

5. Fill in the condition alternatives. Start with the first condition and divide the number of columns by the number of alternatives for that condition. In the foregoing example, there are three columns and two alternatives (Y and N), so eight divided by two is four. Then choose one of the alternatives and write Y in all of the four columns. Finish by writing N in the remaining four columns as follows :

Condition : YYYYNNNN

Repeat this for each condition using a subset of the table :

Y	Y	Y	Y	N	N	N	N
Y	Y	N	N				
Y	N						

and continue the pattern for each condition :

Y	Y	Y	Y	N	N	N	N
Y	Y	N	N	Y	Y	N	N
Y	N	Y	N	Y	N	Y	N

6. Complete the table by inserting a √ where rules suggest certain actions.
7. Combine rules where it is apparent that an alternative does not make a difference in the outcome; for example :

Condition 1	YY
Condition 2	YN
Condition 3	YY
Action 1	√√

Condition 1	Y
Condition 2	Y/N
Condition 3	Y
Action 1	√√

8. Check the table for any impossible situations, contradictions, redundancies.
9. Rearrange the conditions and actions (or even rules) to make the decision table more understandable.

Que 2.8. **Why decision table is important ?**

Answer

1. Decision tables are very much helpful in test design technique.
2. It helps testers to search the effects of combinations of different inputs and other software states that must correctly implement business rules.
3. It also provides a regular way of stating complex business rules, that is, helpful for developers as well as for testers.
4. Testing combinations can be a challenge, as the number of combinations can often be huge.

Join Telegram Group :- @AKTU_Group Scanned by TapScanner

Join Telegram Group :- @AKTU_Group
www.aktutor.in

5. It assists in development process with developer to do a better job.
6. Testing with all combination might be unrealistic or unfeasible.
7. We have to be happy with testing just a small subset of combinations but making the option of which combinations to test and which to leave out is also significant.
8. If you do not have efficient way of selecting combinations, an arbitrary subset will be used and this may result in an ineffective test effort.
9. A decision table is basically an outstanding technique used in both testing and requirements management.
10. It is a structured exercise to prepare requirements when dealing with complex business rules. It is also used in model complicated logic.

Que 2.9. **Write advantages and disadvantages of decision table.**

OR

Write the application of decision table.

Answer

Advantages of decision table :

1. This type of testing works iteratively.
2. These tables guarantee that we consider every possible combination of condition values. This is known as its "completeness property".
3. Decision tables are declarative.

Disadvantages of decision table :

1. Decision tables do not scale up well.
2. We need to "factor" large tables into smaller ones to remove redundancy.

Applications of decision table :

1. Prominent if-then-else logic.
2. Logical relationships among input variables.
3. Calculations involving subsets of the input variables.
4. Cause-and-effect relationships between inputs and outputs.
5. High cyclomatic complexity.

Que 2.10. **What is cause-effect graphing technique ? Also, write the notations.**

OR

What notations are used in cause-effect graphing technique ?

OR

Write the benefits of cause-effect graphing technique.

Answer

1. In software testing, a cause-effect graph is a directed graph that maps a set of causes to a set of effects.

2. The causes may be thought of as the input to the program, and the effects may be thought of as the output.
3. Usually, the graph shows the nodes representing the causes on the left side and the nodes representing the effects on the right side.
4. There may be intermediate nodes in between that combine inputs using logical operators such as AND and OR.
5. A "cause" represents a distinct input condition that brings about an internal change in the system.
6. An "effect" represents an output condition, a system transformation or a state resulting from a combination of causes.
7. The graph's direction is as follows :
   Causes → intermediate nodes → Effects

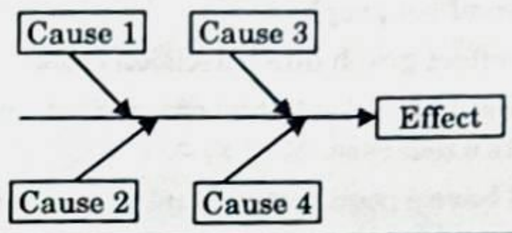


Fig. 2.10.1. Flow diagram of cause-effect graphing.

The cause-effect diagram can be used under these circumstances :

1. To determine the current problem so that right decision can be taken very fast.
2. To narrate the connections of the system with the factors affecting a particular process or effect.
3. To recognize the probable root cause, the cause for an exact effect, problem, or outcome.

Symbols used in cause-effect graphs :

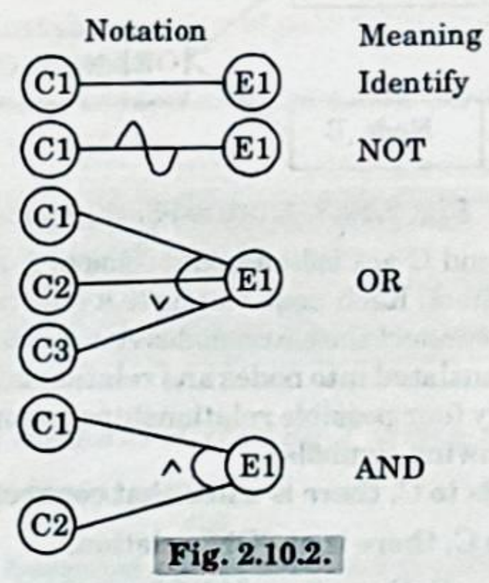


Fig. 2.10.2.

Join Telegram Group :- @AKTU_Group Scanned by TapScanner

Join Telegram Group :- @AKTU_Group www.aktutor.in

1. Just assume that each node having the value 0 or 1 where 0 shows the 'absent state' and 1 shows the 'present state'.
2. The identity function states when C1 = 1, E1 = 1 or we can say if C0 = 0 and E0 = 0.
3. The NOT function states that, if C1 = 1, E1 = 0 and vice-versa.
4. Likewise, OR function states that, if C1 or C2 or C3 = 0, E1 = 0 else E1 = 1.
5. The AND function states that, if both C1 and C2 = 1, E1 = 1, else E1 = 0.
6. The AND and OR functions are permitted to have any number of inputs.

Steps to proceed on cause-effect diagram :

1. Recognize and describe the input conditions (causes) and actions (effect).
2. Build up a cause-effect graph.
3. Convert cause-effect graph into a decision table.
4. Convert decision table rules to test cases. Each column of the decision table represents a test case.
5. For example : I have a requirement that says : "If A OR B, then C." The following rules hold for this requirement :
   a. If A is true and B is true, then C is true.
   b. If A is true and B is false, then C is true.
   c. If A is false and B is true, then C is true.
   d. If A is false and B is false, then C is false.

The cause-effect graph that represents this requirement is provided in Fig. 2.10.3. The cause-effect graph shows the relationship between the causes and effects.

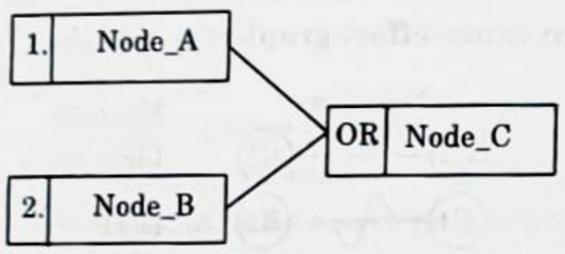


Fig. 2.10.3. A cause-effect graph.

In Fig. 2.10.3, A, B and C are called nodes. Nodes A and B are the causes, while node C is an effect. Each node can have a true or false condition. The lines, called vectors, connect the cause nodes A and B to the effect node C. All requirements are translated into nodes and relationships on the cause-effect graph. There are only four possible relationships among nodes, and they are indicated by the following symbols :

a. If A always leads to C, there is a line that connects the nodes.
b. If A or B lead to C, there is an "OR" relation.
c. If A and B lead to C, there is an "AND" relation.

d. If not A leads to C, there is a line that connects nodes.

Benefits of cause-effect graphing :

1. It helps us to determine the root causes of a problem or quality using a structured approach.
2. It uses an orderly, easy-to-read format to diagram cause-and-effect relationships.
3. It indicates possible causes of variation in a process.
4. It identifies areas, where data should be collected for further study.
5. It encourages team participation and utilizes the team knowledge of the process.
6. It increases knowledge of the process by helping everyone to learn more about the factors at work and how they relate.

PART-2

Structural Testing : Control Flow Testing, Path Testing, Independent Paths, Generations of Graph from Program, Identification of Independent Paths, Cyclomatic Complexity, Data Flow Testing, Mutation Testing.

CONCEPT OUTLINE : PART-2

- Structural testing is the testing of structure of the system or component.
- Control flow testing uses the control structure of a program to develop the test cases for the program.
- In general, the number of linearly independent paths is the minimum acceptable number of path for 100 percent coverage of paths in the system.
- Cyclomatic complexity is a software metric used to measure the complexity of a program.

Questions-Answers

Long Answer Type and Medium Answer Type Questions

Que 2.11. **What is structural testing ? Write advantages and disadvantages of structural testing.**

OR

Write the difference between structural and functional testing.

Join Telegram Group :- @AKTU_Group Scanned by TapScanner

Join Telegram Group :- @AKTU_Group www.aktutor.in

Answer

1. The structural testing is the testing of the structure of the system or component.
2. Structural testing is often referred to as 'white box' or 'glass box' or 'clear-box testing' because in structural testing we are interested in what is happening inside the system/application.
3. In structural testing, the testers are required to have the knowledge of the internal implementations of the code.
4. Here, the testers require knowledge of how the software is implemented, how it works. It checks the implementation of the program or code.
5. The objective of structural testing is not to check different input or output conditions but to check different data and programming structure used in the program.
6. During structural testing, the tester is concentrating on how the software does it.
7. For example, a structural technique wants to know how loops in the software are working.
8. Different test cases may be derived to exercise the loop once, twice, and many times.
9. This may be done regardless of the functionality of the software.
10. Structural testing is done :
    a. To understand what is missing in our test suite.
    b. To complement functional testing.
    c. It helps to identify obvious inadequacies.

Categories of structural testing : The whole structural testing is categorized into four divisions :

1. **Statement coverage :** It is the weakest form of testing, as it requires that every statement in the code has to be executed at least once.
2. **Branch coverage :** In this, each branch condition for the program is tested for its true or false values.
3. **Path coverage :** For path coverage, the path of the program is executed at least once, it test individual path for the program.
4. **Condition coverage :** In this type of testing, it checks all possible combinations of conditions. For conditional branches, we execute the TRUE branch at least once and the FALSE branch, at least once. Unlike branch coverage, it tests for both conditional as well as non-conditional branches.

Advantages of structural testing :

1. Forces test developer to carefully give reason about implementation.
2. Reveals errors in "hidden" code.

3. Spots the dead code or other issues with respect to best programming practices.

Disadvantages of structural testing :

1. Expensive as one has to spend both time and money to perform white box testing.
2. Every possibility that few lines of code are missed accidentally.
3. Deep knowledge about the programming language is necessary to perform white box testing.

Difference between structural and functional testing :

S. No.	Structural testing	Functional testing
1.	This testing will uncover error occurred during the coding of the program.	This testing will uncover error occurred during the implementation of requirements and design specifications.
2.	It is concerned about both the result and the process.	It is concerned only about the result but not the processing.
3.	Structural testing is often referred as white box testing. The knowledge of code is very much essential.	Functional testing is often referred as black box testing, no need to know about the coding of the program.

Que 2.12. **What are the steps of control flow testing ?**

OR

Why control flow testing is important ? What are the limitations of control flow testing ?

Answer

Control flow testing :

1. Control flow testing uses the control structure of a program to develop the test cases for the program.
2. It is a testing technique that comes under white box testing.
3. The entire structure, design, and code of the software have to be studied for this type of testing.
4. Often, the testing method is used by developers themselves to test their own code and design as they are very familiar with the code.
5. The test cases are developed to sufficiently cover the whole control structure of the program.
6. This method is implemented with the intention to test logic of the code so that the required results or functionalities can be achieved.

Join Telegram Group :- @AKTU_Group Scanned by TapScanner

Join Telegram Group :- @AKTU_Group
www.aktutor.in

7. Its applicability is mostly to relatively small programs or segments of larger programs, thus it is mostly used for unit testing.
8. The control structure of a program can be represented by the control flow graph of the program.

Process of control flow testing :

The basic idea behind control flow testing is to select paths and come up with the inputs (input values/test cases) to execute through those paths. It includes the following steps :

Step 1 : From the source code, a control flow graph (CFG) is created either manually or automatically (using software).

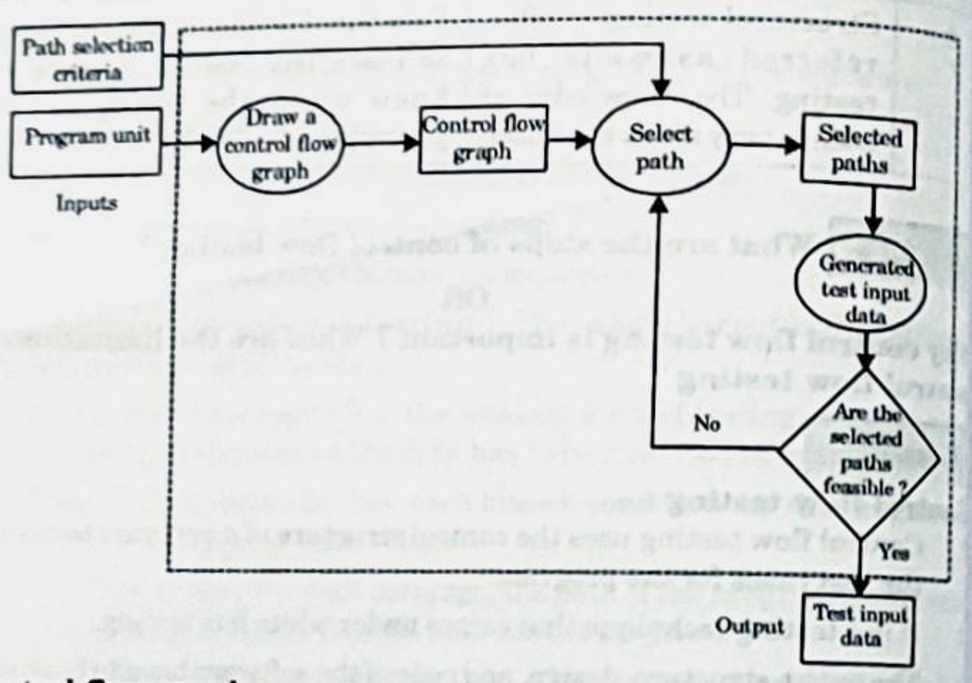
Step 2 : A coverage target is defined over the CFG, for example, nodes, paths, branches, edges etc.

Step 3 : Test cases are created using CFG to cover the coverage target.

Step 4 : Execute the test cases.

Step 5 : Analyze the results and determine whether the program is error free or has some bugs.

Outline of control flow testing :



Control flow graph :

1. The control flow graph is a graphical representation of control structure of a program.
2. The control flow graph $G = (N, E)$ of a program consists of a set of nodes N and a set of edge E. Each node represents a set of program statements.
3. Following notations are used for a flow graph:
   a. **Node :** It represents one or more procedural statements.
   b. **Edges or links :** They represent the flow of control in a program.

   c. **Decision node :** A node with more than one arrow leaving is called a decision node.
   d. **Junction node :** A node with more than one arrow entering it is called a junction.
   e. **Regions :** Areas bounded by edges and nodes are called regions.
4. **For example :**

```
int evensum(int i)
{
int sum = 0;
while (i <= 10)
{
if (i/2 == 0)
sum = sum + i;
i++;
}
return sum;
}
```

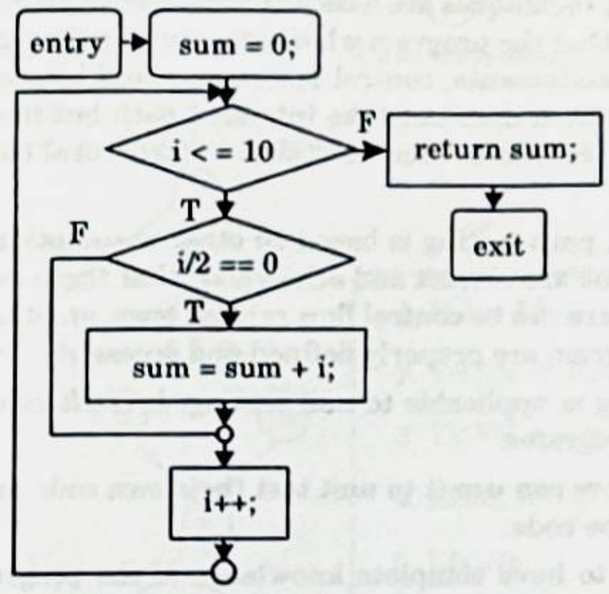


Fig. 2.12.1. Control flow graph.

Limitations of control flow based testing :

1. Control flow testing cannot catch all initialization mistakes.
2. Specification mistakes are not caught by control flow testing.
3. It is unlikely to find missing paths and features if the program and the model on which the tests are based are done by the same person.
4. Interface mismatches and mistakes are not caught.
5. Studies show that control flow testing catches about 50% of all bugs during unit testing.
6. Control flow testing does not verify the correctness of data values "defined" compared to data flow testing.

Join Telegram Group :- @AKTU_Group Scanned by TapScanner

Join Telegram Group :- @AKTU_Group www.aktutor.in

7. It does not focus on the points at which variables receive values and the points at which these values are used.

Que 2.13. **What is path testing ? Also, write the benefits of path testing.**

Answer

Path testing :

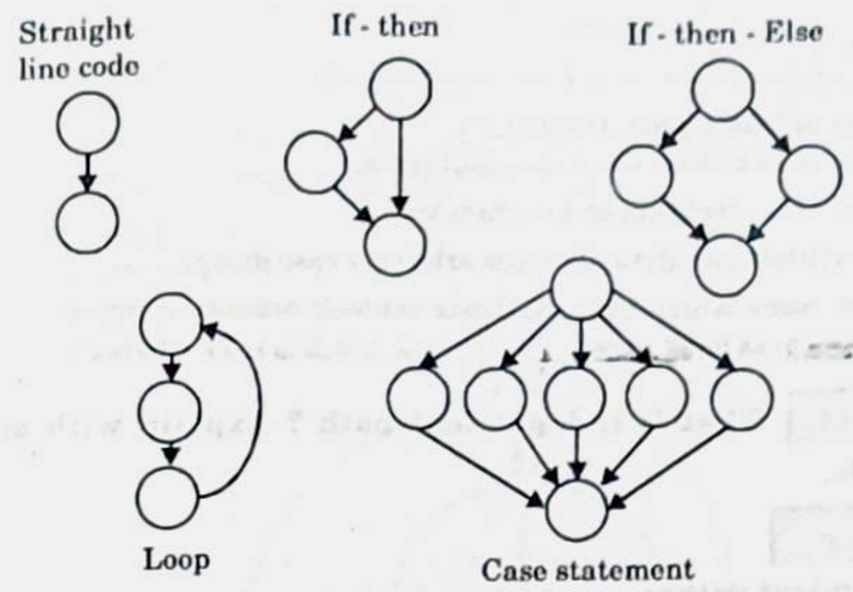
1. The term 'path testing' represents a group of test techniques based on judiciously selecting a set of test paths through the program.
2. Path testing technique is a family of structural testing techniques.
3. The basis for path testing is to select a set of test paths through the program and providing appropriate inputs so as to execute every source statement and branch at least once in that program.
4. Based on how many such test paths are executed successfully, one can measure test thoroughness.
5. Path testing techniques are basically dependent on the bug assumption which says that the program whose structural part is a combination of sequential statements, control statements and loop constructs while getting executed does not take intended path but instead take some other path because of some mistakes in structural constructs of that program.
6. In addition, path testing is based on other assumptions that program specifications are correct and achievable; that there are no processing bugs but there can be control flow related bugs; and that data required for the program are properly defined and accessed.
7. Path testing is applicable to unit testing. It is effective for small and midsized programs.
8. Programmers can use it to unit test their own code and also, for peer review of the code.
9. One needs to have complete knowledge of the program structure in order to make effective path testing.
10. This types of testing involves :
    a. Generating a set of paths that will cover every branch in the program.
    b. Finding a set of test cases that will execute every path in this set of program path.

Path testing techniques :

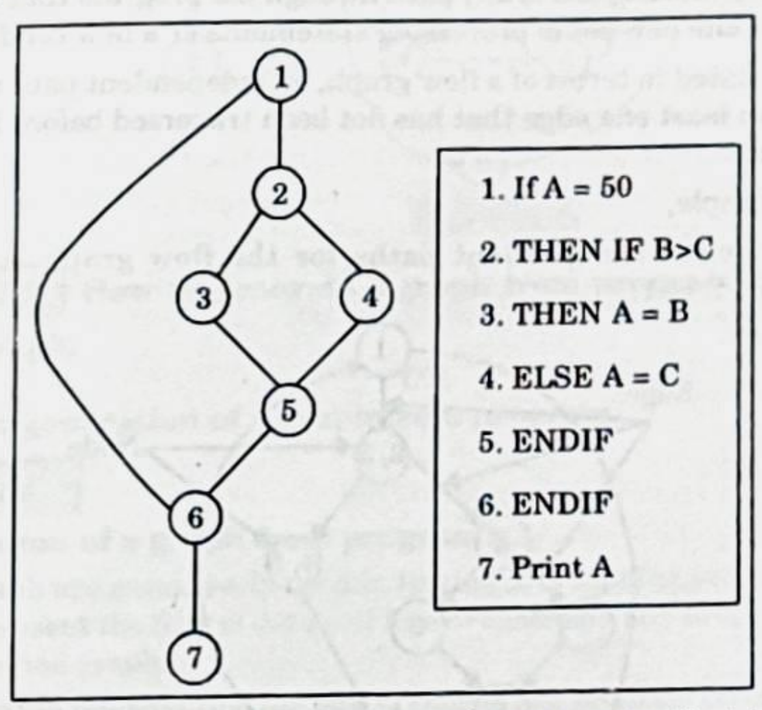
1. **Control flow graph (CFG) :** The program is converted into flow graphs by representing the code into nodes, regions and edges.
2. **Decision to decision path (D-D) :** The CFG can be broken into various decision to decision paths and then collapsed into individual nodes.

3. **Independent (basis) paths :** Independent path is a path through a DD path graph which cannot be reproduced from other paths by other methods.

Notation for representing control flow :



For example,



We can see that there are few conditional statements, that is, executed depending on what condition it ? Here there are 3 paths or conditions that need to be tested to get the output.

1. Path 1 : 1,2,3,4,5,6,7
2. Path 2 : 1,2,4,5,6,7
3. Path 3 : 1,6,7

Join Telegram Group :- @AKTU_Group Scanned by TapScanner

Join Telegram Group :- @AKTU_Group
www.aktutor.in

Steps for basis path testing :

1. Draw a control graph (to determine different program paths).
2. Calculate cyclomatic complexity (metrics to determine the number of independent paths).
3. Find a basis set of paths.
4. Generate test cases to exercise each path.

Benefits of basis path testing :

1. It helps to reduce the redundant tests.
2. It focuses attention on program logic.
3. It facilitates analytical versus arbitrary case design.
4. Test cases which exercise basis set will execute every statement in program at least once.

Que 2.14. **What is independent path ? Explain with suitable example.**

Answer

Independent paths :

1. An independent path is any path through the program that introduces at least one new set of processing statements or a new condition.
2. When stated in terms of a flow graph, an independent path must move along at least one edge that has not been traversed before the path is defined.
3. For example,
   a. A set of independent paths for the flow graph is shown in Fig. 2.14.1.

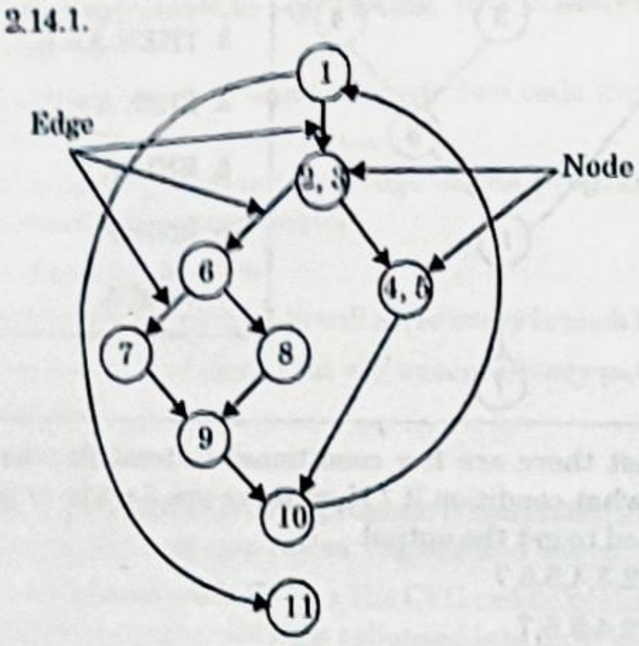


Fig. 2.14.1.

Path 1 : 1-11
Path 2 : 1-2-3-4-5-10-1-11
Path 3 : 1-2-3-6-8-9-10-1-11
Path 4 : 1-2-3-6-7-9-10-1-11

   b. Note that each new path introduces a new edge.
   c. The path 1-2-3-4-5-10-1-2-3-6-8-9-10-1-11 is not considered to be an independent path because it is simply a combination of already specified paths and does not traverse any new edges.
   d. Independent paths for Fig. 2.14.2, are :

Path 1 : 1-7
Path 2 : 1-2-6-1-7
Path 3 : 1-2-3-4-5-2-6-1-7
Path 4 : 1-2-3-5-2-6-1-7

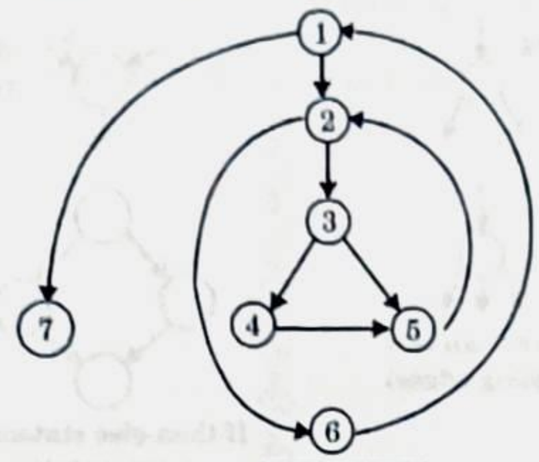


Fig. 2.14.2.

Que 2.15. **How to generate a graph from program ? Explain DD path graph.**

OR

Explain generation of a graph from program.

Answer

Generation of a graph from program :

1. Graph are extensively used in testing of computer programs. We may represent the flow of data and flow of control of any program in terms of directed graphs.
2. A graph representing the flow of control of a program is called a program graph.
3. The program graph may further be transformed into the DD path graph.
4. Both of these graphs may provide foundation to many testing techniques.

Program graph :

1. Program graph is a graphical representation of the source code of program.

Join Telegram Group :- @AKTU_Group Scanned by TapScanner

Join Telegram Group :- @AKTU_Group
www.aktutor.in

2. The statements of a program are represented by nodes and flow of control by edges in the program graph.
3. A program graph is a directed graph in which nodes are either statements or fragments of a statement and edges represent flow of control.
4. The program graph helps us to understand the internal structure of the program which may provide the basis for designing the testing techniques.
5. The basis constructs of the program graph are :

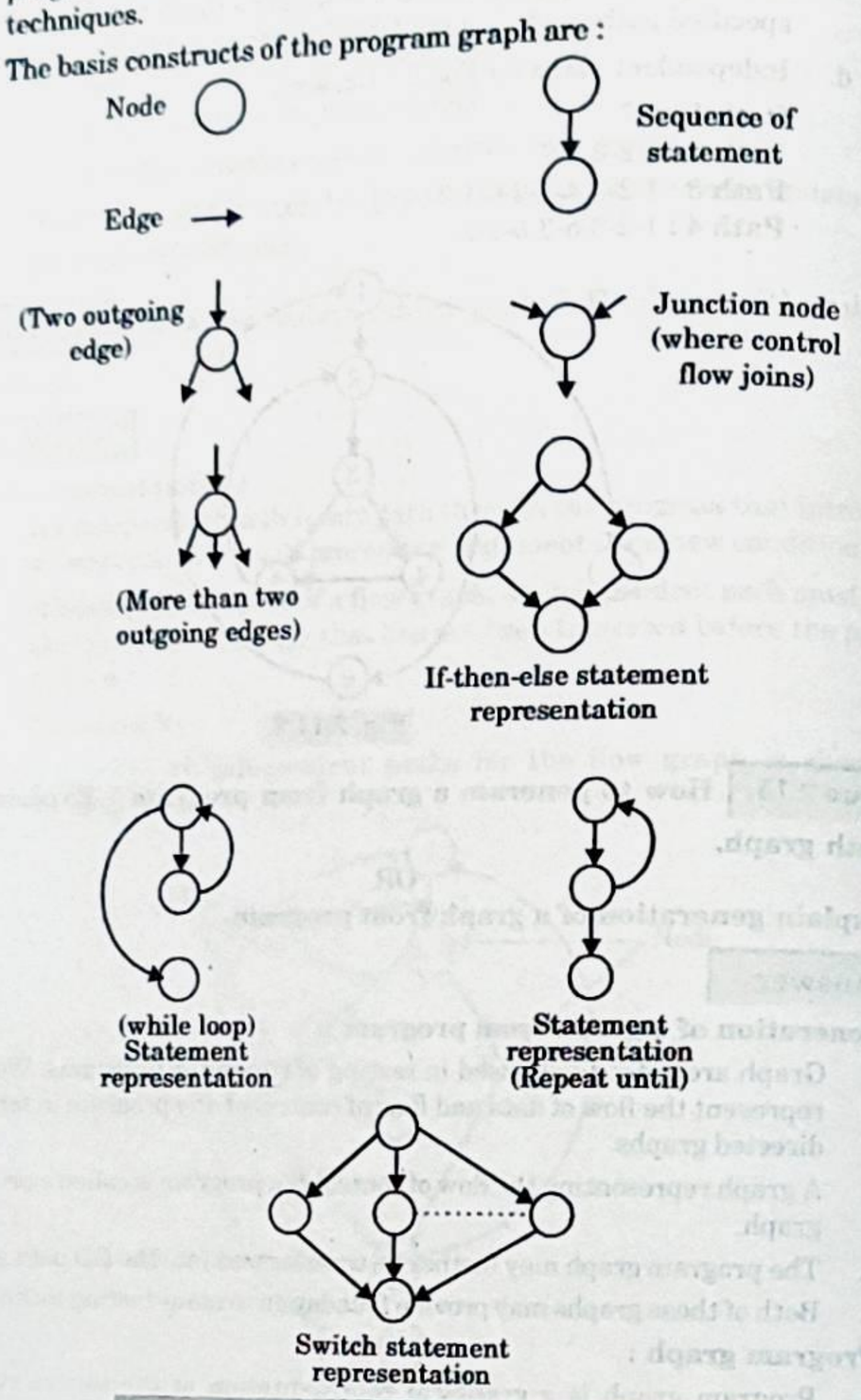


Fig. 2.15.1. Basic constructs of a program graph.

6. The basic constructs are used to convert a program is its program graph.
7. Consider the program square which takes a number as an input and generates the square of the number.

```
#include <stdio.h>
void main()
{
int number, result;
printf("Enter the value");
scanf("%d", &number);
result = number * number;
printf ("The result is : %d", result);
}
```

8. This program has 8 sequential statements which are represented by 8 nodes.
9. All nodes are arranged sequentially which may lead to only one path in this program graph.
10. Every program graph has one source node and one destination node.

Fig. 2.15.2. Program graph for square program.

(Decision to decision) DD path graphs :

1. The decision to decision (DD) path graph is an extension of a program graph.
2. It is widely known as DD path graph.
3. There may be many nodes in a program graph which are in a sequence.
4. When we enter into the first node of the sequence, we can exit only from the last node of that sequence.
5. In DD path graph, such nodes which are in sequence are combined into a block and are represented by a single node.
6. Hence, the DD path graph is a directed graph in which nodes are sequences of statements and edges are control flow amongst the nodes.
7. DD path graph also has a source node and a destination node.
8. Prepare a mapping table for the program graph and the DD path graph nodes.

Join Telegram Group :- @AKTU_Group Scanned by TapScanner

Join Telegram Group :- @AKTU_Group www.aktutor.in

9. A mapping table maps nodes of the program graph to the corresponding nodes of the DD path graph.
10. This may combine sequential nodes of the program graph into a block and, that is, represented by a single node in the DD path graph.
11. This process may reduce the size of the program graph and convert it into a more meaningful DD path graph.
12. Consider square program and its program graph.
13. All nodes are sequential except node 1 and node 8 which are source node and destination node respectively.

Program graph nodes	DD path graph corresponding nodes	Remarks
1	S	Source node
2-7	N1	Sequential flow
8	D	Destination node

14. Mapping of program node and DD path graph node :

Fig. 2.15.3. Mapping table and DD path graph of program graph.

Que 2.16. **How to identify the independent paths in the graph ?**

Answer

Identification of independent paths / Calculation of independent paths :

1. There are three equations from graphing theory that we will use to calculate the number of linearly independent paths through any structured system.
2. These three equations and the theory of linear independence were the work of a Dutch scholar named C.
3. Berge who introduced them in his work graphs and hypergraphs.
4. Specifically, Berge's graph theory defines the cyclomatic number $V(G)$ of a strongly connected graph G with N nodes, E edges, and one connected component.
5. This cyclomatic number is the number of linearly independent paths through the system.

6. We have three definitions of the cyclomatic number.
7. This gives us the following three equations :
   a. (G) = IP = Edges – Nodes + 2 (IP = E – N + 2)
   b. (G) = IP = Regions + 1 (IP = R + 1)
   c. (G) = IP = Decisions + 1 (IP = D + 1)
8. Even though the case statement and the series of required decisions do not have the same number of total paths, they do have the same number of linearly independent paths.
9. The number of linearly independent paths through a system is usually the minimum number of end-to-end paths required to touch every path segment at least once.
10. In some cases, it is possible to combine several path segments that have not been taken previously in a single traversal.
11. This can have the result that the minimum number of paths required to cover the system is less than the number of independent paths.
12. In general, the number of linearly independent paths is the minimum acceptable number of paths for 100 percent coverage of paths in the system.
13. This is the answer to the question, "How many ways can you get through the system without retracing our path ?"
14. The total paths in a system are combinations of the linearly independent paths through the system.
15. If a looping structure is traversed one time, it has been counted as shown in Fig. 2.16.1.

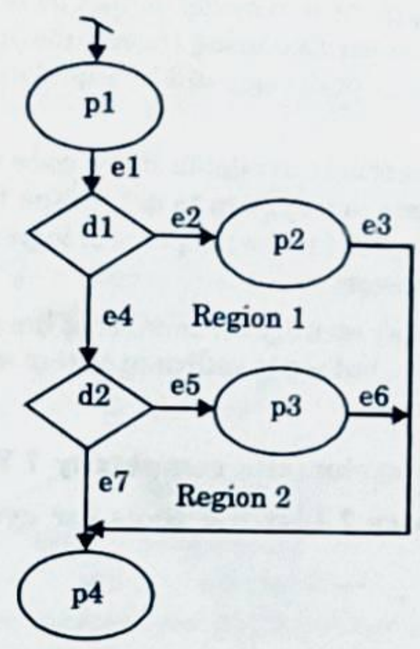


Fig. 2.16.1. Looping structure.

Join Telegram Group :- @AKTU_Group Scanned by TapScanner

Join Telegram Group :- @AKTU_Group www.aktutor.in

16. All three equations must be equal to the same number for the logic circuit to be valid. If the system is not a valid logic circuit, it can not work.
17. First, you have to know the total number of nodes. This is simple to calculate :

    Nodes = Decisions + Processes
18. There are then three equations that you can use to calculate the independent paths.
    a. **Independent paths** = Edges – Nodes + 2
    b. **Independent paths** = Regions + 1
    c. **Independent paths** = Decisions + 1

    These,

    Edges = 7, Decisions = 2, Processes = 4, Regions = 2

    Nodes = Decisions + Processes = 2 + 4 =6

    Independent paths = Edges - Nodes + 2 = 7 – 6 + 2 = 3

    Independent paths = Regions + 1 = 2 +1 = 3

    Independent paths = Decisions +1 = 2 + 1 = 3
19. When inspections are conducted with this in mind, logic problems can be identified quickly.
20. Testers who develop the logic flow diagrams for the system as an aid in test design find all sorts of fuzzy logic errors before they ever begin to test the system.
21. When it is not possible to represent a system without edges that cross, the count of the regions becomes problematic and is often neglected.
22. If the number of regions in a model cannot be established reliably, the logic flow cannot be verified using these equations, but the number of linearly independent paths can still be calculated using the other two equations.
23. Most of the commercially available static code analyzers use only the number of decisions in a system to determine the number of linearly independent paths, but for the purposes of logic flow analysis, all three equations are necessary.
24. Any one by itself may identify the number of linearly independent paths through the system, but is not sufficient to test whether the logic flow of the system is valid.

Que 2.17. **What is cyclomatic complexity ? What are the uses of cyclomatic complexity ? List the tools for cyclomatic complexity calculation.**

Answer

1. The cyclomatic complexity is also known as structural complexity because it gives internal view of the code.

2. This approach is used to find the number of independent paths through a program. This provides us the upper bound for the number of tests that must be conducted to ensure that all statements have been executed at least once and every condition has been executed on its true and false side.
3. If a program has backward branch then it may have infinite number of paths. Although it is possible to define a set of algebraic expressions that gives the total number of possible paths through a program, however, using total number of paths has been found to be impractical.
4. Because of this, the complexity measure is defined in terms of independent paths that when taken in combination will generate every possible path.
5. An independent path is any path through the program that introduces at least one new set of processing statements or a new condition.
6. McCabe's cyclomatic metric, $V(G)$ of a graph G with n vertices, e edges, and P connected components is

$$V(G) = e - n + 2P.$$

7. Given a program, we will associate it with a directed graph that has unique entry and exit nodes. Each node in the graph corresponds to a block of code in the program where the flow is sequential and the arcs correspond to branches taken in the program.
8. This graph is classically known as flow graph and it is assumed that each node can be reached by the entry node and each node can reach the exit node.
9. For example, a flow graph shown in Fig. 2.17.1, below with entry node 'a' and exit node 'f'.
10. The value of cyclomatic complexity can be calculated as :

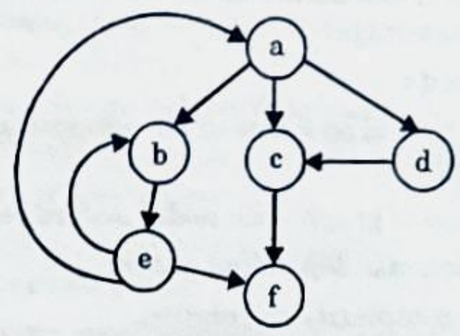


Fig. 2.17.1.

$$V(G) = 9 - 6 + 2 = 5$$

Here $e = 9, n = 6$ and $P = 1$

There will be five independent paths for the flow graph illustrated in Fig. 2.17.1.

path 1 : a c f

path 2 : a b e f

Join Telegram Group :- @AKTU_Group Scanned by TapScanner

Join Telegram Group :- @AKTU_Group www.aktutor.in

path 3 : a d c f
path 4 : a b e a c f or a b e a b e f
path 5 : a b e b e f

Notice that the sequence of an arbitrary number of nodes always has unit complexity and that cyclomatic complexity conforms to our intuitive notion of minimum number of paths. Several properties of cyclomatic complexity are stated below :

a. V(G) > = 1
b. V(G) is the maximum number of independent paths in graph G.
c. Inserting and deleting functional statements to G does not affect V(G).
d. G has only one path if and only if V(G) = 1.
e. Inserting a new row in G increases V(G) by unity.
f. V(G) depends only on the decision structure of G.

For example,

```
if (Condition 1)
statement 1
else
statement 2
if (Condition 2)
statement 3
else
statement 4
```

Cyclomatic complexity for this program will be 9-7+2=4.

As complexity has calculated as 4, four test cases are necessary to the complete path coverage.

Steps to be followed :

The following steps should be followed for computing cyclomatic complexity and test cases design.

Step 1 : Construction of graph with nodes and edges from the code.

Step 2 : Identification of independent paths .

Step 3 : Cyclomatic complexity calculation.

Step 4 : Design of test cases.

Once the basic set is formed, test cases should be written to execute all the paths.

Tools used for cyclomatic complexity calculation :

1. Many tools are available for determining the complexity of the application.
2. Some complexity calculation tools are used for specific technologies.

3. Complexity can be found by the number of decision points in a program.
4. The decision points are if, for, for-each, while, do, catch, case statements in a source code.
5. **Examples of tools are :**
   a. **OCLint :** Static code analyzer for C and related languages.
   b. **devMetrics :** Analyzing metrics for C# projects.
   c. **Reflector Add In :** Code metrics for .NET assemblies.
   d. **GMetrics :** Find metrics in java related applications.
   e. **NDepends :** Metrics in java applications.

Uses of cyclomatic complexity :

1. It helps developers and testers to determine independent path executions.
2. Developers can assure that all the paths have been tested at least once.
3. It helps us to focus more on the uncovered paths.
4. Improve code coverage.
5. Evaluate the risk associated with the application or program.
6. Using these metrics early in the cycle reduces more risk of the program.

Que 2.18. **Explain data flow testing and its strategies. Write some advantages of data flow testing.**

Answer

Data flow testing :

1. Data flow testing is a technique used to detect improper use of data in a program.
2. By looking data usage, risky areas of code can be found and more test cases can be applied.
3. To test data flow, we devise control flow graph.
4. A data flow graph (DFG) is a graph which represents data dependencies between a numbers of operations.
5. In control flow testing, we find various paths of a program and design test case to execute those paths.
6. We may like to execute every statement of program at least once before the completion of testing.
7. Consider a program :

```
#include< stdio.h>
void main()
{
int a, b, c;
a = b + c;
printf("%d", a);
```

Join Telegram Group :- @AKTU_Group Scanned by TapScanner

Join Telegram Group :- @AKTU_Group
www.aktutor.in

8. What will be the output ? The value of 'x' may be previously stored in the memory location assigned to variable 'x' or garbage value.
9. If we execute the program, we may get an unexpected value.
10. The mistake is in the usages of this variable without first assigning a value to it.
11. Data flow testing may help us to minimize such mistakes.
12. It has nothing to do with data flow diagram.
13. It is based on variables, their usages and their definitions in the program.
14. The main point of concern are :
    a. Statements where variable receive values (definitions).
    b. Statements where these values are used (referenced).

Data flow testing strategies :

1. **All definitions :** Test cases for each definition of each variable.
2. **All predicate uses :** Test cases are generated so that there is at least one path of each variable definition to each P-use of variable.
3. **All computational uses :** Test cases are generated so that there is at least one path of each variable definition to each C- use of variable.
4. **All P uses some C uses :** Test cases for every variable, there is a path from every definition to every p-use of that definition. If there is a definition with no p-use following it, then a c-use of the definition is considered.
5. **All C uses some P uses :** Test cases for every variable, there is a path from every definition to every c-use of that definition. If there is a definition with no c-use following it, then a p-use of the definition is considered.
6. **All uses :** Test cases for every use of the variable, there is a path from the definition of that variable to the use.
7. **All du paths :** This is the strongest data flow testing strategy. Every du path from every definition of every variable to every use of that definition.

Data flow anomalies :

1. Data flow anomalies represent the rules which help to detect improper use of data. The notation used to define these anomalies :
    a. d - defined, created, initialized
    b. k - undefined, killed
    c. u - used
       c - Computation use
       p - Predicate use
    d. ~s - all prior actions are not of interest to s

    e. s~ - all post actions are not of interest to s
2. Anomalies which shows normal use of data :

a.	~d	Defined first
b.	dd	define -> use
c.	kd	kill -> define
d.	ud	use -> define
e.	uk	use -> kill
f.	uu	use -> use
g.	k~	kill last
h.	u~	use last

3. Anomalies which shows bug and should be avoided in the programs :

a.	~u	first use Bug.	Data is used without definition.
b.	~k	first kill Bug.	Data is killed without defining it.
c.	dd	define-define	Bug. Redefinition of data.
d.	dk	define -> kill	Bug. Data is killed without using it.
e.	kk	kill - kill	Bug. Destroying already killed data.
f.	ku	kill - use	Bug. Data is used after destroying it.
g.	d~	define last	Bug. Defining but not using it.

Que 2.19. **What is mutation testing and mutation score ? Write advantages and disadvantages of mutation testing.**

OR

Write the steps to execute mutation testing. Explain types of mutation testing.

Answer

Mutation testing :

1. Mutation testing is a method of software testing in which program or source code is deliberately manipulated, followed by suite of testing against the mutated code.
2. The mutations introduced to source code are designed to imitate common programming errors.
3. A good unit test suite typically detects the program mutations and fails automatically.
4. Mutation testing is a kind of testing in which the application is tested for the code that was modified after fixing a particular bug/defect.
5. It also helps in finding out which code and which strategy of coding can help in developing the functionality effectively.
6. Mutation testing ensures the accuracy of our unit tests.

Join Telegram Group :- @AKTU_Group Scanned by TapScanner

Join Telegram Group :- @AKTU_Group www.aktutor.in

7. In some cases, our unit tests will be successful and would not detect any bugs or defects in our source code, but that may be not true.
8. In these situations, we will do some alteration in source code (mutating) for making errors and again will run those unit test.

Following are the steps to execute mutation testing :

Step 1 : Faults are introduced into the source code of the program by creating many versions called mutants. Each mutant should contain a single fault, and the goal is to cause the mutant version to fail which demonstrates the effectiveness of the test cases.

Step 2 : Test cases are applied to the original program and also to the mutant program. A test case should be adequate, and it is tweaked to detect faults in a program.

Step 3 : Compare the results of original and mutant program.

Step 4 : If the original program and mutant programs generate the same output, then that the mutant is killed by the test case. Hence, the test case is good enough to detect the change between the original and the mutant program.

Step 5 : If the original program and mutant program generate different output, Mutant is kept alive. In such cases, more effective test cases need to be created that kill all mutants.

Types of mutation testing :

1. **Value mutations :** An attempt to change the values to detect errors in the programs. We usually change one value to a much larger value or one value to a much smaller value. The most common strategy is to change the constants.
2. **Decision mutations :** The decisions/conditions are changed to check for the design errors. Typically, one changes the arithmetic operators to locate the defects and also, we can consider mutating all relational operators and logical operators (AND, OR, NOT).
3. **Statement Mutations :** Changes done to the statements by deleting or duplicating the line which might arise when a developer is copy pasting the code from somewhere else.

Create mutant programs :

A mutation is nothing but a single syntactic change that is made to the program statement. Each mutant program should differ from the original program by one mutation.

Original program : If (x>y) Print "Hello" Else Print "Hi"

Mutant program : If (x<y< strong =" ">)</y<> Print "Hello" Else Print "Hi"

What to change in a mutant program ?

There are several techniques that could be used to generate mutant programs. Let's look at them :

Operand replacement operators	Expression modification operators	Statement modification operators
Replace the operand with another operand (x with y or y with x) or with the constant value.	Replace an operator or insertion of new operators in a program statement.	Programmatic statements are modified to create mutant programs.
For example-If (x>y) replace x and y values If (5>y) replace x by constant 5.	For example-If (x == y) we can replace == into >= and have mutant program as If(x>=y) and inserting ++ in the statement If(x==++y)	For example-Delete the else part in an if-else statement. Delete the entire if-else statement to check how program behaves. Some of sample mutation operators : 1. GOTO label replacement 2. Return statement replacement 3. Statement deletion 4. Unary operator insertion (Like- and ++)' 5. Logical connector replacement

Automation of mutation testing : Mutation testing is extremely time consuming and complicated to execute manually. To speed up the process, it is advisable to go for automation tools. Automation tools reduce cost of testing as well.

Tools used in mutation testing are :

1. **Mutagenesis :** PHP mutation testing framework.
2. **Jester :** Mutation testing tool for java.

Mutation score :

1. The mutation score is defined as the percentage of killed mutants with the total number of mutants.

   Mutation score = (Killed mutants / Total number of mutants) * 100
2. Test cases are mutation adequate if the score is 100%.
3. Experimental results have shown that mutation testing is an effective approach for measuring the adequacy of the test cases.
4. But, the main drawback is that the high cost of generating the mutants and executing each test case against that mutant program.

Join Telegram Group :- @AKTU_Group Scanned by TapScanner

Join Telegram Group :- @AKTU_Group www.aktutor.in

Advantages of mutation testing :

1. It is a powerful approach to attain high coverage of the source program.
2. This testing is capable of testing the mutant program.
3. Mutation testing brings a good level of error detection to the software developer.
4. This method uncovers ambiguities in the source code, and has the capacity to detect all the faults in the program.
5. Customers are benefited from this testing by getting most reliable and stable system.

Disadvantages of mutation testing :

1. Mutation testing is extremely costly and time consuming since there are many mutant programs that need to be generated.
2. Since it is time consuming, it is fair to say that this testing cannot be done without an automation tool.
3. Each mutation will have the same number of test cases than that of the original program. So, a large number of mutant programs may need to be tested against the original test suite.
4. As this method involves source code changes, it is not at all applicable for black box testing.

☺☺☺

3 UNIT

Regression Testing and Prioritization Technique

Part-1 ..... (90E - 101E)

- *Regression Testing : What is Regression ?*
- *Regression Test Cases Selection*
- *Reducing the Number of Test Cases*
- *Code Coverage Prioritization Technique*

A. Concept Outline : Part-1 ..... 90E
B. Long and Medium Answer Type Questions 90E

Part-2 ..... (101E - 106E)

- *Reducing the Number of Test Cases*
- *Prioritization Guidelines*
- *Priority Category, Scheme*
- *Risk Analysis*

A. Concept Outline : Part-2 ..... 101E
B. Long and Medium Answer Type Questions 101E

Join Telegram Group :- @AKTU_Group Scanned by TapScanner

Join Telegram Group :- @AKTU_Group www.aktutor.in

PART-1

Regression Testing : What is Regression ? Regression Test Cases Selection, Reducing the Number of Test Cases, Code Coverage Prioritization Technique.

CONCEPT OUTLINE : PART-1

- Regression testing identifies new faults that may have been introduced as correct ones.
- Prioritizing the test cases depends on the impact, critical and frequently used functionality.
- Various schemes for test case reduction are :
  - a. Priority category scheme
  - b. Risk analysis
  - c. Interviewing to identify problem areas
  - d. Combination schemes.

Questions-Answers

Long Answer Type and Medium Answer Type Questions

Que 3.1. **What is regression testing ? Discuss various types of regression testing.**

Answer

Regression Testing :

1. Software undergoes constant changes.
2. Such changes are necessitated because of defects to be fixed, enhancements to be made to existing functionality or new functionality to be added.
3. Anytime such changes are made, it is important to ensure that :
   - a. The changes or additions work as designed.
   - b. The changes or additions do not break something that is already working and should continue to work.
4. Regression testing is designed to address the above two purposes.
5. For example :
   - a. Assume that in given release of a product, there were three defects *i.e.*, D^1, D^2 and D^3, the development team will fix these defects and the testing team will perform tests to ensure that these defects are indeed fixed.

   - b. When the customers start using the product, they may encounter new defects, D^4 and D^5.
   - c. Again, the development and testing team will fix and test these new defect fixes.
   - d. But, in this process defect D^1 is again occur.
6. Thus, the testing team should not only ensure that the fixes take care of the defects they are supposed to fix, but also that they do not break anything else that was already working.
7. Regression testing enables the test team to meet this objective.
8. Regression testing enables that any new feature introduced to the existing product does not adversely affect the current functionality.
9. Regression testing follows "Selective re-testing technique".
10. Whenever the defect fixes are done, a set of test cases that need to be run to verify the defect fixes are selected by the test team.
11. Since, this testing technique focuses on reuse of existing test cases that have already been executed, the technique is called selective re-testing. But also sometimes new test cases need to be developed for testing.

Types of Regression Testing : There are two types of regression testing in practice :

1. Regular regression testing
2. Final regression testing

1. Regular regression testing :

- a. A regular regression testing is done between test cycles to ensure that the defect fixes that are done and the functionality that were working with the earlier test cycles continue to work.
- b. A regular regression testing can use more than one product build for the test cases to be executed.
- c. A build is an aggregation of all the defect fixes and features that are present in the product.

2. Final regression testing :

- a. It is done to validate the final build before release.
- b. The CM engineer delivers the final build with the media and other contents exactly as it would go to the customer.
- c. The final regression test cycle is conducted for a specific period of duration, which is mutually agreed upon between the development and testing teams. This is called "Cook time" for regression testing.
- d. Cook time is necessary to keep testing the product for certain duration, since some of the defects (as memory leaks) can be unearthed only after the product has been used for certain time duration.

Join Telegram Group :- @AKTU_Group Scanned by TapScanner

Join Telegram Group :- @AKTU_Group www.aktutor.in

e. The product is continuously exercised for the complete duration of the cook time to ensure that such time-bound defects are identified.

f. All the defect fixes for the release should have been completed for the build used of the final regression test cycles.

g. The final regression test cycle is more critical than any other type or phase of testing, as this is the only testing that ensures the same build of the product that was tested reaches the customer.

Que 3.2. When to do regression testing and how to do regression testing ?

Answer

When to do regression testing : It is necessary to perform regression testing when :

1. Whenever changes happen to software. A reasonable amount of initial testing is already carried out. Between test cycles to find out if the software delivered is as good than the build received in the past.
2. A good number of defects has been fixed to ensure that there are no side-effects after fixing defects.
3. Defect fixes that can produce side-effects are taken care of.
4. May also be performed periodically, as a pro-active measure.
5. Regression test is applicable to all phase in a software development life cycle (SDLC) and also to component, integration, system, and acceptance test phases.

How to do regression testing (Regression Testing Process) :

There are several methodologies for regression testing made of the following steps :

1. Performing an initial "Smoke" or "Sanity" test.
2. Understanding the criteria for selecting the test cases.
3. Classifying the test cases into different priorities.
4. A methodology for selecting test cases.
5. Resetting the test cases for test execution.
6. Concluding the results of a regression cycle.

1. **Performing an initial "Smoke" or "Sanity" test :** Smoke testing consists of :

   a. Identifying the basic functionality that a product must satisfy. For example : If one builds a database, then any build of the database should be able to start it up and perform basic operations such as queries, data manipulation and shutdown of the database.

   b. Designing test cases to ensure that these basic functionality work and packaging them into a smoke test suite;

   c. Ensuring that every time a product is built, this suite is run successfully before anything else is run;

   d. If this suite fails, escalating to the developers to identify the changes and perhaps change or roll back the changes to a state where the smoke test suite succeeds.

   To make sure that problems in smoke testing are defected upfront, some organizations mandate that anytime a developer makes a change, he should run the smoke test suit successfully on that build before checking the code into the Configuration Management repository. Smoke testing enables the uncovering those errors introduced by the build procedures such as defects produced by the build scripts that are used for compiling and linking the programs.

2. **Understanding the criteria for selecting the test cases :** There are two approaches of selecting the test cases for a regression run.

   First : Choose a constant set of regression tests that run for every build or change. In such cases, deciding what test to run is simple.

   Second : Select the test cases dynamically for each build by making judicious choices of the test cases.

   The selection of test cases for regression testing requires knowledge of :

   a. The defect fixes and changes made in the current build.

   b. The way to test the current changes;

   c. The impact that the current changes may have on other parts of the system;

   d. The ways of testing the other impacted parts.

   Some of the criteria to select test cases for regression testing are as follows :

   a. Include test cases that have produced the maximum defects in the past.

   b Include test cases for a functionality in which a changes has been made.

   c. Include test cases in which problems are reported.

   d. Include test cases that test the basic functionality or the core features of the product which are mandatory requirements of the customer.

   e. Include test cases that test the end-to-end behaviour of the application.

   f. Include test cases to test the positive test conditions.

   g. Includes the area which is highly visible to the users.

3. **Classifying test cases :** To enable choosing the right tests for regression run, the test cases can be classified into various priorities based on importance and customer usage.

Join Telegram Group :- @AKTU_Group Scanned by TapScanner

Join Telegram Group :- @AKTU_Group www.aktutor.in

Priority 0 : These test cases are called sanity test cases which check basic functionality and are run for accepting the build for further testing and project goes through a major changes.

Priority 1 : Uses the basic and normal setup and deliver high project value to both developer and customer.

Priority 2 : Deliver moderate project value and are executed as part of testing cycle.

4. **Methodology for selecting test cases :** There are several methodologies available in the industry for selecting regression test cases.

   Case 1 : If the defect fixes are low, then it is enough that a test engineer select a few test cases from TCDB (test case database) which fall under any priority (0, 1, 2).

   Case 2 : If the impact of defect fixes are medium, then only Priority-0 and Priority-1 test case are need to be executed.

   Case 3 : If the impact of the defect fixes are high, then we need to execute all Priority-0, Priority-1 and a carefully selected subset of priority-2 test cases.

5. **Resetting the test case for regression testing :** Resetting test cases reduces the risk involved in testing defect fixes by making the tester go through all the test cases and selecting appropriate test cases based on the impact of those defect fixes. It needs to be done with the following consideration :

   a. When there is a major change in product.

   b. When there is a change in the build procedure which affects the product.

   c. Large release cycle where some test cases were not executed for a long time.

   d. When the product is in the final regression test cycle with a few selected test cases.

   e. Whenever existing application functionality is removed, the related test cases can be reset.

6. **Concluding the results of regression testing :** Sometimes testers or developers monitor the results from regression as they would like to know how well their defect fixes work in the product. It is used to conclude whether regression was successful or not.

   a. If the result of a particular test case was a pass using the previous builds and a fail in the current build, then regression has failed.

   b. If the result of particular test case was a fail using the previous builds and a pass in the current build then it is safe to assume the defect fixes worked.

   c. If the result of a particular test case was a fail using the previous builds but works with a documented workaround then it should be considered as a pass.

   d. If you are satisfied with the workaround, then it should be considered as a pass for both system test cycle and regression test cycle.

Que 3.3. **What methodology is used for selection of regression test cases ?**

OR

What is regression test cases selection ?

Answer

i. Once the test cases are prioritized, test cases can be selected. There could be several approaches to regression testing which need to be decided on a case by case basis.

ii. For example :

   1. **Case 1 :** If criticality and impact of the defect fixes are low, then it is enough to select few test cases from Test Case DataBase (TCDB) and execute them. These can fall under any priority (0, 1, or 2).

   2. **Case 2 :**

      a. If the criticality and the impact of the bug fixes are medium, then we need to execute all Priority-0 and Priority-1 test cases.

      b. If bug fixes need additional test cases from Priority-2, then those test cases can also be selected and used for regression testing.

      c. Selecting Priority-2 test cases in this case is desirable but not a must.

   3. **Case 3:** If the criticality and impact of the bug fixes are high, then we need to execute all Priority-0, Priority-1 and carefully selected Priority-2 test cases.

iii. The above methodology requires impact analysis of bug fixes for all defects. It can be a time consuming process.

iv. If there is not enough time and the risk of not doing impact analysis is low, then the following alternative methodologies are used :

   1. **Regress all :** For regression testing, all priority 0, 1, and 2 test cases are re-run.

   2. **Priority based regression :** For regression testing, based on the priority, all priority 0, 1, and 2 test cases are run in order, based on the availability of time.

   3. **Random regression :** Random test cases are selected and executed.

Join Telegram Group :- @AKTU_Group Scanned by TapScanner

Join Telegram Group :- @AKTU_Group
www.aktutor.in

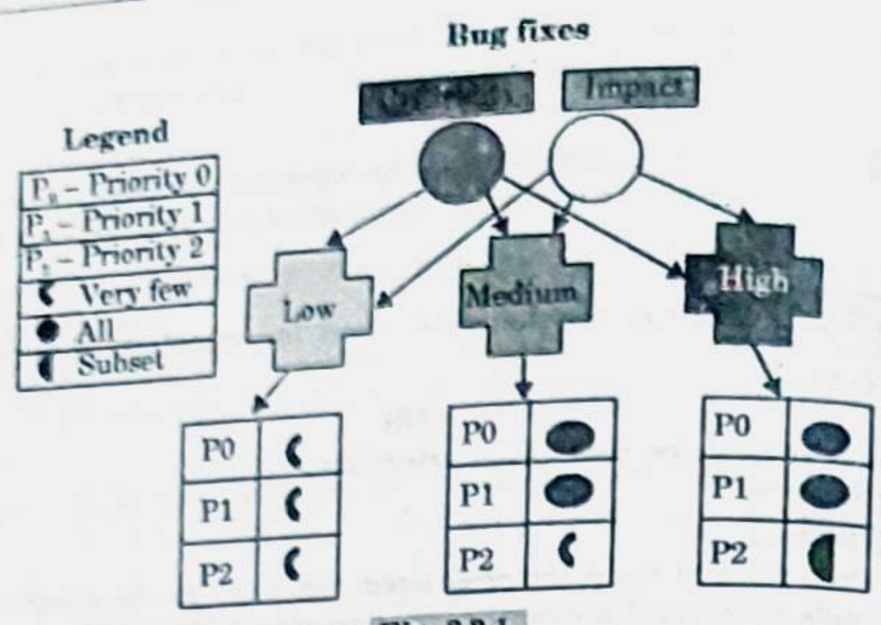


Fig. 3.3.1.

4. **Regress changes :** Code changes are compared to the last cycle of testing and test cases are selected based on their impact on the code.

Que 3.4. **What is prioritization of test cases in regression testing?**

Answer

1. Prioritizing the test cases depends on the business impact, critical and frequently used functionality. Selection of test cases based on priority will reduce the test suit.
2. The test cases may be classified into three categories :
   a. **Priority-0 :** These test cases can be called as Sanity test cases which check the basic functionality and are run for accepting the build for further testing. These are also run when a project goes through major changes. These test cases deliver a very high project value to both development teams and to customers.
   b. **Priority-1 :** Uses the basic and normal setup and these test cases deliver high project value to both development teams and customers.

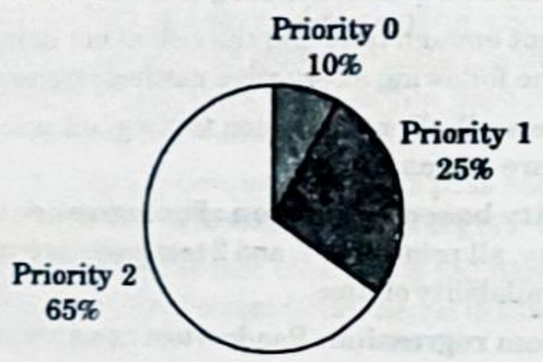


Fig. 3.4.1.

   c. **Priority-2 :** These test cases deliver moderate project value and are executed as a part of software testing life cycle and selected for regression on need basis.
3. Test case prioritization orders test cases with the goal of finding faults in regression testing as quickly as possible.
4. For seed-based symbolic execution, we can also order the seeds, with the goal of gaining incremental coverage as quickly as possible.
5. A large body of work exists on prioritization methods for regression testing, and it is not our purpose to explore all possible strategies.
6. Instead, we aim to demonstrate first, that ordering is important and second, that some approaches that might not be obviously useful in traditional regression testing are effective for seed-based symbolic execution.
7. We considered five ranking strategies, some drawn from the simpler regression approaches and two that were devised with symbolic execution in mind :
   a. **Random Ordering (Random) :** Our first prioritization is not in fact a proposed solution but a baseline to measure the importance of ordering. Random ordering simply chooses a random permutation of test cases.
   b. **Branch Total (BT) :** Prioritize test cases by their branch coverage such that the test with the highest branch coverage is used as a seed first. Ties are broken randomly.
   c. **Branch Additional (BA) :** Prioritize by incremental branch coverage over all tests previously ranked. The first test chosen is therefore the same as with the BT method, but further tests are selected by the additional coverage provided, not absolute coverage.
   d. **Furthest-Point-First (FPF) :**
      i. The first test ranked is chosen randomly. Future test cases are selected by computing the minimum distance to all already-ranked tests, and adding the test case with the largest minimum distance to the ranking.
      ii. The distance function in our case is the Hamming distance between the two test cases branch coverage vectors; even if a test does not produce incremental branch coverage it can therefore be ranked highly if it executes a very different set of branches than any other test.
   e. **Shortest Path (SP) :**
      i. Tests are ranked in order of increasing path length. The number of branches taken during execution is counted, and paths that execute fewer total branches (where the same branch executed multiple times counts each time) in the source of the tested program are chosen first.

Join Telegram Group :- @AKTU_Group Scanned by TapScanner

Join Telegram Group :- @AKTU_Group www.aktutor.in

ii. While BT and BA are selected as examples of simple coverage-based prioritizations from the regression literature, FPF and SP require more justification, since in theory they might well tend to select test cases with poor total or incremental coverage.

iii. In fact, SP seems guaranteed to prioritize test cases that have worse branch coverage, on average.

Que 3.5. **What do you understand by reducing the number of test cases ?**

Answer

1. For most projects it is nearly impossible to execute all of these tests. Although many project schedules are strapped due to tight time constraints, many other reasons exist that necessitate reducing the number of test cases, for example :
   a. Imminent ship date;
   b. Impossibly large number of test cases;
   c. Limited staffing resources;
   d. Limited access to test equipment.
2. There are several techniques that reduce the number of test cases.
3. Although no one method will identify the best test cases, the intent is to categorize the tests on a scale from most important down to least important.
4. The methods described help to identify and prioritize key testing areas, which then guides the tester to create and execute the most crucial test cases first.
5. Any remaining available time can then be used to focus on the next tier of tests.
6. The true art of testing is to select a meaningful subset of test cases that are most likely to uncover problems, thereby reducing the total number of tests while maintaining confidence in the product's operation.
7. Failure to assess the application's potential problems often results in testing the least critical features.
8. Without proper risk assessment, neophyte testers often select inappropriate tests resulting in some undesirable consequences such as :
   a. A tendency to implement and execute the easiest tests;
   b. Focus on one feature, while ignoring others;
   c. Focus on the features that the tester knows well;
   d. Failure to test an entire feature.

9. A tester can apply either or both of these methods during the design phase to identify unique characteristics, and thus avoid creating redundant test cases.
10. These test design methods are :
    a. Equivalence class partitioning consists of dividing the input domain into groups, such that each member of a group evokes similar responses from the application. The tester then creates test cases by selecting representative data from each group.
    b. Orthogonal array testing provides good test coverage when an input domain is small but too large to accommodate testing every possible permutation of the input values. The input domain must have parameters that take on a finite set of possible values, such as enumerated types, a bounded set of numbers, or states in a state machine.
11. Four schemes that focus on prioritizing the existing set of test cases. These reduction schemes are as follows :
    a. Priority category scheme;
    b. Risk analysis;
    c. Interviewing to identify problem areas;
    d. Combination schemes.
12. All of these reduction methods are independent, and no one method is better than the other. Different test case prioritization schemes may generate different lists of prioritized features to test.

Que 3.6. **Explain code coverage prioritization technique in detail.**

Answer

1. We consider a program P with its modified program P_1 and its test suit T created to test P.
2. When we modify P to P_1, we would like to execute modified portion(s) of the source code and portion affected by the modification to see the correctness of modifications.
3. We neither have time nor resources to execute all test cases of T.
4. Our objective is to reduce the size of T to T_1 using some selection criteria, which may help us to execute the modified portion of the source code and the portion affected by modification.
5. A code coverage based technique has been developed which is based on version specific test case prioritization and select T_1 from T which is a subset of T.
6. The technique also prioritizes test case of T_1 and recommends case of high priority test cases first and then law priority test cases in descending order till time and resources are available or a reasonable level of confidence is achieved.

Join Telegram Group :- @AKTU_Group Scanned by TapScanner

Join Telegram Group :- @AKTU_Group www.aktutor.in

Test case selection criteria :

1. The technique is based on version specific test case prioritization where formation about changes in the program is known.
2. Hence, prioritization is focused around the changes in the modified program.
3. We may like to execute all modified line of source code with a minimum number of selected test cases. This technique identifies those test cases that,
   a. Execute the modified line of source code at least once.
   b. Execute the modified lines of source code deletion of deleted line from the execution history of the test case and are not a redundant.
4. The technique uses two algorithms, one for modification and the other for deletion. The following information has been used to design the technique :
   a. Program P with its modified program P_1.
   b. Test suite T with test case t_1, t_2, t_n.
   c. Execution history (number of lines of source code covered by a test case) of each test case suite T.
   d. Line number of lines of source code covered by each test case is stored in two dimensional array $(t_{11}, t_{12}, t_{13}, t_{14} \ldots t_{ij})$.

Modification algorithm : The modification portion of the technique is used to minimize and prioritize test case based on the modification line of source code. The modification algorithm use the following variable name :

1. T1 : It is two dimensional array and is used to store line numbers of lines of source code covered by each test case.
2. modloc : It is used to store the total number of modified line of source code.
3. mod_locode : It is a one-dimensional array and is used to store line numbers of modified line of source code.
4. n found : It is a one dimensional array and is used to store the number of lines of source code matched with modified lines of each test case.
5. pos : It is a one dimensional array and is used to set the position of each test case when nfound is sorted.
6. candidate : It is a one dimensional array. It sets the bit to 1 corresponding to the position of the test case to be removed.
7. priority : It is a one dimensional array and is used to set the priority of the selected test case.

Deletion algorithm : The deletion portion of the technique is used to :

a. Update the execution history of test cases by removing the deleted lines of source code.

b. Identify and remove those test cases that cover only those lines which are covered by other test case of the program.

The variable used in deletion algorithm :

1. T1 : It is a two dimensional array. It keeps the number of lines of source code covered by each test case.
2. deloc : It is used to store the total number of lines of source code deleted.
3. delunderscore locode : It is a one dimensional array and is used to store line numbers of deleted lines of source code.
4. count : It is a two dimensional array. It sets the position corresponding to every matched line of source code of each test case to 1.
5. match : It is a one dimensional array and stores the total count of the number of 1's in the count array for each test case.
6. deleted : It is a one dimensional array. It keeps the record of redundant test cases. If the value corresponding to test case i is 1 in deleted array, then that test case is redundant and should be removed.

PART-2

Reducing the Number of Test Cases : Prioritization Guidelines, Priority Category, Scheme, Risk Analysis.

CONCEPT OUTLINE : PART-2

- The main guide for selecting test is to access risk.
- The easier scheme for categorizing tests is to assign a priority code directly to each test description.
- Risk analysis is a well-defined process that prioritizes modules for testing.

Questions-Answers

Long Answer Type and Medium Answer Type Questions

Que 3.7. **Explain prioritization guidelines.**

Answer

a. Test case prioritization is a compromise, the tester selects the situations to execute at the expense of those to omit. The risk is that some application features will not be tested.

b. The goal of prioritization is to reduce the set of test cases based on some rational, non-arbitrary, criteria, while aiming to select the most appropriate tests.

Join Telegram Group :- @AKTU_Group Scanned by TapScanner

Join Telegram Group :- @AKTU_Group www.aktutor.in

c. The prioritization schemes basically address these key concepts :
   1. What features must be tested ?
   2. What are the consequences if some features are not tested ?
d. The main guide for selecting tests is to assess risk : What price are the project authority and the company willing to pay if a test is not executed ?
e. If the excluded test had been executed, it might have found a problem prior to product delivery. There is no perfect answer.
f. However, initiating a dialog, early in the project life cycle, about the consequences of potential application failures is a vital part of a project.
g. At the end of the test case prioritization exercise, the tester and the project authority must feel confident with the tests that have been selected for execution.
h. If someone is distressed about the omitted tests, then re-evaluate the list and apply another prioritization scheme to analyze the application from another point of view.
i. Ideally, the project authority, and possibly other project members, must buy in and sign off to the prioritized set of tests.

Que 3.8. **What is priority category scheme ?**

Answer

1. The easiest scheme for categorizing tests is to assign a priority code directly to each test description.
2. Although this approach may appear to be arbitrary, it is based on the comparison "is test X more important than test Y ?"
3. The test descriptions can vary in form, such as a test outline, a spreadsheet, a list of tests, a test document's table of contents, or actual test case descriptions.
4. The tester can conduct this exercise himself or as a group exercise with input from the developers, managers, and customer representative.
5. To illustrate, let's consider the following three-level priority categorization scheme :

   Priority 1 : This test must be executed.

   Priority 2 : If time permits, execute this test.

   Priority 3 : If this test is not executed, the team won't be upset.
6. Assigning priority codes is as simple as writing a number adjacent to each test description.
7. Once the priority codes have been assigned, the tester estimates the amount of time required to execute the tests selected in each category.
8. If the time estimate falls within the allotted schedule, then the partitioning exercise is completed and you have identified the tests to use.

9. Otherwise, further partitioning is required. The second round of partitioning can either reuse the same scale or use a new priority scheme.
10. This example uses a new five-level scale to further classify the tests.

    Priority 1a : This test must pass, otherwise, the delivery is in jeopardy.

    Priority 2a : This test must be executed prior to delivery.

    Priority 3a : If time permits, execute this test.

    Priority 4a : This test can wait until the next release or shortly after the delivery date.

    Priority 5a : We will probably never execute this test.
11. Priority (1a) calls attention to the application's most critical features, not only must the test be executed, but the test must also execute successfully.
12. The tests in the original Priority 3 are now moved to Priority 5a. The tests from Priority 2 are now divided between Priorities 3a, 4a, and 5a, whereas the tests from Priority 1 are now partitioned into the new 5-level scheme.
13. Chances are that none of the tests from Priorities 1 and 2 will be reassigned to Priority 5a, but do try to evaluate the validity of tests and determine whether any tests can be downgraded to a lower classification.

Que 3.9. **Explain risk analysis and risk matrix.**

Answer

1. **Risk analysis :**
   a. When the risks have been identified, all items are analyzed using different criteria.
   b. The purpose of the risk analysis is to assess the loss probability and magnitude of each risk item.
   c. The input is the risk statement and context developed in the identification phase.
   d. The output of this phase is a risk list containing relative ranking of the risks and a further analysis of the description, probability, consequence and context.
   e. The main activities in this phase are :
      i. **Group similar risks :** Detect duplicates and find new risk items by grouping the identified risks into categories.
      ii. **Determine risk drivers :** The risk drivers are parameters that affect the identified risk. For example, schedule drivers are included in the critical path model. Determining these properties help to assess and prioritize the risks.
      iii. **Determine source of risks :** The sources of risks are the root causes of the risks. These are determined by asking the

Join Telegram Group :- @AKTU_Group Scanned by TapScanner

Join Telegram Group :- @AKTU_Group www.aktutor.in

question why ? and trying to figure out what may have caused the risk. Several root causes may lead to the same risk.

iv. **Estimate risk exposure :** The risk exposure is a measure of the probability and the consequence of a risk item. The consequence can also be stated in terms of loss (for example, life, money, property, reputation).

v. **Evaluate against criteria :** Each risk item is evaluated using the predefined criteria, which are important for the specific project. Criteria may be stated in terms of the probability of occurrence, the consequence and the time frame. This information is used to prioritize the risks.

Once this is done, risks can be prioritized, and the most serious risks can be identified for monitoring.

2. **Risk matrix :**

a. A risk matrix allows the tester to evaluate and rank potential problems by giving more weight to the probability or severity value as necessary.

b. Use of a risk matrix disregards the risk exposure. The tester uses the risk matrix to assign thresholds that classify the potential problems into priority categories.

c. Typically, the risk matrix contains four quadrants, as shown in Fig. 3.9.1, with each quadrant representing a priority class defined as follows :

Priority 1 : high severity and high probability.

Priority 2 : high severity and low probability.

Priority 3 : low severity and high probability.

Priority 4 : low severity and low probability.

d. In this particular example, a risk with high severity is deemed more important than a problem with high probability. Thus, all risks mapped in the upper left quadrant fall into Priority 2.

e. For an entirely different application, the consensus may be to swap the definitions of Priorities 2 and 3, as shown in Fig. 3.9.2.

f. An organization favouring Fig. 3.9.3, seeks to minimize the total number of defects by focusing on problems with a high probability of occurrence.

g. Although dividing a risk matrix into quadrants is most common, testers can determine thresholds using different types of boundaries based on application-specific needs.

h. Sometimes the best threshold limits are those that appease management fears and address customer needs.

i. If severity and probability tend to be of equal weight, then a diagonal band prioritization scheme, as shown in Fig. 3.9.3, may be more appropriate.

j. This threshold pattern is a compromise for those who have difficulty in selecting between Priority 2 and Priority 3 in the quadrant scheme.

k. Some managers fear that ignoring any problem with high severity, regardless of probability, may make the organization liable for negligence.

l. The prioritization scheme shown in Fig. 3.9.4 addresses this concern by assigning top priority to all high severity problems.

m. The remainder of the risk matrix is partitioned into several lower priorities, whether as quadrants or diagonal bands.

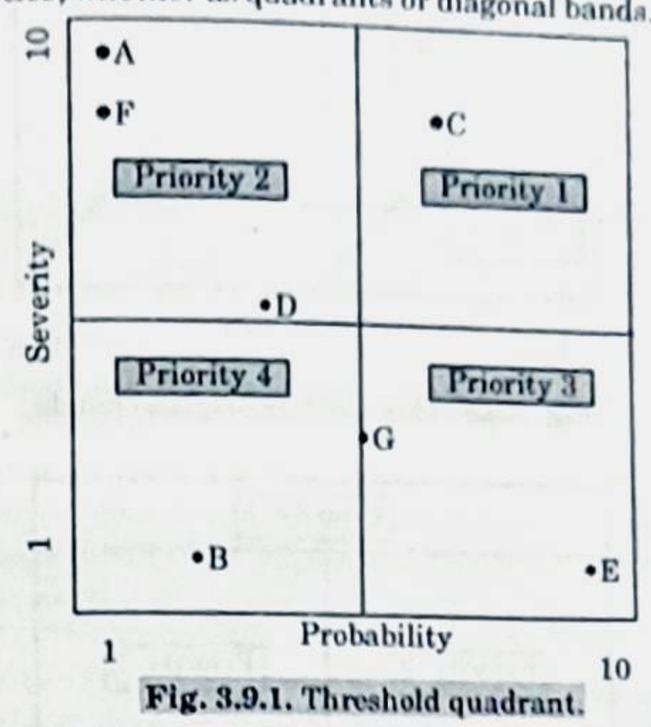


Fig. 3.9.1. Threshold quadrant.

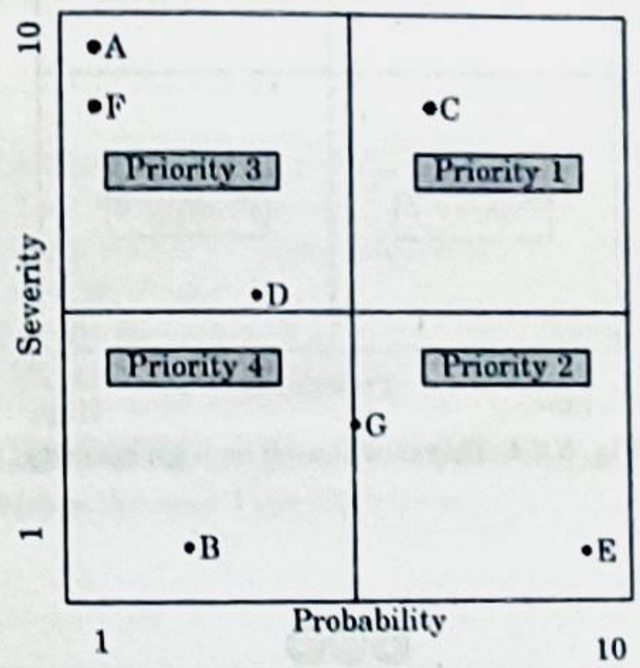


Fig. 3.9.2. Alternate threshold by quadrant.

Join Telegram Group :- @AKTU_Group Scanned by TapScanner

Join Telegram Group :- @AKTU_Group
www.aktutor.in

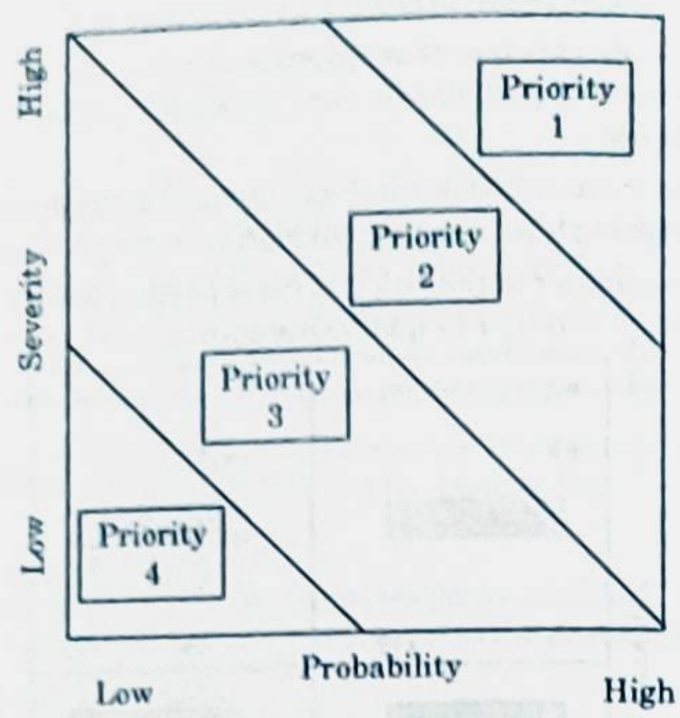


Fig. 3.9.3. Threshold by diagonal bands.

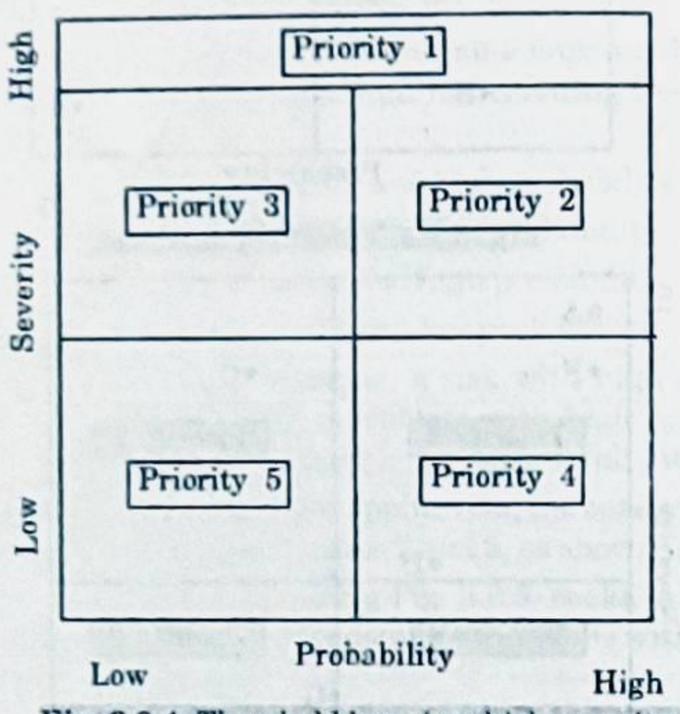


Fig. 3.9.4. Threshold based on high severity.

☺☺☺

4 UNIT

Software Testing Activities

Part-1 ..... (108E - 125E)

- *Software Testing Activities : Levels of Testing*
- *Debugging*

A. Concept Outline : Part-1 ..... 108E
B. Long and Medium Answer Type Questions 108E

Part-2 ..... (126E - 129E)

- *Testing Techniques and their Applicability*
- *Exploratory Testing*

A. Concept Outline : Part-2 ..... 126E
B. Long and Medium Answer Type Questions 126E

Part-3 ..... (130E - 145E)

- *Automated Test Data Generation : Test Data*
- *Approaches to Test Data Generation*
- *Test Data Generation using Genetic Algorithm*
- *Test Data Generation Tools*
- *Software Testing Tools*
- *Software Test Plan*

A. Concept Outline : Part-3 ..... 130E
B. Long and Medium Answer Type Questions 130E

Join Telegram Group :- @AKTU_Group Scanned by TapScanner

Join Telegram Group :- @AKTU_Group www.aktutor.in

PART-1

Software Testing Activities : Levels of Testing, Debugging.

CONCEPT OUTLINE : PART-1

- There are four levels of testing :
  a. Unit/Component testing
  b. Integration testing
  c. System testing
  d. Acceptance testing
- Unit testing focuses verification effort on the smallest unit of software design, the software component or module.
- Various integration techniques are :
  a. Top-down integration
  b. Bottom-up integration
  c. Big-bang testing
- System testing enables testers to ensure that the product meets business requirements, as well as determine that it runs smoothly within its operating environment.
- Acceptance testing is the level in the software testing process which decides whether a product is given the green light or not *i.e.*, approved by the user or not.
- Debugging is defined as a process of analyzing and removing the error.

Questions-Answers

Long Answer Type and Medium Answer Type Questions

Que 4.1. **Explain levels of testing.**

Answer

There are generally four recognized levels of testing :

1. Unit/Component testing
2. Integration testing
3. System testing
4. Acceptance testing

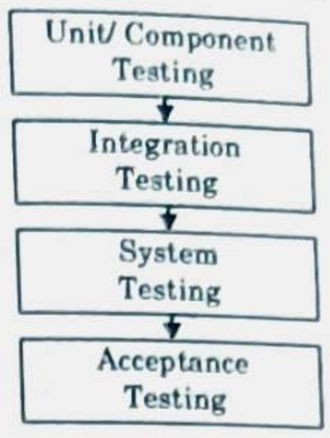


Fig. 4.1.1.

1. **Unit / Component testing :**
   a. The most basic type of testing is unit, or component testing.
   b. Unit testing aims to verify each part of the software by isolating it and then perform tests to demonstrate that each individual component is correct in terms of fulfilling requirements and the desired functionality.
   c. This type of testing is performed at the earliest stages of the development process, and in many cases it is executed by the developers themselves before handing the software over to the testing team.
   d. The advantage of detecting any errors in the software early in the day is that by doing so the team minimizes software development risks, as well as time and money wasted in going back and undo fundamental problems in the program once it is nearly completed.
2. **Integration testing :**
   a. Integration testing aims to test different parts of the system in combination in order to assess if they work correctly together.
   b. By testing the units in groups, any faults in the way they interact together can be identified.
   c. There are many ways to test how different components of the system function at their interface; testers can adopt either a bottom-up or a top-down integration method.
   d. In bottom-up integration testing, testing builds on the results of unit testing by testing higher-level combination of units, (called modules) in successively more complex scenarios.
   e. It is recommended that testers start with this approach first, before applying the top-down approach which tests higher-level modules first and studies simpler ones later.
3. **System testing :**
   a. The next level of testing is system testing.
   b. As the name implies, all the components of the software are tested as a whole in order to ensure that the overall product meets the requirements specified.

Join Telegram Group :- @AKTU_Group Scanned by TapScanner

Join Telegram Group :- @AKTU_Group www.aktutor.in

c. System testing is a very important step as the software is almost ready to ship and it can be tested in an environment which is very close to that which the user will experience once it is deployed.

d. System testing enables testers to ensure that the product meets business requirements, as well as determine that it runs smoothly within its operating environment.

e. This type of test is typically performed by a specialized testing team.

4. **Acceptance testing :**

a. Acceptance testing is the level in the software testing process which decides whether a product is given the green light or not.

b. The aim of this type of testing is to evaluate whether the system complies with the end-user requirements and if it is ready for deployment.

c. The testing team will utilize a variety of methods, such as pre-written scenarios and test cases to test the software and use the results obtained from these tools to find ways in which the system can be improved.

d. The scope of acceptance testing ranges from simply finding spelling mistakes and cosmetic errors, to uncover bugs that could cause a major error in the application.

e. By performing acceptance tests, the testing team can find out how the product will perform when it is installed on the user's system.

Que 4.2. **Explain unit testing and its advantages.**

Answer

1. Unit testing is a testing technique using which individual modules are tested to determine if there are any issues by the developer himself.
2. It is concerned with functional correctness of the standalone modules.
3. The main aim is to isolate each unit of the system to identify, analyze and fix the defects.

Advantages of unit testing :

1. Reduces defects in the newly developed features or reduces bugs when changing the existing functionality.
2. Reduces cost of testing as defects are captured in very early phase.
3. Improves design and allows better refactoring of code.
4. Unit tests, when integrated with build gives the quality of the build as well.

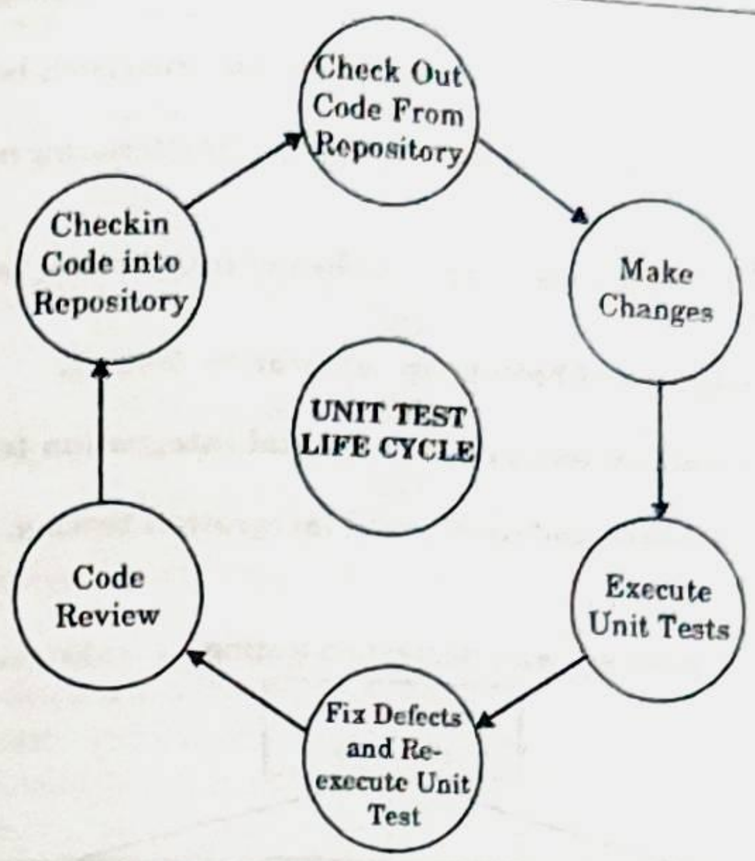


Fig. 4.2.1. Unit Testing Life Cycle.

Que 4.3. **What is integration testing ?**

OR

When all modules have been verified independently, then why integration testing is required ?

Answer

1. In the modular design of software system where the system is composed of different modules, integration is the activity of combining the modules together when all the modules have been prepared.
2. Integration aims at constructing a working software system. But a working software demands full testing and thus, integration comes into the picture.
3. Since modules are not standalone entities, they are a part of a software system which comprises of many interfaces.
4. Even if a single interface is mismatched, many modules may be affected. Thus, integration testing is necessary for the following reasons :
   a. Integration testing exposes inconsistency between the modules such as improper call or return sequence.
   b. Data can be lost across an interface.
   c. One module when combined with another module may not give the desired result.

Join Telegram Group :- @AKTU_Group Scanned by TapScanner

Join Telegram Group :- @AKTU_Group www.aktutor.in

d. Data types and their valid ranges may mismatch between the modules.

Thus, integration testing focuses on bugs caused by interfacing between the modules while integrating them.

Que 4.4. **Discuss various approaches of integration testing.**

OR

Discuss top-down and bottom-up integration testing.

OR

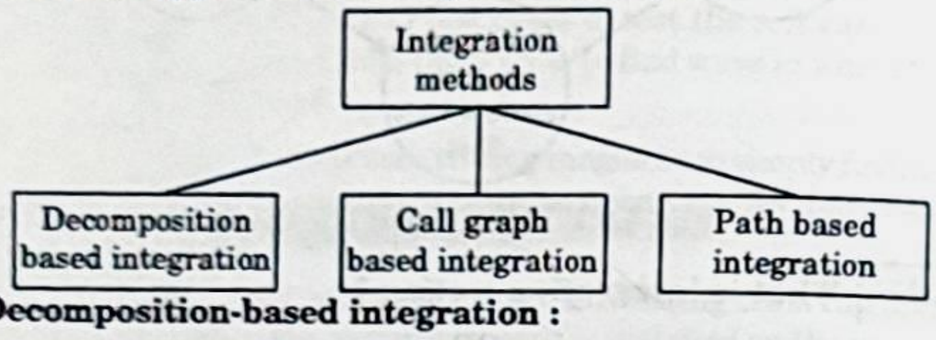
Discuss incremental and non-incremental integration testing.

OR

Discuss graph based and path based integration testing.

Answer

There are three approaches for integration testing :



1. **Decomposition-based integration :**
   a. It is based on the decomposition of design into functional components or modules. It is represented as a tree.
   b. The node represents the modules present in the system and the links/edges between the two modules represent the calling sequence. The nodes on the last level in the tree are leaf nodes.

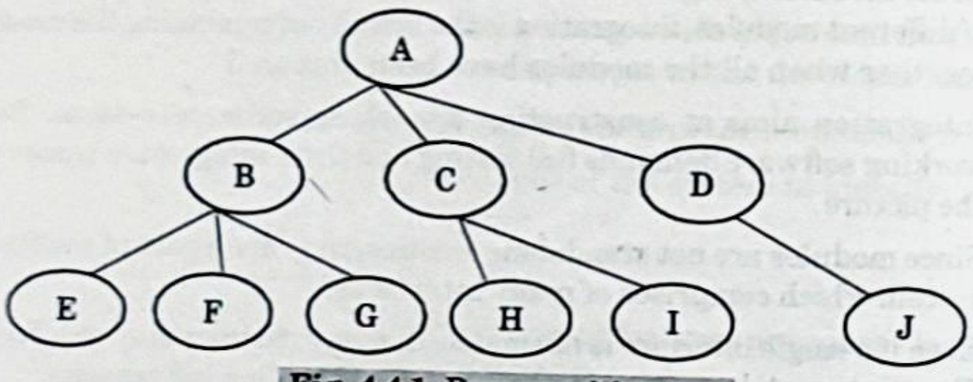


Fig. 4.4.1. Decomposition tree.

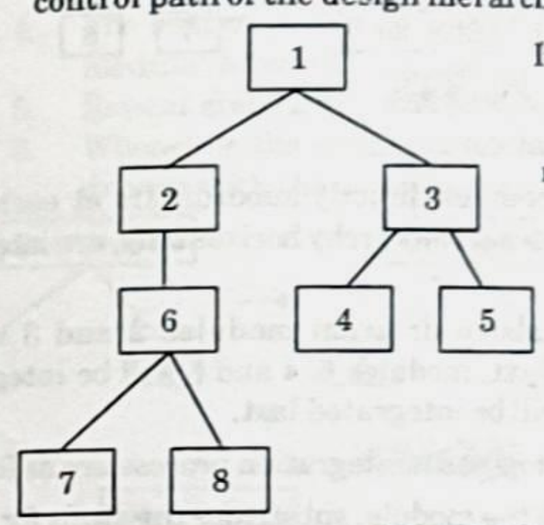
   c. Integration methods in decomposition based integration depend on the methods on which the activity of integration is based.
   d. Integration methods are classified into two categories :
      i. **Non-incremental integration testing :**
         1. In this type of testing, either all untested modules are combined together and then tested or unit tested modules are combined together.

         2. It is also known as Big-bang integration testing.
         3. This method cannot be adopted practically because it will be difficult to localize the errors since the exact location of bugs cannot be found easily and actual modules are not interfaced directly until the end of the software system.
      ii. **Incremental integration testing :**
         1. Here, we can start with one module or unit testing. Then combine the module which has to be merged with it and perform test on both the modules.
         2. In this way, incrementally keep adding the modules and test the recent environment. Thus, an integrated tested software system in achieved.

Types of incremental integration testing : It is divided into two categories :

a. **Top-down integration testing :** The strategy in top-down integration is to look at the design hierarchy from top to bottom. Start with the high level modules and move downward through the design hierarchy. Modules subordinate to the top module are integrated in the following two ways :

   i. **Depth first integration :** In this type, all modules on a major control path of the design hierarchy are integrated first.



In this figure, modules 1, 2, 6, 7/8 will be integrated first. Next, modules 1, 3, 4/5 will be integrated.

Fig. 4.4.2.

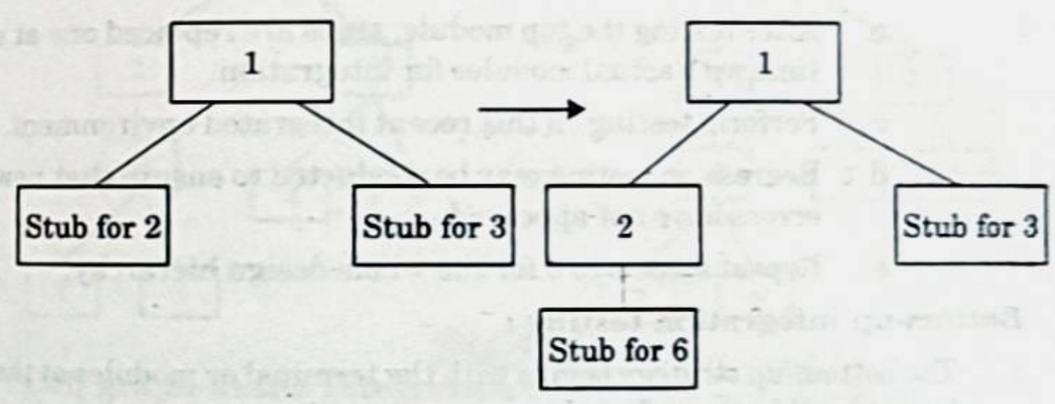


Fig. 4.4.3.

Join Telegram Group :- @AKTU_Group Scanned by TapScanner

Join Telegram Group :- @AKTU_Group www.aktutor.in

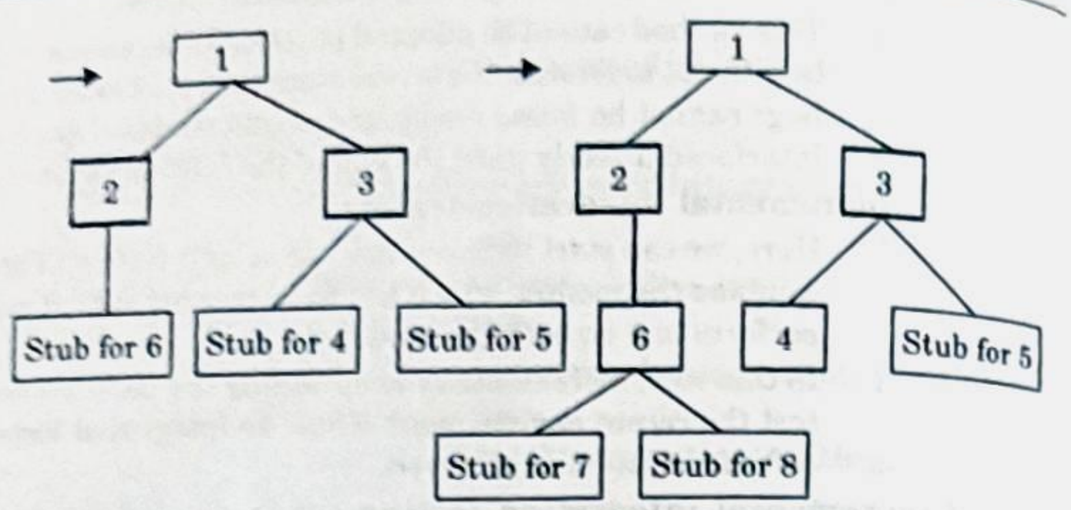


Fig. 4.4.4.

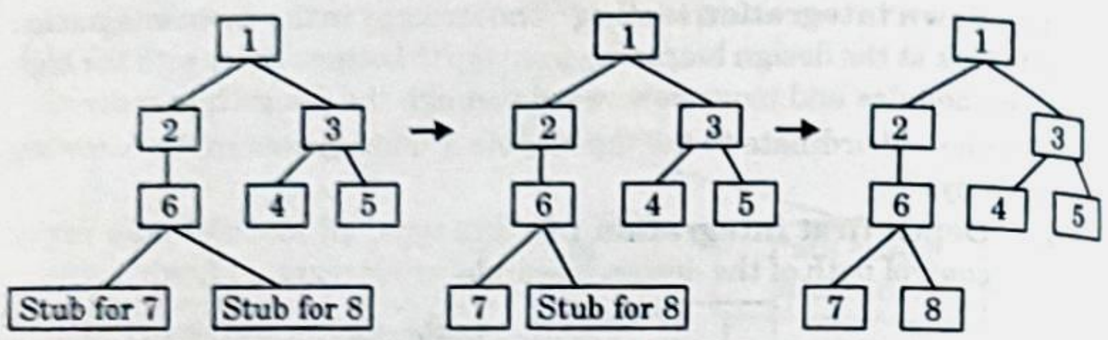


Fig. 4.4.5.

ii. Breadth first integration :

1. In this type, all modules directly subordinate at each level, moving across the design hierarchy horizontally, are integrated first.
2. For example, in above diagram modules 2 and 3 will be integrated first. Next, modules 6, 4 and 5 will be integrated. Modules 7 and 8 will be integrated last.
3. The procedure for top-down integration process are as follows :
   a. Start with the top module, substitute the stubs for all the subordinate modules of top module. Test the top module.
   b. After testing the top module, stubs are replaced one at a time with actual modules for integration.
   c. Perform testing on this recent integrated environment.
   d. Regression testing may be conducted to ensure that new errors have not appeared.
   e. Repeat steps b to d for the whole design hierarchy.

b. Bottom-up integration testing :

i. The bottom-up strategy begins with the terminal or modules at the lowest level in the software structure.

ii. After testing these modules, they are integrated and tested moving from bottom to top level.

iii. Since the processing required for modules subordinate to a given level is always available, stubs are not required in this strategy.

iv. Unlike top-down strategy, this strategy does not require the architectural design of the system to be complete. Thus, bottom-up integration can be performed at an early stage in the development process.

v. It may be used where the system reuses and modifies component from other systems.

vi. The steps in bottom-up integration are as follows :
   1. Start with the lowest level modules in the design hierarchy. These are the modules from which no other module is being called.
   2. Look for the super-ordinate module which calls the module selected in step 1. Design the driver module for this super-ordinate module.
   3. Test the module selected in step 1 with the driver designed in step 2.
   4. The next module to be tested is any module whose subordinate modules have all been tested.
   5. Repeat steps 2 to 5 and move up in the design hierarchy.
   6. Whenever, the actual modules are available, replace stubs and drivers with the actual one and test again.

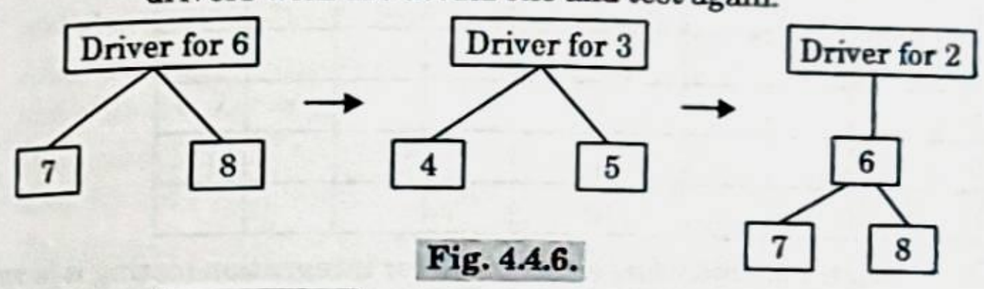


Fig. 4.4.6.

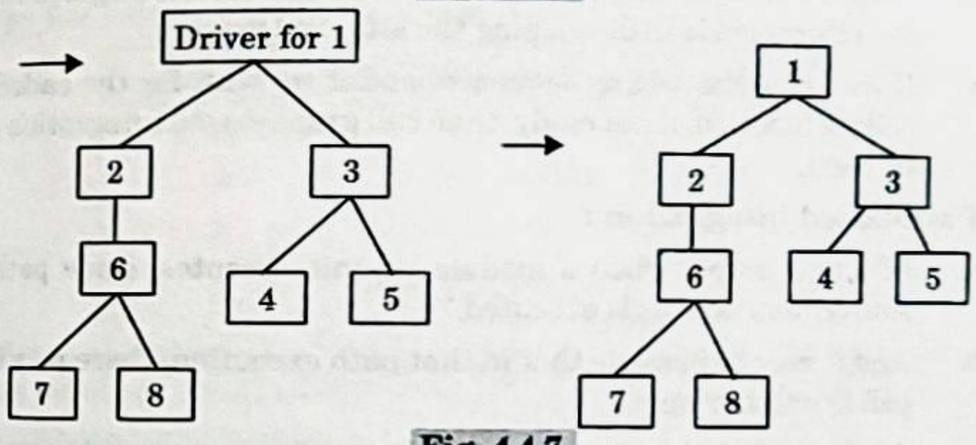


Fig. 4.4.7.

2. Call graph-based integration :

a. Integration testing not only detects bugs which are structural, it also detect some behavioural bugs. This can be done with the help of a call graph.

Join Telegram Group :- @AKTU_Group Scanned by TapScanner

Join Telegram Group :- @AKTU_Group www.aktutor.in

b. A call graph is a directed graph, wherein the nodes are either modules or units and a directed edge from one node to another means one module called another module.

c. The call graph can be captured in a matrix form which is known as the adjacency matrix.

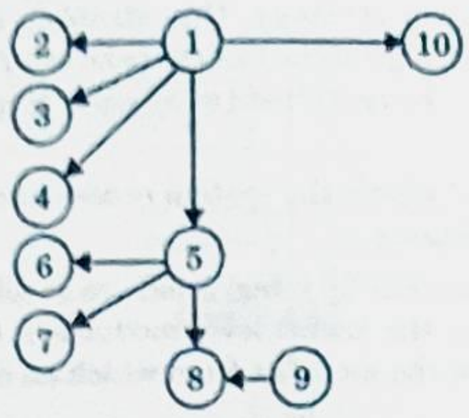


Fig. 4.4.8.

1	2	3	4	5	6	7	8	9	10
1	×	×	×	×					
2									
3									
4									
5					×	×	×		
6									
7									
8								×	
9									
10									

(Adjacency matrix)

d. The idea behind using a call graph for integration testing is to avoid the efforts made in developing the stub and drivers.

e. If we know the calling sequence, and if we wait for the called or calling function, if not ready, then call graph-based integration can be used.

3. **Path-based integration :**

a. In a call graph, when a module or unit executes, some path of source instructions is executed.

b. And it may be possible that in that path execution, there may be a call to another unit.

c. At that point, the control is transferred from one unit to another unit which is necessary for integration testing.

d. Also, there should be information within the module regarding instructions that call the module or return to the module.

e. This must be tested at the time of integration. It can be done with the help of path based integration.

Que 4.5. **What do you understand by Bi-directional integration ?**

OR

What is sandwich integration ?

Answer

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

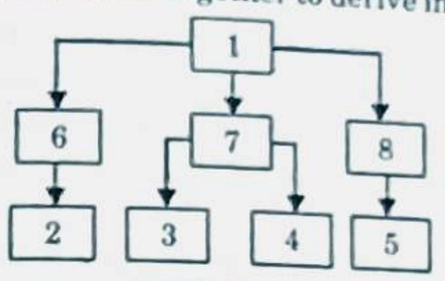


Fig. 4.5.1.

2. Now, the individual modules 1, 2, 3, 4 and 5 are tested separately and bi-directional integration is performed initially with the use of stubs and drivers.
3. Drivers are used to provide upstream connectivity while stubs provide downstream connectivity.
4. A driver is a function which redirects the requests to some other modules and stubs simulate the behaviour of missing module.
5. After the functionality of these integrated modules is tested, the drivers and stub are discarded.
6. Once modules 6, 7 and 8 become available, the integration methodology then focuses only on those modules, which need focus and are new.
7. This approach is also called "Sandwich integration".

Steps for integration using sandwich testing :

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

As shown above, steps 1-3 use a bottom-up integration approach and step 4 uses a top-down integration approach. This approach is used when migrating from two-tier to three-tier environment.

Que 4.6. **What is system testing ?**

Join Telegram Group :- @AKTU_Group Scanned by TapScanner

Join Telegram Group :- @AKTU_Group www.aktutor.in

Answer

1. System testing is a black-box testing technique performed to evaluate the complete system.
2. In system testing, the functionalities of the system are tested from an end-to-end perspective.
3. System testing is usually carried out by a team that is independent of the development team in order to measure the quality of the system unbiased.
4. It includes both functional and non-functional testing.
5. System testing is actually a series of different tests whose sole purpose is to exercise the full computer based system.
6. System testing involves testing the software code for following :
   a. Testing the fully integrated applications including external peripherals in order to check how components interact with one another and with the system as a whole. This is also called end-to-end scenario testing.
   b. Verify thorough testing of every input in the application to check for desired outputs.
   c. Testing of the user's experience with the application.

Types of system testing :

1. **Usability testing :** Usability testing mainly focuses on the user's ease to use the application, flexibility in handling controls and ability of the system to meet its objectives.
2. **Load testing :** Load testing is necessary to know that a software solution will perform under real life loads.
3. **Regression testing :** Regression testing involves testing done to make sure none of the changes made over the course of the development process have caused new bugs. It also makes sure no old bugs appear from the addition of new software modules over time.
4. **Recovery testing :** Recovery testing is done to demonstrate a software solution is reliable, trustworthy and can successfully recoup from possible crashes.
5. **Migration testing :** Migration testing is done to ensure that the software can be moved from older system infrastructures to current system infrastructures without any issues.
6. **Functional testing :** Also known as functional completeness testing, functional testing involves trying to think of any possible missing functions. Testers might make a list of additional functionalities that a product could have to improve it during functional testing.
7. **Hardware/Software testing :** IBM refers to Hardware/Software testing as "HW/SW Testing". This is when the tester focuses his/her attention

on the interactions between the hardware and software during system testing.

Que 4.7. **What is acceptance testing ? Discuss the acceptance criteria used in acceptance testing.**

Answer

Acceptance testing : Refer Q. 4.1, Page 108E, Unit-4.

Acceptance criteria : One or more acceptance criteria are used for directing the creation of test cases in acceptance testing.

These criteria are as follows :

1. **Acceptance criteria-product acceptance :**
   a. During the requirements phase, each requirement is associated with acceptance criteria.
   b. It is possible that one or more requirements may be mapped to form acceptance criteria (for example, all high priority requirements should pass 100%).
   c. Whenever there are changes to requirements, the acceptance criteria are accordingly modified and maintained.
   d. Acceptance testing is not meant for executing test cases that have not been executed before.
   e. Hence, the existing test cases are looked at and certain categories of test cases can be grouped to form acceptance criteria.
2. **Acceptance criteria-procedure acceptance :**
   a. Acceptance criteria can be defined based on the procedures followed for delivery. An example of procedure acceptances could be documentation and release media.
   b. The nature of acceptance criteria are as follows :
      i. User, administration and troubleshooting documentation should be part of the release.
      ii. Along with binary code, the source code of the product with build scripts to be delivered in a CD.
      iii. A minimum of 20 employees are trained on the product usage prior to deployment.
3. **Acceptance criteria-service level agreement :**
   a. Service level agreements (SLA) can become part of acceptance criteria.
   b. They are generally part of a contract signed by the customer and product-organization.
   c. For example, time limits to resolve those defects can be mentioned part of SLA such as :

Join Telegram Group :- @AKTU_Group Scanned by TapScanner

Join Telegram Group :- @AKTU_Group www.aktutor.in

i. All major defects that come up during first three months of deployment need to be fixed free of cost.

ii. Downtime of the implemented system should be less than 0.1%.

iii. All major defects are to be fixed within 48 hours of reporting.

Que 4.8. Describe the procedure of selecting test cases for acceptance testing and execution of the test cases in acceptance testing.

Answer

Selecting test cases for acceptance testing : The test cases for acceptance testing are selected from the existing set of test cases from different phases of testing. For doing so, some guidelines are as follows :

1. **End-to-end functionality verification :** Test cases that include the end-to-end functionality of the product are taken up for acceptance testing. This ensures that all the business transactions are tested as a whole.
2. **Domain tests :** Since acceptance tests focus on business scenarios, the product domain tests are included. Test cases that reflect business domain knowledge are included.
3. **User scenario tests :** Acceptance tests reflect the real-life user scenario verification. So, test cases that portray them are included.
4. **Basic sanity tests :** Tests that verify the basic existing behaviour of the product are included.
5. **New functionality :** When the product undergoes modifications or changes, the acceptance test cases focus on verifying the new features.
6. **A few non-functional tests :** Some non-functional tests are included and executed as part of acceptance testing to double-check that the non-functional aspects of the product meet the expectations.
7. **Test pertaining to legal obligations and service level agreements :** Tests that are written to check if the product compiles with certain legal obligations and SLAs are included in the acceptance test criteria.
8. **Acceptance test data :** Test cases that make use of customer real-life data are included for acceptance testing.

Executing acceptance tests :

1. Sometimes the customers themselves do the acceptance tests. In such cases, the job of the product organization is to assist the customers in acceptance testing and resolve the issues that come out of it.
2. If the acceptance testing is done by the product organization, forming the acceptance test team becomes an important activity.

3. An acceptance test team usually comprises members who are involved in the day to day activities of the product usage or are familiar with such scenarios.
4. The product management, support and consulting team, who have good knowledge of the customers, contribute to the acceptance testing definition and execution.
5. The number of test team members needed to perform acceptance testing is very less, as the scope and effort involved in acceptance testing is not much when compared to other phases of testing.
6. Acceptance test team members may or may not be aware of testing or the process. Hence, before testing, appropriate training on the product and the process needs to be provided to the team.
7. There could also be in-house training material that could serve the purpose.
8. Also, test team members help the acceptance members to get the required test data, select and identify test cases and analyze the acceptance test team results.
9. During test execution, the acceptance test team reports its progress regularly. The defect reports are generated on a periodic basis.
10. Defects reported during acceptance tests could be of different priorities. Test teams help acceptance test team report defects.
11. A high priority defects are necessarily fixed before software is released.
12. In case major defects are identified during acceptance testing, then there is a risk of missing the release date.
13. When the defect fixes point to scope of requirement changes, then it may either result in the extension of the release date to include the feature in the current release or get postponed to subsequent releases.
14. All resolution of those defects is discussed with acceptance test team and their approval is obtained for concluding the completion of acceptance testing.

Que 4.9. What do you understand by debugging in software testing ?

OR

Discuss debugging process.

Answer

1. On successful culmination of software testing, debugging is performed.
2. Debugging is defined as a process of analyzing and removing the error.
3. It is considered necessary in most of the newly developed software or hardware and in commercial products/ personal application programs.
4. For complex products, debugging is done at all the levels of the testing.

Join Telegram Group :- @AKTU_Group Scanned by TapScanner

Join Telegram Group :- @AKTU_Group www.aktutor.in

5. Debugging is considered to be a complex and time consuming process since it attempts to remove errors at all the levels of testing.
6. To perform debugging, debugger (debugging tool) is used to reproduce the conditions in which failure occurred, examine the program state, and locate the cause.
7. With the help of debugger, programmers trace the program execution step by step (evaluating the value of variables) and halt the execution wherever required to reset the program variables.
8. Some guidelines that are followed while performing debugging are :
   a. Debugging is the process of solving a problem. Hence, individuals involved in debugging should understand all the causes of an error before starting with debugging.
   b. No experimentation should be done while performing debugging. The experimental changes instead of removing errors often increase the problem by adding new errors in it.
   c. When there is an error in one segment of a program, there is a high possibility that some other errors also exist in the program. Hence, if an error is found in one segment of a program, rest of the program should be properly examined for errors.
   d. It should be ensured that the new code added in a program to fix errors is correct and is not introducing any new error in it. Thus, to verify the correctness of a new code and to ensure that no new errors are introduced, regression testing should be performed.

Debugging process :

1. During debugging, errors are encountered that range from less damaging (like input of an incorrect function) to catastrophic (like system failure, which lead to economic or physical damage). With the increase in number of errors, the amount of effort to find their causes also increases.
2. Once errors are identified in a software system, to debug the problem, a number of steps are followed, which are :
   a. **Defect confirmation/identification :**
      i. A problem is identified in a system and a defect report is created.
      ii. A software engineer maintains and analyzes this error report and finds solutions to the following questions :
         1. Does a defect exist in the system ?
         2. Can the defect be reproduced ?
         3. What is the expected/desired behaviour of the system ?
         4. What is the actual behaviour ?
   b. **Defect analysis :**
      i. If the defect is genuine, the next step is to understand the root cause of the problem.

      ii. Generally, engineers debug by starting a debugging tool (debugger) and they try to understand the root cause of the problem by following a step-by-step execution of the program.
   c. **Defect resolution :**
      i. Once the root cause of a problem is identified, the error can be resolved by making an appropriate change to the system by fixing the problem.
      ii. When the debugging process ends, the software is retested to ensure that no errors are left undetected.
      iii. Moreover, it checks that no new errors are introduced in the software while making some changes to it during the debugging process.

Que 4.10. **What are the various techniques used in debugging ?**

OR

Discuss debugging strategies.

Answer

Techniques used for debugging / debugging strategies :

1. **Brute force :**
   a. Brute force method of debugging is the most commonly used but least efficient method.
   b. It is generally used when all other available methods fail.
   c. Here, debugging is done by taking memory (or storage) dumps.
   d. Actually, the program is loaded with the output statements that produce a large amount of information including the intermediate values.
   e. Analyzing this information may help to identify the errors cause.
   f. However, using a memory dump for finding errors requires analyzing huge amount of information or irrelevant data leading to waste of time and effort.
   g. This strategy is a 'disciplined thought process' where errors can be debugged by moving outwards from the particulars to the whole.
   h. This strategy assumes that once the symptoms of the error are identified, and the relationships between them are established, the errors can be easily detected by just looking at the symptoms and the relationships.
2. **Induction strategy :** To perform induction strategy, a number of steps are followed, which are :
   a. **Locating relevant data :** All the information about a program is collected to identify the functions, which are executed correctly and incorrectly.

Join Telegram Group :- @AKTU_Group Scanned by TapScanner

Join Telegram Group :- @AKTU_Group www.aktutor.in

b. **Organizing data :** The collected data is organized according to importance. The data can consist of possible symptoms of errors, their location in the program, the time at which the symptoms occur during the execution of the program and the effect of these symptoms on the program.

c. **Devising hypothesis :** The relationships among the symptoms are studied and a hypothesis is devised that provides the hints about the possible causes of errors.

d. **Proving hypothesis :** In the final step, the hypothesis needs to be proved. It is done by comparing the data in the hypothesis with the original data to ensure that the hypothesis explains the existence of hints completely. In case, the hypothesis is unable to explain the existence of hints, it is either incomplete or contains multiple errors in it.

3. **Deduction strategy :**

a. In this strategy, first step is to identify all the possible causes and then using the data each cause is analyzed and eliminated if it is found invalid.

b. In induction strategy, deduction strategy is also based on some assumptions.

c. To use this strategy following steps are followed :

i. **Identifying the possible causes or hypothesis :** A list of all the possible causes of errors is formed. Using this list, the available data can be easily structured and analyzed.

ii. **Eliminating possible causes using the data :** The list is examined to recognize the most probable cause of errors and the rest of the causes are deleted.

iii. **Refining the hypothesis :** By analyzing the possible causes one by one and looking for contradiction leads to elimination of invalid causes. This results in a refined hypothesis containing few specific possible causes.

iv. **Proving the hypothesis :** This step is similar to the fourth step in induction method.

4. **Backtracking strategy :**

a. This method is effectively used for locating errors in small programs.

b. According to this strategy, when an error has occurred, one needs to start tracing the program backward one step at a time evaluating the values of all variables until the cause of error is found.

c. This strategy is useful but in a large program with many thousands lines of code, the number of backward paths increases and becomes unmanageably large.

5. **Debugging by testing :**

a. This debugging method can be used in conjunction with debugging by induction and debugging by deduction methods.

b. Additional test cases are designed that help in obtaining information to devise and prove a hypothesis in induction method and to eliminate the invalid causes and refine the hypothesis in deduction method.

c. The test cases used in debugging are different from the test cases used in testing process.

d. Here, the test cases are specifically designed to explore the internal program state.

Que 4.11. **Write the difference between testing and debugging.**

Answer

S.No.	Testing	Debugging
1.	Testing always starts with known conditions, uses predefined methods, and has predictable outcomes too.	Debugging starts from possibly unknown initial conditions and its end cannot be predicted, apart from statistically.
2.	Testing can and should definitely be planned, designed, and scheduled.	The procedures for, and period of, debugging cannot be so constrained.
3.	It proves a programmers failure.	It is the programmer's vindication.
4.	It is a demonstration of error or apparent correctness.	It is always treated as a deductive process.
5.	Testing as executed should strive to be predictable, dull, constrained, rigid, and inhuman.	Debugging demands intuitive leaps, conjectures, experimentation, and some freedom also.
6.	Much of the testing can be done without design knowledge.	Debugging is impossible without detailed design knowledge.
7.	It can often be done by an outsider.	It must be done by an insider.
8.	Much of test execution and design can be automated.	Automated debugging is still a dream for programmers.
9.	Testing purpose is to find bug.	Debugging purpose is to find cause of bug.

Join Telegram Group :- @AKTU_Group Scanned by TapScanner

Join Telegram Group :- @AKTU_Group
www.aktutor.in

PART-2

Testing Techniques and Their Applicability and Exploratory Testing

CONCEPT OUTLINE : PART-2

- People-based techniques focus on who does the testing.
- Coverage-based techniques focus on what gets tested.
- Problems-based techniques focus on why you are testing.
- Activity-based techniques focus on how you test.
- Evaluation-based techniques forces on how to tell whether the test passed or failed.
- Goals of exploratory testing are :
  - a. To gain an understanding of how an application works, what its interface looks like, and what functicnality it implement.
  - b. To force the software to exhibit its capabilities.
  - c. To find bugs.

Questions-Answers

Long Answer Type and Medium Answer Type Questions

Que 4.12. **Give various testing techniques.**

OR

Classified the testing in different categories.

Answer

1. **People-based techniques focus on who does the testing :**
   - a. User testing
   - b. Alpha testing
   - c. Beta testing
   - d. Bug bashes
   - e. Subject-matter expert testing
   - f. Paired testing
2. **Coverage-based techniques focus on what gets tested :**
   - a. Function testing
   - b. Feature or function integration testing
   - c. Menu tour

   - d. Domain testing
   - e. Equivalence class analysis
   - f. Boundary testing
   - g. Best representative testing
   - h. Map and test all the ways to edit a field
   - i. Logic testing
   - j. State-based testing
   - k. Path testing
   - l. Specification-based testing
   - m. Requirements-based testing
   - n. Combination testing
3. **Problems-based techniques focus on why you're testing :**
   - a. Risk-based testing
4. **Activity-based techniques focus on how you test :**
   - a. Regression testing
   - b. Scripted testing
   - c. Smoke testing
   - d. Exploratory testing
   - e. Guerrilla testing
   - f. Scenario testing
   - g. Installation testing
   - h. Load testing
   - i. Long sequence testing
   - j. Performance testing
5. **Evaluation-based techniques focus on how to tell whether the test passed or failed :**
   - a. Self-verifying data
   - b. Comparison with saved results
   - c. Comparison with the specification or other authoritative document
   - d. Heuristic consistency
   - e. Oracle-based testing

Que 4.13. **Discuss exploratory testing. Describe various techniques used in exploratory testing.**

Join Telegram Group :- @AKTU_Group Scanned by TapScanner

Join Telegram Group :- @AKTU_Group www.aktutor.in

Answer

1. It is an ad-hoc testing which keep exploring the product, covering more depth and breadth. Exploratory testing tries to do that with specific objectives, tasks and plans. Exploratory testing can be done during any phase of testing.
2. Exploratory testers may execute their tests based on their past experiences in testing a similar product.
3. They also uses their past experience of finding defects in the previous product release and check if the same problems persist in the current version.
4. Exploratory testing can be used to test software that is untested, unknown, or unstable. It is used when it is not obvious what the next test should be or when we want to go beyond the obvious tests.
5. Exploring can happen not only for functionality but also for different environments, configuration parameters, test data and so on.
6. Since there is large creative element to exploratory testing, similar test cases may result in different kinds of defects when done by two different individuals.

Exploratory testing techniques : There are many ways of doing exploratory testing :

1. **Guesses :** Guesses are used to find the part of the program that is likely to have more errors. Previous experience on working with a similar product helps in guessing.
2. **Architecture diagrams, use cases :**
   a. Architecture diagrams depict the interactions and relationships between different components and modules.
   b. Use cases give an insight of the product's usage from the end user's perspective. Exploration technique may use these diagrams and use cases to test the product.
3. **Study of past defects :** Defect reports of the previous releases act as a pointer to explore an area of the product further.
4. **Error handling :**
   a. Error handling is a portion of the code which prints appropriate messages or actions in case of failures. Error handling provides a message or corrective action in failure situation.
   b. Tests can be performed to simulate such situations to ensure that the product code takes care of this aspect.
5. **Discussions :** Exploration may be planned based on the understanding of the system during project discussions or meetings. Plenty of

information can be picked up during these meeting regarding implementation of different requirements for the product.

6. **Questionnaires and Checklists :** Questions like, "What, When, How, and Why", can provide leads to explore areas in the product. Such questions will provide insights to what more can be tested.

Goals of exploratory testing :

1. **To gain an understanding of how an application works, what its interface looks like, and what functionality it implements :**
   a. Such a goal is often adopted by testers new to a project or those who want to identify test entry points, identify specific testing challenges, and write test plans.
   b. This is also the goal used by experienced testers as they explore an application to understand the depth of its testing needs and to find new unexplored functionality.
2. **To force the software to exhibit its capabilities :**
   a. The idea is to make the software work hard and to ask it hard questions that put it through its paces.
   b. This may or may not find bugs, but it will certainly provide evidence that the software performs the function for which it was designed and that it satisfies its requirements.
3. **To find bugs :**
   a. Exploring the edges of the application and hitting potential soft spots is a specialty of exploratory testing.
   b. The goal is purposeful, rather than aimless, exploration to identify untested and historically buggy functionality.

Advantages of exploratory testing :

1. It doesn't require preparation for testing as we don't have documents for testing.
2. In this type of testing time saves due to all tasks are doing simultaneously like testing, designing test scenarios and executing test scenarios.
3. Tester can report many issues due to incomplete requirement or missing requirement document.

Disadvantages of exploratory testing :

1. Few issues cannot be catch in this type of testing.
2. There is review of test planning and designing of test cases/scenario while testing may cause issues.
3. Testers have to remember the scenario what he is executing because if any bug is found then tester should report a bug with proper steps.

Join Telegram Group :- @AKTU_Group Scanned by TapScanner

Join Telegram Group :- @AKTU_Group www.aktutor.in

PART-3

Automated Test Data Generation : Test Data, Approaches to Test Data Generation, Test Data Generation using Genetic Algorithm, Test Data Generation Tools, Software Testing Tools, and Software Test Plan.

CONCEPT OUTLINE : PART-3

- Test data is actually the input given to a software program.
- Test data generators based on their approaches are typically classified into following :
  a. Random test data generators
  b. Pathwise data generators
  c. Goal-oriented generators
  d. Intelligent test data generators
- The tools are divided into different categories as follows :
  a. Test management tools
  b. Functional testing tools
  c. Load testing tools
- Software test plan typically contains a detailed understanding of the eventual workflow.

Questions-Answers

Long Answer Type and Medium Answer Type Questions

Que 4.14. **What is test data ? Why is it important ?**

OR

Write the types of test data.

OR

What are the important points keep in mind before preparing test data ?

Answer

1. Test data is data which has been specifically identified for use in tests, typically of a computer program.
2. Some data may be used in a confirmatory way, typically to verify that a given set of input to a given function produces some expected result.
3. Other data may be used in order to challenge the ability of the program to respond to unusual, extreme, exceptional, or unexpected input.
4. Test data may be produced in a focused or systematic way (as is typically the case in domain testing), or by using other, less-focused approaches (as is typically the case in high-volume randomized automated tests).

5. Test data may be produced by the tester, or by a program or function that aids the tester.
6. Test data may be recorded for re-use, or used once and then forgotten.
7. Test data is actually the input given to a software program.
8. It represents data that affects or is affected by the execution of the specific module.
9. Some data may be used for positive testing, typically to verify that a given set of input to a given function produces an expected result.
10. Other data may be used for negative testing to test the ability of the program to handle unusual, extreme, exceptional, or unexpected input.
11. Poorly designed testing data may not test all possible test scenarios which will hamper the quality of the software.

Limitations of test data :

1. It is really difficult to create sufficient test data for testing.
2. The quantity of an efficiency data to be tested is determined or limited by time, cost and quality.

Types of test data : Test data can be classified into following type :

1. **No data/Blank file :** Refers to those files which do not have any data *i.e.* no input is given to the application and this verifies that application handles such exceptions and throws proper error.
2. **Valid set of test data :** Refers to the valid or supported files by the application. These should give the expected output when given as input.
3. **Invalid set of test data :** Refers to all the unsupported file formats in order to see that application handles all of them properly without breaking and warns user with proper error message.
4. **Huge test data :** For load, performance and stress, testing cannot be made at the time of execution and should be prepared while making your test environment ready.
5. **Test data :** To check all the boundary conditions includes data which has all possible combinations of boundary values. For example, if a text box can have number 2-20 then input 2 (minimum) and then 20 (maximum) values.

Ideal test data is the one which has all the combinations of data so that no major defects are missed.

Important point while creating test data :

1. Always make sure that test data files are not corrupted. This can leads to invalid output and might miss important defects as well.
2. Test data should be updated on a regular basis. This will give a clear picture of expected output.
3. Test data should be created before test cases execution to save time and meet deadline.

Join Telegram Group :- @AKTU_Group Scanned by TapScanner

Join Telegram Group :- @AKTU_Group www.aktutor.in

4. It is a good practice to use some automation tool to create huge amount of test data as manual effort in creating such data would be more and also it will be time consuming.
5. Test data should have invalid inputs to test negative scenarios.
6. Tester can take developer's help to create test data.
7. It is always a better practice to include all possible combinations of supported and unsupported formats in test data to ensure that test coverage is maximum.

Que 4.15. Explain the various approaches of test data generation.

Answer

1. Test data generation is the process of creating a set of data for testing the adequacy of new or revised software applications.
2. It may be the actual data that has been taken from previous operations or artificial data created for this purpose.
3. Test data generation is seen to be a complex problem and though a lot of solutions have come forth most of them are limited to toy programs.

Test data generators based on their approaches are typically classified into :

1. Random test data generators
2. Pathwise data generators
3. Goal-oriented generators
4. Intelligent test data generators

1. Random test data generators :

a. Random test data generation is probably the simplest method for generation of test data.

b. The advantage of this is that it can be used to generate input for any type of program.

c. Thus, to generate test data we can randomly generate a bit stream and let it represent the data type needed.

d. However, random test data generation does not generate quality test data as it does not perform well in terms of coverage.

e. Since the data generated is based solely on probability, it cannot accomplish high coverage as the chances of it finding semantically small faults is quite low.

2. Pathwise test data generators :

a. Pathwise test data generation is considered to be one of the best approaches to test data generation.

b. This approach does not give the generator the choice of selecting between multiple paths but just gives it one specific path for it to work on. Hence, it is named as pathwise test data generator.

c. Thus, except for the fact that this method uses specific paths, it is quite similar to goal-oriented test data generation.

d. The use of specific paths leads to a better knowledge and prediction of coverage.

e. However, this also makes it harder to generate the needed test data.

f. Pathwise test data generators require two inputs from the user
   i. The program to be tested.
   ii. Testing criterion (example, path coverage, statement coverage etc.).

3. Goal-oriented test data generators :

a. The goal-oriented approach provides guidance towards a certain set of paths.

b. The test data generators in this approach generate an input for any path instead of the usual approach of generating input from the entry to the exit of a block of code.

c. Thus, the generator can find any input for any path p which is a subset of the path u.

d. This drastically reduces the risk of generating relatively infeasible paths and provides a way to direct the search.

e. Two methods follow this technique :
   i. The Chaining approach
   ii. Assertion-oriented approach.

i. Chaining approach :

1. The chaining approach is an extension of the goal-oriented approach.
2. It is seen that the main limitation of the test data generation methods is that only the control flow graph is used to generate the test data.
3. This limited knowledge may make our selection harder.
4. Thus, it is seen that the path-oriented approach usually has to generate a large number of paths before it finds the "right" path.
5. This is because the path selection is blind.
6. The chaining approach tries to identify a chain of nodes that are vital to the execution of the goal node.

ii. Assertion oriented approach :

1. The assertion oriented approach is an extension of the chaining approach.
2. In this approach, assertions - that is constraint conditions are inserted.
3. This can be done either manually or automatically.
4. If the program does not hold on execution there is an error in the program or the assertion.
5. When an assertion is executed it must hold, otherwise there is an error either in the program or in the assertion.

Join Telegram Group :- @AKTU_Group Scanned by TapScanner

Join Telegram Group :- @AKTU_Group www.aktutor.in

4. **Intelligent test data generators :**
   a. Intelligent test data generators depend on sophisticated analysis of the code to guide the search of the test data.
   b. Intelligent test data generators are essentially utilizing one of the test data generation method coupled with the detailed analysis of the code.
   c. This approach may generate test data quicker than the other approaches but the analysis required for the utilization of this approach over a wide variety of programs is quite complex and requires a great deal of insight to anticipate the different situations that may arise.

Que 4.16. **What are the benefits of automated test data generation?**

Answer

Automated test data generation has the following benefits :

1. **Fast :** If the number of test data items is large, automated test data generation may be substantially faster than creating the same test data by hand.
2. **Accurate :** Since test data generation is a tedious task, you are likely to make mistakes. Test data generated by a tool can be much more accurate.
3. **Can use built-in algorithms :**
   a. The test data generator may implement algorithms to generate correctly formatted special data like postal codes and e-mail addresses.
   b. Therefore, you need not look up the algorithms for generating the special test data yourself.
4. **Create both valid and invalid test data :**
   a. Just as we can create valid test data, we can also create invalid test data by specifying incorrect constraints.
   b. Further, by specifying the quantity of the valid and invalid data, we can control the percentage distribution between valid and invalid data.
5. **Data exportable to multiple formats :** The automated test data generators usually provide options to either generate the test data directly in the desired format (example, Excel, or SQL) or export to the desired format.
6. **Create a professional look :**
   a. Since automated test data generators use their own databases to source data, the test data added/ edited in our application looks professional.
   b. This way, we can avoid test data that looks odd (example, Name as aa or ababl2).
   c. This is important in a way because the test data used by us is stored in our application and reported in our bug reports.

   d. If your client or management looks at your bug reports, you would feel better if the test data looks good.

Que 4.17. **Discuss test data generation using genetic algorithm.**

Answer

Fig. 4.17.1 shows the schematic representation of test data generation using GA.

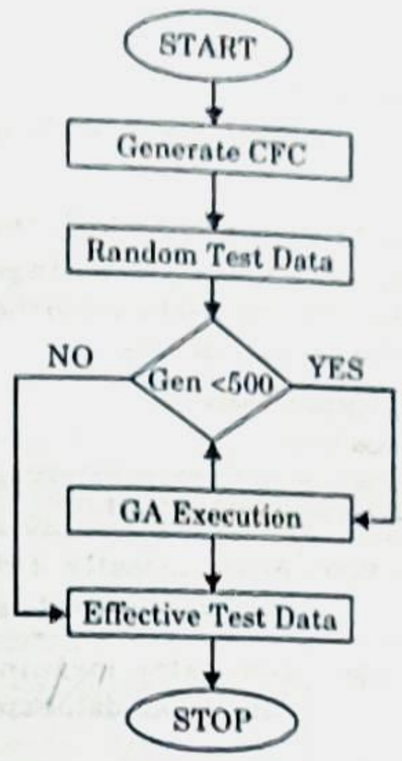


Fig. 4.17.1.

Algorithm :

Input : Randomly generated numbers (initial population act as test data) based on the target path to be covered.

Output : Test data for the target path.

1. Gen = 0
2. While Gen < 500
3. do
4. Evaluate the fitness value of each chromosome based on the objective function.
5. Use Elitism as selection operator, to select the individuals to enter into the mating pool.
6. Perform two-point cross over on the individuals in the mating pool, to generate the new population.
7. Perform bitwise mutation on chromosomes of the new population.

Join Telegram Group :- @AKTU_Group Scanned by TapScanner

Join Telegram Group :- @AKTU_Group www.aktutor.in

8. Gen = Gen+1
9. go to Step 3
10. end
11. Select the chromosome having the best fitness value as the desired result (test data for target path).

Que 4.18. **What is a test data generation tool ? List the various test data generation tools.**

Answer

Test data generation tool :

1. Testing a data-aware application is one of the most important but time consuming tasks.
2. It is important to test your application with "real" data.
3. To fill your database with test data, you need a generator. The generators will generate realistic data for you based on the column characteristics and/or based on what the user defines.

List of test data generation tools :
Some tools are mentioned here :

Product	Product	Kind of tool	Databases
DTM Data Generator	SQL Edit	Automatically fills a database with test data.	SQL Server, DB2, Oracle
GS Data Generator	GSApps	Generates meaningful data for our database.	SQL Server, DB2, Oracle, MS Access
Advanced Data Generator	Upscene Productions	It can generate real-life-like data into our database, SQL script or CSV files.	InterBase, Firebird, My SQL
SQL Data Generator	Red-Gate	Create realistic data based on column and table names.	MS SQL Server
EMS Data Generator	EMS	This utility can help you simulating the database production environment and allows you to populate several database tables with test data. Multiple editions, one for each supported database.	Oracle, MySQL, MS SQL, Postgre SQL, DB2, Firebird.

Datanamic Data Generator MultiDB	Datanamic Solution BV	Generates meaningful, realistic test data based on column characteristics. MultiDB edition supports data generation for 5 database types.	Oracle, MySQL, MS SQL Server, MS Access and Postgre SQL
IBM DB2 Test Database Generator	IBM	Creates realistic test data for your database application development projects. Only for DB2.	DB2
E-Naxos DataGen	E-Naxos	Mainly focused on generating random data. A free online version is also available.	Exports insert scripts for your database.

Que 4.19. **List the various software testing tools.**

Answer

1. Selection of tools is totally based on the project requirements and commercial (Proprietary/Commercial tools) or free tools (Open source tools).
2. Free testing tools may have some limitation in the features, so it is totally based on requirement fulfilled in free version or go for paid software testing tools.
3. The tools are divided into different categories as follows :
   a. Test Management Tools
   b. Functional Testing Tools
   c. Load Testing Tools

Open source tools :

1. Test management tools :
   a. **TET (Test Environment Toolkit) :**
      i. The goal behind creating the Test Environment Toolkit (TET) was to produce a test driver that accommodated the current and anticipated future testing needs of the test development community.
      ii. To achieve this goal, input from a wide sample of the community was used for the specification and development of TET's functionality and interfaces.
   b. **TETware :**
      i. The TETware is the Test Execution Management Systems which allows you to do the test administration, sequencing of

Join Telegram Group :- @AKTU_Group Scanned by TapScanner

Join Telegram Group :- @AKTU_Group www.aktutor.in

test, reporting of the test result in the standard format (IEEE Std 1003.3 1991) and this tool supports both UNIX as well as 32-bit Microsoft Windows operating systems, so portability of this is with test cases you developed.

ii. The TETware tools allow testers to work on a single, standard test harness, which helps you to deliver software projects on time.

c. **Test Manager :**

i. The test manager is an automated software testing tool and is used in day to day testing activities.

ii. The Java programming language is used to develop this tool.

iii. Such test management tools are used to facilitate regular software development activities, automate & manage the testing activities.

d. **RTH :**

i. RTH is called as "Requirements and Testing Hub".

ii. This is a open source test management tool where you can use a requirement management tool along with this. It also provides the bug tracking facilities.

2. **Functional testing tools :**

a. Selenium
b. Soapui
c. Watir
d. HTTP::Recorder
e. WatiN
f. Canoo WebTest
g. Webcorder
h. Solex
i. Imprimatur
j. SAMIE
k. Swete
l. ITP
m. WET
n. Webinject

3. **Load testing tools :**

a. Jmeter
b. FunkLoad

Proprietary/Commercial tools :

1. **Test management tools :**

a. HP Quality Center/ALM
b. QA Complete
c. T-Plan Professional
d. Automated Test Designer (ATD)
e. Testuff
f. SMARTS
g. QAS.TCS (Test Case Studio)
h. PractiTest
i. Test Manager Adaptors
j. SpiraTest
k. TestLog
l. ApTest Manager
m. DevTest

2. **Functional testing tools :**

a. QuickTest Pro
b. Rational Robot
c. Sahi
d. SoapTest
e. Badboy
f. Test Complete
g. QA Wizard
h. Netvantage Functional Tester
i. PesterCat
j. AppsWatch
k. Squish
l. actiWATE
m. liSA
n. vTest
o. Internet Macros
p. Ranorex

3. **Load testing tools ;**

a. WebLOAD Professional
b. HP LoadRunner
c. LoadStorm

Join Telegram Group :- @AKTU_Group Scanned by TapScanner

Join Telegram Group :- @AKTU_Group www.aktutor.in

d. NeoLoad
e. Loadtracer
f. Forecast
g. ANTS - Advanced .NET Testing System
h. vPerformer
i. Webserver Stress Tool
j. preVue-ASCII
k. Load Impact

Que 4.20. What is software test plan ? Explain major elements of test plan.

OR

Discuss test plan activities. Explain test plan structure.

Answer

Software test plan :

1. A test plan is a document detailing the objectives, target market, internal beta team, and processes for a specific beta test for a software or hardware product.
2. The plan typically contains a detailed understanding of the eventual workflow.
3. Test planning, the most important activity to ensure that there is initially a list of tasks and milestones in a baseline plan to track the progress of the project.
4. It also defines the size of the test effort.
5. It is the main document often called as master test plan or a project test plan and usually developed during the early phase of the project.
6. A test plan documents the strategy that will be used to verify and ensure that a product or system meets its design specifications and other requirements.
7. A test plan is usually prepared by or with significant input from test engineers.
8. Depending on the product and the responsibility of the organization to which the test plan applies, a test plan may include a strategy for one or more of the following :
   a. **Design Verification or Compliance test :** To be performed during the development or approval stages of the product, typically on a small sample of units.
   b. **Manufacturing or Production test :** To be performed during preparation or assembly of the product in an ongoing manner for purposes of performance verification and quality control.

   c. **Acceptance or Commissioning test :** To be performed at the time of delivery or installation of the product.
   d. **Service and Repair test :** To be performed as required over the service life of the product.
   e. **Regression test :** To be performed on an existing operational product, to verify that existing functionality did not get broken when other aspects of the environment are changed (example, upgrading the platform on which an existing application runs).

Major elements of test plan :

Test plan document formats can be as varied as the products and organizations to which they apply. There are three major elements that should be described in the test plan :

1. Test Coverage
2. Test Methods
3. Test Responsibilities

1. **Test coverage :**
   a. Test coverage in the test plan states what requirements will be verified during what stages of the product life.
   b. Test coverage is derived from design specifications and other requirements, such as safety standards or regulatory codes, where each requirement or specification of the design ideally will have one or more corresponding means of verification.
   c. Test coverage for different product life stages may overlap, but will not necessarily be exactly the same for all stages.
   d. For example, some requirements may be verified during design verification test, but not repeated during acceptance test.
   e. Test coverage also feeds back into the design process, since the product, may have to be designed to allow test access.
2. **Test methods :**
   a. Test methods in the test plan state how test coverage will be implemented.
   b. Test methods may be determined by standards, regulatory agencies, or contractual agreement, or may have to be created new.
   c. Test methods also specify test equipment to be used in the performance of the tests and establish pass/fail criteria.
   d. Test methods used to verify hardware design requirements can range from very simple steps, such as visual inspection, to elaborate test procedures that are documented separately.
3. **Test responsibilities :**
   a. Test responsibilities include what organizations will perform the test methods and at each stage of the product life.

Join Telegram Group :- @AKTU_Group Scanned by TapScanner

Join Telegram Group :- @AKTU_Group www.aktutor.in

b. This allows test organizations to plan, acquire or develop test equipment and other resources necessary to implement the test methods for which they are responsible.

c. Test responsibilities also includes, what data will be collected, and how that data will be stored and reported (often referred to as "deliverables").

d. One outcome of a successful test plan should be a record or report of the verification of all design specifications and requirements as agreed upon by all parties.

Test planning activities :

1. To determine the scope and the risks that need to be tested and that are not to be tested.
2. Documenting test strategy.
3. Making sure that the testing activities have been included.
4. Deciding entry and exit criteria.
5. Evaluating the test estimate.
6. Planning when and how to test and deciding how the test results will be evaluated, and defining test exit criterion.
7. The test artifacts delivered as part of test execution.
8. Defining the management information, including the metrics required and defect resolution and risk issues.
9. Ensuring that the test documentation generates repeatable test assets.

Test plan structure :

S.No.	Parameter	Description
1.	Test plan identifier	Unique identifying reference.
2.	Introduction	A brief introduction about the project and to the document.
3.	Test items	A test item is a software item that is the application under test.
4.	Features to be tested	A feature that needs to tested on the testware.
5.	Features not to be tested	Identify the features and the reasons for not including as part of testing.
6.	Approach	Details about the overall approach to testing.
7.	Item pass/fail criteria	Documented whether a software item has passed or failed its test.
8.	Test deliverables	The deliverables that are delivered as part of the testing process, such as test plans, test specifications and test summary reports.

9.	Testing tasks	All tasks for planning and executing the testing.
10.	Environmental needs	Defining the environmental requirements such as hardware, software, OS, network configurations, tools required.
11.	Responsibilities	Lists the roles and responsibilities of the team members.
12.	Staffing and training needs	Captures the actual staffing requirements and any specific skills and training requirements.
13.	Schedule	States the important project delivery dates and key milestones.
14.	Risks and Mitigation	High-level project risks and assumptions and a mitigating plan for each identified risk.
15.	Approvals	Captures all approvers of the document, their titles and the sign off date.

Que 4.21. **Explain software testing tools.**

Answer

There are three broad categories of software testing tools :

1. Static
2. Dynamic
3. Process management

1. **Static software testing tools :**

a. Static software testing tools are those that perform analysis of the program without executing them at all.

b. They may also find the source code which will be hard to test and maintain.

c. Static testing is about prevention and dynamic testing is about cure. Both tools are used but prevention is always better than cure.

d. Static tools will find more bugs as compared to dynamic testing tools.

e. There are many areas for which effective static testing tools are available and they have show their results for the improvement of the quality of the software.

Join Telegram Group :- @AKTU_Group Scanned by TapScanner

Join Telegram Group :- @AKTU_Group www.aktutor.in

Types of static software testing tools :

i. **Complexity analysis tools :**

a. Complexity of a program play a very important role while determining its quality. A higher value of cyclomatic complexity may indicate poor design and risky implementation.

b. Complexity analysis tools may take the program as an input, process it and produce a complexity value as output.

c. This value may be an indicator of the quality of design and implementation.

ii. **Syntax and Semantic analysis tools :**

a. These tools find syntax and semantic errors. These tools are language dependent and may parse the source code, maintain a list of errors and provide implementation information.

iii. **Flow graph generator tools :** These tools are language dependent and convert it to flow graph. These tools assist us to understand the risky and poorly designed area of the source code.

iv. **Code comprehension tools :** These tools may help to understand unfamiliar source code.

v. **Code inspectors :**

a. Source code inspectors do the simple job of enforcing standard in a uniform way for many programs. They inspect the programs and force us to implement the guidelines of good programming practices.

b. These tools are simple and may find many critical and weak areas of the program. They may also suggest possible change in the source code for improvement.

2. **Dynamic software testing tools :**

a. Dynamic software testing tools select test cases and execute the program to get the result. They also analyze the result and find reasons for failures of the program.

b. They will be used after the implementation of the program and may also test non-functional requirements like efficiency, performance, reliability etc.

Types of dynamic software testing tools :

i. **Coverage analysis tools :**

a. These tools are used to find the level of coverage of the program after executing the selected test cases. They give an idea about the effectiveness of the selected test cases.

b. They highlight the unexecuted portion of the source code.

ii. **Performance testing tools :**

a. We may like to test the performance of the software under stress/load.

b. For example, if we are testing a result management software, we may observe the performance when 10 users are entering the data and also when 100 users are entering the data simultaneously.

c. Similarly, we may like to test a website with 10 users, 100 users, 1000 users etc. working simultaneously.

d. This may require huge resources and sometimes, it may not be possible to create such real life environment for testing in the company.

e. A tool may help us to simulate such situations and test there situations in various stress conditions.

f. Some of the popular tools are Mercury Interactive's Load Runner, Apache's I Meter, Segue Software's Silk Performer, IBM Rational's performance Tester, Comuware's QALOAD and Autotester's AutoController.

iii. **Fuctional/Regression testing tools :**

a. These tools are used to test the software on the basis of its functionality without considering the implementation details.

b. They may also generate test cases automatically and execute them without human intervention.

c. Some of the popular available tools are IBM Rational's Robot, Mercury Interactive's Win Runner, Compuware's QA center

3. **Process management tools :**

a. These tools help us to manage and improve the software testing process.

b. We may create a test plan, allocate resources and prepare a schedule for unattended testing for tracking the status of a bug using such tools.

c. They improve many aspects of testing and make it a disciplined process.

d. Some of the tools are IBM Rational Test Manager, Mercury Interactive's Test Director, Segue Software's Silk plan Pro and Compuware's QA Director.

☺☺☺

Join Telegram Group :- @AKTU_Group Scanned by TapScanner

Join Telegram Group :- @AKTU_Group www.aktutor.in

5 UNIT

Object-Oriented Testing

Part-1 ..... (147E - 165E)

- *Object-Oriented Testing : Definition, Issues*
- *Class Testing, Object-Oriented Integration and System Testing*
- *Testing Web Applications : What is Web Testing ?*
- *User Interface Testing*
- *Usability Testing*

A. Concept Outline : Part-1 ..... 147E
B. Long and Medium Answer Type Questions 147E

Part-2 ..... (165E - 180E)

- *Software Testing*
- *Performance Testing*
- *Database Testing*
- *Post Deployment Testing*

A. Concept Outline : Part-2 ..... 165E
B. Long and Medium Answer Type Questions 166E

PART-1

Object-Oriented Testing : Definition, Issues, Class Testing, Object-Oriented Integration and System Testing, Testing Web Application : What is Web Testing ?, User Interface Testing, Usability Testing.

CONCEPT OUTLINE : PART-1

- Object-oriented testing begins in the small with a series of tests designed to exercise class operations and check whether errors exist as one class collaborates with other classes.
- Various issues involved in object-oriented testing are :
  1. Decentralized code
  2. Encapsulation / Data abstraction
  3. Inheritance
  4. Polymorphism
  5. Abstract classes
- Web testing is the name given to software testing that focuses on testing the web applications.
- Usability testing is a technique used in user-centered interaction design to evaluate a product by testing it on users.

Questions-Answers

Long Answer Type and Medium Answer Type Questions

Que 5.1. **What is object-oriented testing ?**

OR

Which types of tools are used for testing object-oriented system ?

Answer

1. Object-oriented testing begins in the small with a series of tests designed to exercise class operations and check whether errors exist as one class collaborates with other classes.
2. Use based testing along with fault based testing is applied to integrated classes.
3. In the end, use cases are used to uncover errors at software validation level.
4. Encapsulation can create a minor problem while testing because testing reports on concrete and abstract state of an object.
5. Multiple inheritance complicates testing by increasing number of contexts for which testing is required.

Join Telegram Group :- @AKTU_Group Scanned by TapScanner

Join Telegram Group :- @AKTU_Group www.aktutor.in

Testing OO systems broadly covers the following topics :

1. Unit testing of a class
2. Integration testing of classes
3. System testing
4. Regression testing
5. Tools for testing OO system

1. Unit testing a set of classes :

a. The base of object-oriented system is "Class". A class is a representation of a real-life object.

b. Each class is made up of attributes or variables and methods that operate on the variables.

c. As a class is build before it is "published" for use by others, it has to be tested to see if it is ready for use.

d. The unit test required for the building blocks (classes) is necessary for the following reasons :

i. A class is intended of heavy reuse. A residual defect in a class can affect every instance of reuse.

ii. Many defects get introduced at the time a class gets defined. A delay in catching these defects makes them go into the client of these classes.

iii. A class is a combination of data and methods. If the data and methods do not work in sync at a unit test level, it may cause defects that are difficult to find later.

iv. An OO system has building blocks like procedural language and also has special features like inheritance. Thus, unless the building blocks are thoroughly tested stand alone, defects arising out of these contexts may surface, magnified many times.

e. Some conventional methods for unit testing can be applied on classes as following :

i. Every class has certain variable. A boundary value analysis and equivalence partitioning can be applied to make sure the most effective test data is used to find as many defects as possible.

ii. The methods of function coverage of white box testing can be used to ensure that every method is exercised.

iii. The techniques of condition coverage, branch coverage, code complexity of white box testing can be used to make sure as many branches and conditions are covered as possible and to increase the maintainability of the code.

iv. System and acceptance testing can be performed for early detection of stress-related problems such as memory leaks.

2. Integration testing :

a. The various methods of integration like top-down, bottom-up, big bang and so on can all be applicable in OO system.

b. The extra points to be noted about integration testing OO system are that :

i. OO systems are inherently meant to be built out of small, reusable components. Hence, integration testing will be even more critical for OO systems.

ii. There is typically more parallelism in the development of the underlying components of OO system; thus the need for frequent integration is higher.

iii. Given the parallelism in development, the sequence of availability of the classes will have to be taken into consideration while performing integration testing. This would also require the design of stubs and harnesses to simulate the function of yet-unavailable classes.

3. **System testing and interoperability of OO system :** The system testing is required in OO system for the following reasons :

a. A class may have different parts, not all of which are used at the same time. When different clients start using a class, they may be using different parts of a class and this may introduce defects at a later (systems testing) phase.

b. Different classes may be combined together by a client and this combination may lead to new defects that are needed to be uncovered.

c. An instantiated object may not free all its allocated resources, thus causing memory leaks and such related problems, which will show up only in the system testing phase.

4. Regression testing of OO systems :

a. After integration testing, regression testing becomes very crucial for OO systems.

b. As a result of the heavy reliance of OO systems on reusable components, changes to any one component could have potentially unintended side-effects on the clients that use the component.

c. Hence, frequent integration and regression runs become very essential for testing OO systems.

d. Also, because of the cascaded effects of changes resulting from properties like inheritance, it makes sense to catch the defects as early as possible.

5. **Tools for testing of OO systems :** There are several tools that aid in testing OO systems some of which are :

Join Telegram Group :- @AKTU_Group Scanned by TapScanner

Join Telegram Group :- @AKTU_Group www.aktutor.in

a. **Use case :**

i. Use case represents the various tasks that a user will perform when interacting with the system.

ii. This fits in place for the object-oriented paradigm, as the tasks and responses are akin to messages, passed to the various objects.

b. **Class diagrams :** Class diagrams represent the different entities and the relationship that exists among the entities.

It is useful for testing in several ways :

i. It identifies the elements of a class and hence enables the identification of boundary value analysis, equivalence portioning and such tests.

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

c. **Sequence diagrams :** A sequence diagram represents a sequence of messages passed among objects to accomplish a given application scenario or used case. It helps to :

i. Identify temporal end-to-end messages.

ii. Tracing the intermediate points in an end-to-end transaction, thereby enabling easier narrowing down of problems.

d. **Activity diagram :** An activity diagram depicts the sequence of activities that take place. It is helpful for :

i. The ability to derive various paths through execution, similar to the flow graph, an activity diagram can be used to arrive at the code complexity and independent paths through a program code.

ii. Ability to identify the possible message flow between an activity and an object, thereby making the message-based testing more robust and effective.

Que 5.2. **Explain conventional methods that are applied to test class in object-oriented testing.**

Answer

Conventional methods that are applied to testing class are :

1. Every class has certain variables. The techniques of boundary value analysis and equivalence partitioning can be applied to make sure that most effective test data is used to find as many defects as possible.
2. Every class will have methods that have procedural logic. The technique of condition coverage, branch coverage, code complexity testing can be used to make sure as many branches and conditions are covered as possible and to increase the maintainability of the code.

3. Since a class is meant to be instantiated multiple time by different clients, the various technique of stress testing, system and acceptance testing, can be performed for early detection of stress related problems such as memory leaks.

a. A class is a combination of data and methods that operate on the data, in some cases, it can be visualized as an object going through different states.

b. The message that are passed to the class act as inputs to trigger the state transition.

c. It is useful to capture this view of class during the design phase, so that testing can be more natural.

d. Some of the criteria that can be used for testing are :

i. Is every state reached at least once ?

ii. Is every message (that is, input that cause a state transition) generated and tested ?

iii. Is every state transition achieved at least once ?

iv. Are illegal state transitions tested ?

Que 5.3. **Explain various issues involved in object-oriented testing.**

Answer

1. When obstacles to testing OO systems are discussed, the problem that the OO specific features of inheritance, encapsulation and polymorphism create is usually the focus.
2. "The combination of polymorphism, inheritance and encapsulation are unique to object-oriented languages, presenting opportunities for error that do not exist in conventional languages".
3. Some other complexities of OO systems are also decentralized code, test case identification and raising exceptions.

1. **Decentralized code :**

a. With procedural systems where all the functions and procedures are centralized, locating the source of a bug does not require us to look outside the walls of the program.

b. The same functionality that was in a single procedural program is broken out into many smaller classes in an OO system.

c. It can be argued that smaller classes ease the process of locating the source of a bug because smaller chunks of code are examined.

d. However, the side effect is that the code is no longer centralized so finding the source of a bug requires looking in many places

Join Telegram Group :- @AKTU_Group Scanned by TapScanner

Join Telegram Group :- @AKTU_Group www.aktutor.in

2. **Encapsulation :**
   a. "Encapsulation is a technique for enforcing information hiding where the interface and implementation of a program unit are syntactically separated".
   b. This enables the programmer to hide design decisions within the implementation and to narrow the possible interdependencies with other components by means of interface.
   c. If a programmer changes only the implementation of unit leaving the interface same then he needs to retest that unit and any units that explicitly depend on it.
   d. Therefore, if we modify the super class then it is necessary to retest all its subclasses because they depend on it in the sense that they inherit its methods.
3. **Data abstraction :**
   a. Data abstraction refers to the act of representing essential features without including the background details.
   b. Also due to data abstraction there is no visibility of the insight of objects.
   c. The data is not accessible to the outside world and only those functions which are wrapped in the class can access it.
   d. This data hiding makes it difficult for the tester to check what happens inside an object during testing.
4. **Inheritance :**
   a. Inheritance is one of the primary strengths of object-oriented programming. "Inheritance means properties defined for a class are inherited by its subclasses, unless it is otherwise stated".
   b. So, actually it provides the idea of reusability. However, method that is tested to be "correct" in the context of the base class does not guaranteed that it will work "correctly" in the context of the derived class.
   c. Therefore, it is precisely because of inheritance that we find problems arising with respect to testing. It also complicates inter-class testing as multiple classes are coupled through inheritance.
5. **Polymorphism :**
   a. Polymorphism means the ability to assume more than one form, both in terms of data and operations.
   b. It is the capability of an operation exhibiting different behaviour in different instances.
   c. However, polymorphism results in lack of controllability as actual binding of object reference is not known till runtime.
   d. In program based testing as it can lead to messages sent to wrong object.

6. **Abstract classes :**
   a. Abstract class is the way to push up common implementation into a base class. Hence, adding new objects are easier, because a lot of the common interfaces may already be implemented.
   b. These classes are designed only to act as a base class. However, since their features are not fully implemented, these classes cannot be instantiated and thus pose challenges for execution base testing.
   c. Only classes derived from the abstract class can be easily tested, but errors can be present also in the super class *i.e.* abstract class.

Que 5.4. **Explain**

i. **State-based testing**
ii. **Fault-based testing**
iii. **Scenario-based testing**

OR

Discuss methods of object-oriented testing.

Answer

As many organizations are currently using or targeting to switch to the OO paradigm, the importance of OO software testing is increasing. The methods used for performing object-oriented testing are :

1. State-based testing
2. Fault-based testing
3. Scenario-based testing

1. **State-based testing :**
   a. State-based testing is used to verify whether the methods of a class are interacting properly with each other.
   b. This testing seeks to exercise the transitions among the states of objects based upon the identified inputs.
   c. For this testing, finite-state machine (FSM) or state-transition diagram representing the possible states of the object and how state transition occurs is built.
   d. In addition, state-based testing generates test cases, which check whether the method is able to change the state of object as expected.
   e. If any method of the class does not change the object state as expected, the method is said to contain errors.
   f. To perform state-based testing, a number of steps are followed, which are listed below :
      i. Derive a new class from an existing class with some additional features, which are used to examine and set the state of the object.

Join Telegram Group :- @AKTU_Group Scanned by TapScanner

Join Telegram Group :- @AKTU_Group www.aktutor.in

ii. Next, the test driver is written. This test driver contains a main program to create an object, send messages to set the state of the object, send messages to invoke methods of the class that is being tested and send messages to check the final state of the object.

iii. Finally, stubs are written. These stubs call the untested methods.

2. **Fault-based testing :**

a. Fault-based testing is used to determine or uncover a set of plausible faults.

b. In other words, the focus of tester in this testing is to detect the presence of possible faults.

c. Fault-based testing starts by examining the analysis and design models of OO software as these models may provide an idea of problems in the implementation of software.

d. With the knowledge of system under test and experience in the application domain, tester designs test cases where each test case targets to uncover some particular faults.

e. The effectiveness of this testing depends highly on tester experience in application domain and the system under test. This is because if he fails to perceive real faults in the system to be plausible, testing may leave many faults undetected.

f. However, examining analysis and design models may enable tester to detect large number of errors with less effort.

g. As testing only proves the existence and not the absence of errors, this testing approach is considered to be an effective method and hence is often used when security or safety of a system is to be tested.

h. Integration testing applied for OO software targets to uncover the possible faults in both operation calls and various types of messages (like a message sent to invoke an object).

i. These faults may be unexpected outputs, incorrect messages or operations, and incorrect invocation.

j. The faults can be recognized by determining the behaviour of all operations performed to invoke the methods of a class.

3. **Scenario-based testing :**

a. Scenario-based testing is used to detect errors that are caused due to incorrect specifications and improper interactions among various segments of the software.

b. Incorrect interactions often lead to incorrect outputs that can cause malfunctioning of some segments of the software.

c. The use of scenarios in testing is a common way of describing how a user might accomplish a task or achieve a goal within a specific context or environment.

d. Note that these scenarios are more context and user specific instead of being product-specific. Generally, the structure of a scenario includes the following points :

i. A condition under which the scenario runs.

ii. A goal to achieve, which can also be a name of the scenario.

iii. A set of steps of actions.

iv. An end condition at which the goal is achieved.

v. A possible set of extensions written as scenario fragments.

e. Scenario-based testing combines all the classes that support a use-case (scenarios are subset of use-cases) and executes a test case to test them.

f. Execution of all the test cases ensures that all methods in all the classes are executed at least once during testing.

g. However, testing all the objects (present in the classes combined together) collectively is difficult.

h. Thus, rather than testing all objects collectively, they are tested using either top-down or bottom-up integration approach.

i. This testing is considered to be the most effective method as scenarios can be organized in such a manner that the most likely scenarios are tested first with unusual or exceptional scenarios considered later in the testing process.

j. This satisfies a fundamental principle of testing that most testing effort should be devoted to those paths of the system that are mostly used.

Que 5.5. **How to develop test cases in object-oriented testing ?**

Answer

a. The methods used to design test cases in OO testing are based on the conventional methods.

b. However, these test cases should encompass special features so that they can be used in the object-oriented environment.

c. The points that should be noted while developing test cases in an object-oriented environment are listed below :

1. It should be explicitly specified with each test case which class it should test.

2. Purpose of each test case should be mentioned.

Join Telegram Group :- @AKTU_Group Scanned by TapScanner

Join Telegram Group :- @AKTU_Group www.aktutor.in

3. External conditions that should exist while conducting a test should be clearly stated with each test case.
4. All the states of object that is to be tested should be specified.
5. Instructions to understand and conduct the test cases should be provided with each test case.

Que 5.6. **Write the difference between testing of procedural and object-oriented software.**

Answer

S. No.	Test method	Procedural software	Object-oriented software
1.	Unit testing	Test individual, functionally cohesive operations.	Unit testing is really integration testing, test logically related operations and data.
2.	Integration testing	Test an integrated set of units (operations and common global data).	Object-oriented's unit testing. No more bugs related to common global data (though could have errors associated with global object and classes).
3.	Boundary value testing	Used on units or integrated units and systems.	Of limited value if using strongly typed object-oriented languages and proper data abstraction is used to restrict the values of attributes.
4.	Basis path testing	Generally performed on units.	Limited to operations of objects. Must address exception handling and concurrency issues (if applicable). Lowered complexity of object lessens the need for this.
5.	Equivalence and black box testing	Used on units, integrated units and systems.	Emphasized for object-oriented : objects are black boxes, equivalence classes are messages.

Que 5.7. **What is web testing ?**

Answer

1. Web testing is the name given to software testing that focuses on testing the web applications.
2. Web application testing, a software testing technique exclusively adopted to test the applications that are hosted on web in which the application interfaces and other functionalities are tested.
3. In web-based, application is completely tested before going to production environment. This stage of web testing find out the possible bugs in the system.

Web application testing checklist :

1. **Functionality testing :**
   a. In functional testing, we need to check that each component is functioning as expected or not, so it is also called as "Component Testing".
   b. Functional testing is to test the functionality of the software application. Basically, it is to check the basic functionality mentioned in the functional specification document.
   c. Also, check whether software application is meeting the user expectations. We can also say that checking the behaviour of the software application against test specification.
   d. In this, testing activities should include :
      i. Link testing
      ii. Web form testing
      iii. Cookies testing
      iv. Test HTML and CSS
2. **Usability testing :**
   a. This testing is to be carried out by testers to ensure that it cover all possible test cases which targeted audience of the web application are doing regularly.
   b. This would include :

      Navigation testing of the web site :
      i. All possible options like menus, links or buttons on web pages should be visible & accessible from all the web pages.
      ii. Web pages navigation should be easy to use.
      iii. Help instruction content should be clear & should satisfy the purpose.
      iv. All options on header, footer & left/right navigation should be consistent throughout the pages.

Join Telegram Group :- @AKTU_Group Scanned by TapScanner

Join Telegram Group :- @AKTU_Group www.aktutor.in

Content testing of the web site :

i. No spelling or grammatical errors mistake in content throughout the page.

ii. All text should be present on images.

iii. No broken images.

iv. Its task is to validate all for UI testing.

v. Follow some standard on content building on web page.

vi. All content should be legible & easy to understand.

vii. Proper size images should be placed on web page.

3. **Compatibility testing :**

a. In software application testing, the compatibility testing is the non-functional part of testing. It is ensuring that how application's working in the supported environments.

b. Customers are using web application on different operating systems, browser compatibility, computing capacity of hardware platform, databases and bandwidth handling capacity of networking hardware.

c. The compatibility testing is to make sure that "Is web application show correctly across different devices ?"

d. This would include :

Browser compatibility test :

i. Web applications are rendering differently on different browsers and mobile browser.

ii. The objective of browser compatibility testing is to ensure that no error exist on the different web browsers while rendering the sites.

iii. In browser compatibility testing, we need to ensure that our web application is being displayed properly on different browsers.

OS compatibility :

i. In new technology newer graphics designs are used & different APIs are used which may not work on different operating systems.

ii. Also on rendering of different objects like text fields, buttons may display different on different operating system.

iii. So, testing of web application should be carried out on different OS like Windows, MAC, Solaris, Unix, Linux with different OS flavours.

4. **Database testing :**

a. Data reliability is key part in the database testing.

b. Testing activities would include :

i. Check if queries are executed without any errors.

ii. Creating, updating or deleting data in database should maintain the data integrity.

iii. More time should not be taken to execute the queries, if required, tune the queries for better performance.

iv. Check load on database while executing heavier queries & check the result.

v. Collect data from database & represent on the web pages correctly.

5. **Crowd testing :**

a. Crowd testing is when a large group of perfect strangers try product then give us phenomenally helpful feedback on usability, bugs and features.

b. It not limited to web applications, but for all kinds of applications including mobile application testing. Crowd testing is dependent on the quality of the crowd.

c. Also it depends on a crowd that is composed out of a large group of diver's people.

6. **Interface testing :**

a. In the interface testing mainly three areas should be covered : Web Server, Application Server and Database Server.

b. Ensure that all the communications between all these servers should be carried out correctly.

c. Verify that if connection between any servers is reset or lost then what is happening.

d. Check if any request interrupts in-between then how application is responding.

i. **Web server :** Check if all web requests are accepted and not any requests are denied or leaked.

ii. **Application server :** Check if request is sending correctly to any server and displayed correctly. Check if errors are cached properly and displayed to admin user.

iii. **Database server :** Check if database server returns correct result on query request.

7. **Performance testing :**

a. Performance testing is to check the web application is working under the heavy load. Performance testing is categorized into two parts : web stress testing, web load testing.

b. Testing activities will include but not limited to :

Join Telegram Group :- @AKTU_Group Scanned by TapScanner

Join Telegram Group :- @AKTU_Group www.aktutor.in

i. Website application response times at different connection speeds.

ii. Load test your web application to determine its behaviour under normal and peak loads.

iii. Stress test your web site to determine its break point when pushed to beyond normal loads at peak time.

iv. Test if a crash occurs due to peak load, how does the site recovers from such an event.

v. Make sure optimization techniques like zip compression, browser and server side cache enabled to reduce load times.

8. **Security testing :**

a. Security testing is to check whether how to store sensitive information such as credit cards etc.

b. Testing activities will include :

i. Test unauthorized access to secure pages should not be permitted.

ii. Restricted files should not be downloadable without appropriate access.

iii. Check sessions are automatically killed after prolonged user inactivity.

iv. On use of SSL certificates, website should re-direct to encrypted SSL pages.

Que 5.8. **What do you understand by user interface testing ?**

OR

What is graphical user interface testing ? What we check in GUI testing ?

OR

What approaches are used for graphical user interface testing ?

Answer

1. Graphical User Interface (GUI) testing is the process of testing the system's GUI of the system under test.
2. GUI testing is the process of ensuring proper functionality of the graphical user interface for a given application and making sure it conforms to its written specifications.
3. In addition to functionality, GUI testing evaluates design elements such as layout, colours, fonts, font sizes, labels, text boxes, text formatting, captions, buttons, lists, icons, links and content.
4. GUI testing processes can be either manual or automatic, and are often performed by third-party companies, rather than developers or end users.

5. **The following checklist will ensure detailed GUI testing :**

a. Check whether all the GUI elements for size, position, width, length and acceptance of characters or numbers. For instance, you must be able to provide inputs to the input fields.

b. Check whether you can execute the intended functionality of the application using the GUI.

c. Check whether error messages are displayed correctly.

d. Check for clear demarcation of different sections on screen.

e. Check whether font used in application is readable.

f. Check whether the alignment of the text is proper.

g. Check whether the colour of the font and warning messages is aesthetically pleasing.

h. Check that the images have good clarity.

i. Check that the images are properly aligned.

j. Check the positioning of GUI elements for different screen resolution.

Approaches of GUI testing :

GUI testing can be done through three ways :

1. **Manual based testing :** Under this approach, graphical screens are checked manually by testers in conformance with the requirements stated in the business requirements document.
2. **Record and Replay :** GUI testing can be done using automation tools. This is done in two parts. During record, test steps are captured by the automation tool. During playback, the recorded test steps are executed on the application under test.
3. **Model based testing :** A model is a graphical description of system's behaviour. It helps us to understand and predict the system behaviour. Models help in a generation of efficient test cases using the system requirements.

Que 5.9. **What is usability testing ? Explain various methods of its.**

Answer

1. Usability testing is a technique used in user-centred interaction design to evaluate a product by testing it on users.
2. This can be seen as an irreplaceable usability practice, since it gives direct input on how real users use the system.
3. This is in contrast with usability inspection methods where experts use different methods to evaluate a user interface without involving users.

Join Telegram Group :- @AKTU_Group Scanned by TapScanner

Join Telegram Group :- @AKTU_Group www.aktutor.in

4. Usability testing is a technique used to evaluate a product by testing it on users. Most people who set up a usability test carefully construct a scenario wherein a person performs a list of tasks that someone who is using the website for the first time is likely to perform.
5. Someone else observes and listens to the person who is performing the tasks while taking notes.
6. Watching someone perform common tasks on a website is a great way to test whether the site is usable because you will immediately be able to see whether they are able to perform the tasks and any difficulties they have while doing so.
7. Usability testing focuses on measuring a human-made product's capacity to meet its intended purpose.
8. Usability testing measures the usability, or ease of use, of a specific object or set of objects, whereas general human-computer interaction studies attempt to formulate universal principles.

Methods of usability testing :

Methods of usability testing are :

1. **Hallway testing :**
   a. Hallway testing is a quick, cheap method of usability testing in which randomly-selected people, example those passing by in the hallway are asked to try using the product or service.
   b. This can help designers to identify "brick walls", problems are so serious that users simply cannot advance, in the early stages of a new design.
   c. The idea behind hallway usability testing began as an alternative to hiring trained or certified personnel to test a particular software or technology product.
   d. The idea is that you can go out and grab random individuals passing by an office in a hallway and get them to test a product being developed.
   e. Another way to think of it is that random individuals are gathered from the street and then assembled in the hallway before having them test a product under development.
   f. Some experts believe that using hallway usability testing can reveal up to 95% of usability problems with a given interface or product.
   g. In a lot of ways, hallway usability testing is like developing a beta testing phase, where the product or interface is constrained to a random sample group before it is released to the public.
2. **Remote usability testing :**
   a. In a scenario where usability evaluators, developers and prospective users are located in different countries and time zones, conducting a traditional lab usability evaluation creates challenges both from the cost and logistical perspectives.
   b. These concerns led to research on remote usability evaluation, with the user and the evaluators separated over space and time.
   c. Remote testing, which facilitates evaluations being done in the context of the user's other tasks and technology, can be either synchronous or asynchronous.
   d. The former involves real time one-on-one communication between the evaluator and the user, while the latter involves the evaluator and user working separately.
   e. Synchronous usability testing methodologies involve video conferencing or employ remote application sharing tools such as WebEx.
   f. Asynchronous methodologies include automatic collection of user's click streams, user logs of critical incidents that occur while interacting with the application and subjective feedback on the interface by users.
3. **Expert review :**
   a. Expert review is another general method of usability testing.
   b. As the name suggests, this method relies on bringing in experts with experience in the field (possibly from companies that specialize in usability testing) to evaluate the usability of a product.
   c. A heuristic evaluation or usability audit is an evaluation of an interface by one or more human factors experts.
   d. Evaluators measure the usability, efficiency, and effectiveness of the interface based on usability principles, such as the ten usability heuristics originally defined by Jakob Nielsen.
   e. Nielsen's usability heuristics, which have continued to evolve in response to user research and new devices, include :
      i. Visibility of system status
      ii. Match between system and the real world
      iii User control and freedom
      iv. Consistency and standards
      v. Error prevention
      vi. Recognition rather than recall
      vii. Flexibility and efficiency of use
      viii. Aesthetic and minimalist design
      ix. Help users recognize, diagnose, and recover from errors
      x. Help and documentation

Join Telegram Group :- @AKTU_Group Scanned by TapScanner

Join Telegram Group :- @AKTU_Group www.aktutor.in

4. **Automated expert review :**
   a. Similar to expert reviews, automated expert reviews provide usability testing but through the use of programs given rules for good design and heuristics.
   b. Though an automated review might not provide as much detail and insight as reviews from people, they can be finished more quickly and consistently.
   c. The idea of creating surrogate users for usability testing is an ambitious direction for the artificial intelligence community.
5. **A/B testing :**
   a. In web development and marketing, A/B testing or split testing is an experimental approach to web design (especially user experience design), which aims to identify changes to web pages that increase or maximize an outcome of interest (example, click-through rate for a banner advertisement).
   b. As the name implies, two versions (A and B) are compared, which are identical except for one variation that might impact a user's behaviour.
   c. Version A might be the one currently used, while version B is modified in some respect.
   d. For instance, on an e-commerce website the purchase funnel is typically a good candidate for A/B testing, as even marginal improvements in drop-off rates can represent a significant gain in sales.
   e. Significant improvements can be seen through testing elements like copy text, layouts, images and colours.

Que 5.10. **What are the goals and advantages of usability testing?**

Answer

Goals of usability testing :

1. **Effectiveness of the system :**
   a. Is the system is easy to learn ?
   b. Is the system useful and adds value to the target audience ?
   c. Is content, colour, icons, images used are aesthetically pleasing ?
2. **Efficiency :**
   a. Navigation required to reach desired screen/webpage should be very less. Scroll bars should not be used frequently.
   b. Uniformity in the format of screen/pages in your application/website.
   c. Provision to search within your software application or website.

3. **Accuracy :**
   a. No outdated or incorrect data like contact information/address should be present.
   b. No broken links should be present.
4. **User friendliness :**
   a. Controls used should be self-explanatory and must not require training to operate.
   b. Help should be provided for the users to understand the application/website.
   c. Alignment with above goals helps in effective usability testing.

Advantages of usability testing :

1. Usability test can be modified to cover many other types of testing such as functional testing, system integration testing, unit testing, smoke testing etc.
2. Usability testing can be very economical if planned properly, yet highly effective and beneficial.
3. If proper resources (experienced and creative testers) are used, usability test can help in fixing all the problems that user may face even before the system is finally released to the user. This may result in better performance and a standard system.
4. Usability testing can help in discovering potential bugs and potholes in the system which generally are not visible to developers and even escape the other type of testing.

PART-2

Security Testing, Performance Testing, Database Testing, Post Deployment Testing.

CONCEPT OUTLINE : PART-2

- Security testing is a process intended to reveal flows in the security mechanisms of an information system that protect data and maintain functionality as intended.
- Performance testing measures the quality attributes of the system, such as scalability, reliability and resource usage.
- Types of database testing :
  a. Structural testing
  b. Functional testing
  c. Non-functional testing
- The purpose of post deployment testing is to ensure that the performance of the web site remains good.

Join Telegram Group :- @AKTU_Group Scanned by TapScanner

Join Telegram Group :- @AKTU_Group www.aktutor.in

Questions-Answers

Long Answer Type and Medium Answer Type Questions

Que 5.11. **What is security testing ? Why it is important ?**

OR

Explain various types of security testing.

Answer

a. Security testing is a process intended to reveal flaws in the security mechanisms of an information system that protect data and maintain functionality as intended.

b. Due to the logical limitations of security testing, passing security testing is not an indication that no flaws exist or that the system adequately satisfies the security requirements.

c. Security testing is the activity of assessing a system for the presence of security weaknesses.

d. Actual security requirements tested depend on the security requirements implemented by the system.

e. Security testing as a term has a number of different meanings and can be completed in a number of different ways.

f. Typical security requirements may include specific elements of confidentiality, integrity, authentication, availability, authorization, non-repudiation and availability.

g. The six basic security concepts are :

1. **Authentication :** The origin of the application and its data is genuine.
2. **Authorization :** Specific users should only get access to authorized functions.
3. **Confidentiality :** Data/information is secure from theft.
4. **Integrity :** The application and its data are not altered in course of time during transmission.
5. **Non repudiation :** Guarantee that sender and receiver of information cannot deny having sent or received the data.
6. **Availability :** Assuring information and communications services will be ready for use when expected. Information must be kept available to authorized persons when they need it.

Classification of security testing :

1. **Discovery :**
   a. The purpose of this is to identify systems within scope and the services in use.

   b. It is not intended to discover vulnerabilities, but version detection may highlight deprecated versions of software / firmware and thus indicate potential vulnerabilities.

2. **Vulnerability scan :**
   a. Following the discovery stage this looks for known security issues by using automated tools to match conditions with known vulnerabilities.
   b. The reported risk level is set automatically by the tool with no manual verification or interpretation by the test vendor.
   c. This can be supplemented with credential based scanning that looks to remove some common false positives by using supplied credentials to authenticate with a service.

3. **Vulnerability assessment :**
   a. This uses discovery and vulnerability scanning to identify security vulnerabilities and places the findings into the context of the environment under test.
   b. An example would be removing common false positives from the report and deciding risk levels that should be applied to each report finding to improve business understanding and context.

4. **Security assessment :**
   a. Builds upon vulnerability assessment by adding manual verification to confirm exposure, but does not include the exploitation of vulnerabilities to gain further access.
   b. Verification could be in the form of authorized access to a system to confirm system settings and involve examining logs, system responses, error messages, codes, etc.
   c. A security assessment is looking to gain a broad coverage of the systems under test but not the depth of exposure that a specific vulnerability could lead to.

5. **Penetration test :**
   a. Penetration test simulates an attack by a malicious party. Builds on the previous stages and involves exploitation of found vulnerabilities to gain further access.
   b. Using this approach will result in an understanding of the ability of an attacker to gain access to confidential information, affect data integrity or availability of a service and the respective impact.
   c. Each test is approached using a consistent and complete methodology in a way that allows the tester to use their problem solving abilities, the output from a range of tools and their own knowledge of networking and systems, to find vulnerabilities that would/ could not be identified by automated tools.
   d. This approach looks at the depth of attack as compared to the security assessment approach that looks at the broader coverage.

Join Telegram Group :- @AKTU_Group Scanned by TapScanner

Join Telegram Group :- @AKTU_Group www.aktutor.in

6. **Runtime testing :**
   a. Also referred to as dynamic testing and black box testing. This kind of test involves assessing the system for security issues from the perspective of an end user.
   b. The main difference between this and code review is that the tester does not have access to source code or other detailed knowledge of system internals.
   c. This is an accurate reflection of the kind of knowledge an external attacker has.
   d. Not having access to source code limits the tester's visibility into potential security issues.
   e. Because runtime tests are often time-limited in order to control costs, they may not accurately capture the kinds of attacks a dedicated adversary can find with more time.
7. **Security audit :**
   a. Driven by an audit / risk function to look at a specific control or compliance issue.
   b. Characterised by a narrow scope, this type of engagement could make use of any of the earlier approaches discussed (vulnerability assessment, security assessment, penetration test).
8. **Security review :**
   a. Verification that industry or internal security standards have been applied to system components or product.
   b. This is typically completed through gap analysis and utilizes build/code reviews or by reviewing design documents and architecture diagrams.
   c. This activity does not utilize any of the earlier approaches (Vulnerability Assessment, Security Assessment, Penetration Test, Security Audit).

Que 5.12. **Why we do performance testing ? What are the attributes of performance testing ?**

OR

What are the types of performance testing ?

Answer

1. Performance testing, a non-functional testing technique performed to determine the system parameters in terms of responsiveness and stability under various workload.
2. Performance testing measures the quality attributes of the system, such as scalability, reliability and resource usage.

3. Software performance testing is a means of quality assurance. It involves testing software applications to ensure they will perform well under their expected workload.

Attributes of performance testing :

1. **Speed :** Determines whether the application responds quickly.
2. **Scalability :** Determines maximum user load, the software application can handle.
3. **Stability :** Determines if the application is stable under varying loads.
4. **Reliability :** Reliability refers to the consistency of a measure.

Types of performance testing :

1. **Load testing :**
   a. Load testing is the simplest form of performance testing. A load test is usually conducted to understand the behaviour of the system under a specific expected load.
   b. This load can be the expected concurrent number of users on the application performing a specific number of transactions within the set duration.
   c. This test will give out the response times of all the important business critical transactions.
   d. The database, application server, etc. are also monitored during the test, this will assist in identifying bottlenecks in the application software and the hardware that the software is installed on.
2. **Stress testing :**
   a. Stress testing is normally used to understand the upper limits of capacity within the system.
   b. This kind of test is done to determine the system's robustness in terms of extreme load and helps application administrators to determine if the system will perform sufficiently if the current load goes well above the expected maximum.
3. **Soak testing :**
   a. Soak testing, also known as endurance testing, is usually done to determine if the system can sustain the continuous expected load.
   b. During soak tests, memory utilization is monitored to detect potential leaks.
   c. Also important, but often overlooked is performance degradation, i.e. to ensure that the throughput and/or response times after some long period of sustained activity are as good as or better than at the beginning of the test.
   d. It essentially involves applying a significant load to a system for an extended, significant period of time.

Join Telegram Group :- @AKTU_Group Scanned by TapScanner

Join Telegram Group :- @AKTU_Group www.aktutor.in

c. The goal is to discover how the system behaves under sustained use.

4. **Spike testing :**
   a. Spike testing is done by suddenly increasing the load generated by a very large number of users, and observing the behaviour of the system.
   b. The goal is to determine whether performance will suffer, the system will fail, or it will be able to handle dramatic changes in load.

5. **Configuration testing :**
   a. Rather than testing for performance from a load perspective, tests are created to determine the effects of configuration changes to the system's components on the system's performance and behaviour.
   b. A common example would be experimenting with different methods of load-balancing.

6. **Isolation testing :**
   a. Isolation testing is not unique to performance testing but involves repeating a test execution that resulted in a system problem.
   b. Such testing can often isolate and confirm the fault domain.

Que 5.13. **What are the parameters of performance testing ?**

Answer

1. **Processor usage :** Amount of time processor spends executing non-idle threads.
2. **Memory use :** Amount of physical memory available to processes on a computer.
3. **Disk time :** Amount of time disk is busy in executing a read or write request.
4. **Bandwidth :** Shows the bits per second used by a network interface.
5. **Private bytes :** Number of bytes a process has allocated that can not be shared amongst other processes. These are used to measure memory leaks and usage.
6. **Committed memory :** Amount of virtual memory used.
7. **Memory pages/second :** Number of pages written to or read from the disk in order to resolve hard page faults. Hard page faults are when code not from the current working set is called up from elsewhere and retrieved from a disk.
8. **Page faults/second :** The overall rate in which fault pages are processed by the processor. This again occurs when a process requires code from outside of its working set.

9. **CPU interrupts per second :** Is the average number of hardware interrupts a processor is receiving and processing each second.
10. **Disk queue length :** Is the average number of read and write requests queued for the selected disk during a sample interval.
11. **Network output queue length :** Length of the output packet queue in packets. Anything more than two means a delay and bottle necking needs to be stopped.
12. **Network bytes total per second :** Rate at which bytes are sent and received on the interface including framing characters.
13. **Response time :** Time from when a user enters a request until the first character of the response is received.
14. **Throughput :** Rate a computer or network receives requests per second.
15. **Amount of connection pooling :** The number of user requests that are met by pooled connections. The more requests met by connections in the pool, the better the performance will be.
16. **Maximum active sessions :** The maximum number of sessions that can be active at once.
17. **Hit ratios :** This has to do with the number of SQL statements that are handled by cached data instead of expensive I/O operations. This is a good place to start for solving bottlenecking issues.
18. **Hits per second :** The number of hits on a web server during each second of a load test.
19. **Rollback segment :** The amount of data that can rollback at any point in time.
20. **Database locks :** Locking of tables and databases needs to be monitored and carefully tuned.
21. **Top waits :** Are monitored to determine what wait times can be cut down when dealing with the how fast data is retrieved from memory.
22. **Thread counts :** An applications health can be measured by the number of threads that are running and currently active.
23. **Garbage collection :** Has to do with returning unused memory back to the system. Garbage collection needs to be monitored for efficiency.

Que 5.14. **Explain process of performance testing.**

Answer

Process of performance testing includes :

1. **Identify your testing environment :**
   a. Know your physical test environment, production environment and what testing tools are available.

Join Telegram Group :- @AKTU_Group Scanned by TapScanner

Join Telegram Group :- @AKTU_Group www.aktutor.in

b. Understand details of the hardware, software and network configurations used during testing before you begin the testing process.

c. It will help testers to create more efficient tests. It will also help to identify possible challenges that testers may encounter during the performance testing procedures.

2. **Identify the performance acceptance criteria :**

a. This includes goals and constraints for throughput, response times and resource allocation.

b. It is also necessary to identify project success criteria outside of these goals and constraints.

c. Testers should be empowered to set performance criteria and goals because often the project specifications will not include a wide enough variety of performance benchmarks.

d. Sometimes there may be none at all. When possible, finding a similar application to compare to is a good way to set performance goals.

3. **Plan and design performance tests :**

a. Determine how usage is likely to vary amongst end users and identify key scenarios to test for all possible use cases.

b. It is necessary to simulate a variety of end users, plan performance test data and outline what metrics will be gathered.

4. **Configuring the test environment :** Prepare the testing environment before execution. Also, arrange tools and other resources.

5. **Implement test design :** Create the performance tests according to your test design.

6. **Run the tests :** Execute and monitor the tests.

7. **Analyze, tune and retest :**

a. Consolidate, analyze and share test results. Then fine tune and test again to see if there is an improvement or decrease in performance.

b. Since improvements generally grow smaller with each retest, stop when bottlenecking is caused by the CPU.

c. Then you may have to consider option of increasing CPU power.

Que 5.15. **What is database testing ? Why we do database testing?**

Answer

1. Databases, the collection of interconnected files on a server, storing information, may not deal with the same type of data, *i.e.* databases may be heterogeneous.

2. As a result, many kinds of implementation and integration errors may occur in large database systems, which negatively affect the system's performance, reliability, consistency and security.

3. Thus, it is important to test in order to obtain a database system which satisfies the ACID properties (Atomicity, Consistency, Isolation, and Durability) of a system.

4. Database testing usually consists of a layered process, including the user interface (UI) layer, the business layer, the data access layer and the database itself.

5. The UI layer deals with the interface design of the database, while the business layer includes databases supporting business strategies.

6. One of the most critical layers is the data access layer, which deals with databases directly during the communication process.

7. Database testing mainly takes place at this layer and involves testing strategies such as quality control and quality assurance of the product databases.

Need of testing database :

1. **Data mapping :** In the software systems, data often travels back and forth from the user interface to the backend database and vice versa. So following are the aspects to look for :

a. To check whether the fields in the UI/Front end forms and mapped consistently with the corresponding database table. Typically this mapping information is defined in the requirements documents.

b. Whenever a certain action is performed in the front end of an application, a corresponding CRUD (Create, Retrieve, Update and Delete) action gets invoked at the back end. A tester will have to check if the right action is invoked and the invoked action in itself is successful or not.

2. **ACID properties validation :** Every transaction a database performs has to adhere to these four properties :

a. **Atomicity :** Atomicity means that a transaction either fails or passes. This means that even if a single part of transaction fails, it means that the entire transaction has failed. Usually this is called the "all-or nothing" rule.

b. **Consistency :** A transaction will always result in a valid state of the database.

c. **Isolation :** If there are multiple transactions and they are executed all at once, the result/state of the database should be the same as if they were executed one after the other.

d. **Durability :** Once a transaction is done and committed, no external factors like power loss or crash should be able to change it.

3. **Data integrity :**

a. This means that following any of the CRUD operations, the updated and most recent values/status of shared data should appear on all the forms and screens.

Join Telegram Group :- @AKTU_Group Scanned by TapScanner

Join Telegram Group :- @AKTU_Group www.aktutor.in

b. A value should not be updated on one screen and display an older value on another one.

c. So, devise your database test cases in a way to include checking the data in all the places it appears to see if it is consistently the same.

4. **Business rule conformity :**

a. More complex databases means more complicated components like relational constraints, triggers, stored procedures, etc.

b. So, testers will have to come up with appropriate SQL queries in order to validate these complex objects.

Que 5.16. Explain types of database testing.

Answer

Based on the function and structure of a database, DB testing can be categorized into three categories :

1. **Structural database testing :** It deals with table and column testing, schema testing, stored procedures and views testing, checking triggers, etc.
2. **Functional testing :** It involves checking functionality of database from user point of view. Most common type of functional testing are white box and black box testing.
3. **Nonfunctional testing :** It involves load-testing, risk testing in database, stress testing, minimum system requirements, and deals with the performance of the database.

Structural database testing :

1. Structural database testing involves verifying those components of database, which are not exposed to end users.
2. It involves all the components of repository, which are used to store the data and are not changed by the end users.
3. Database administrators with good command over SQL stored procedures and other concepts normally perform this testing.
4. Discussed are the common components tested with respect to structural testing :

a. **Schema / Mapping testing :**

i. It involves validating the objects of front-end application with database object mapping.

ii. In schema testing :

1. Sometimes it happens that the end user application objects are not correctly mapped or compatible with database objects.

2. Therefore, checking the validation of the various schema formats associated with the databases is required.
3. It is required to find the unmapped objects in database, like tables, views, columns etc.
4. There are various tools in the market that can be used to perform object mapping in schemas.

b. **Stored procedures and views testing :**

i. In this testing, a tester ensures that the manual execution of stored procedures and views generate the required result.

ii. The tester ensures :

1. If it enables the required triggers to be executed as expected.
2. If the development team has covered all the loops and conditions by passing input to applications in the procedures.
3. If there are any unused stored procedures in the database.
4. TRIM operations are applied properly when the data is fetched from required tables in database.
5. Validation of the overall integration of the stored procedure modules as per the requirements of the application under test.
6. Exception and error handling mechanisms are followed.
7. The most common tools that are used to perform stored procedures testing are LINQ, SP Test tool, etc.

c. **Trigger testing :** In trigger testing, a tester needs to ensure the following :

i. Whether the coding conventions are followed during the coding phase of the triggers.

ii. See the triggers executed meets the required conditions.

iii. Whether the trigger updates the data correctly, once they have been executed.

iv. Validation of Update/Insert/Delete triggers functionality with respect to application under test.

d. **Tables and column testing :** The key areas covered in this testing are :

i. Validating the data types in the database to field values in front-end application.

ii. Validating the length of data field in database to length of data types in the application.

iii. Checking if there are any unmapped tables or columns in the database from application field objects.

Join Telegram Group :- @AKTU_Group Scanned by TapScanner

Join Telegram Group :- @AKTU_Group www.aktutor.in

iv. Naming conventions of database tables and columns are verified, if they are in accordance with business requirement or not.

v. Validating the keys and indexes in the database, *i.e.*, primary and foreign keys in tables are defined as per requirement.

vi. Check if the primary keys and their corresponding foreign keys are same in two tables.

vii. Check Unique and NOT NULL characteristics of keys are maintained.

viii. Length and data type of keys and indexes are maintained as per requirement.

e. **Database server check :** Database server check involves verifying :

i. If the database server can handle the expected number of transactions as per the business requirement.

ii. If the configuration details of database servers meets the business requirement.

iii. If the user authorization is maintained as per requirement.

Functional testing : Functional testing is performed keeping in mind an end-user point of view; whether the required transactions and operations run by the end-users meet the business specifications.

1. **Black box testing :**

a. Black box testing involves verifying the integration of database to check the functionality.

b. The test cases are simple and are used to verify incoming data and outgoing data from the function.

c. Various techniques such as cause-effect graphing technique, equivalence partitioning and boundary-value analysis are used to test the functionality of the database.

d. Its advantages are as follows :

i. It is fairly simple and is performed in the early stages of development.

ii. Cost of developing test-cases is less as compared to white-box testing.

e. Its disadvantages are as follows :

i. A few errors cannot be detected.

ii. It is unknown how much program needs to be tested.

2. **White box testing :**

a. White box testing deals with the internal structure of the database and the specification details are hidden from the users.

b. It involves the testing of database triggers and logical views, which are going to support database refactoring.

c. It performs module testing of database functions, triggers, views, SQL queries etc.

d. This type of testing validates database tables, data models, database schema etc.

e. It checks rules of referential integrity.

f. It selects default table values to check on database consistency.

g. The most common techniques used to perform white box testing are condition coverage, decision coverage, statement coverage, etc.

h. Coding errors can be detected in white box testing, so internal bugs in the database can be eliminated.

i. The limitation of white box testing is that SQL statements are not covered.

Nonfunctional testing : Nonfunctional testing involves performing load testing, stress testing, checking minimum system requirements to meet business specification, risk finding and performance optimization of database.

1. **Load testing :**

a. The primary target of load testing is to check if most running transactions have performance impact on the database.

b. In load testing, the tester checks :

i. The response time for executing the transactions for multiple remote users.

ii. Time taken by the database to fetch specific records.

iii. Running most used transaction repeatedly to see performance of database system.

iv. Downloading a series of large files from the internet.

v. Running multiple applications on a computer or server simultaneously.

2. **Stress testing :**

a. Stress testing is performed to identify the system breakpoint.

b. In this testing, application is loaded in such a way that the system fails at one point.

c. This point is called the breakpoint of database system.

d. Determining the state of database transactions involves a significant amount of effort.

e. Proper planning is required to avoid any time and cost-based issues.

f. The most commonly used stress testing tools are LoadRunner and WinRunner.

Join Telegram Group :- @AKTU_Group Scanned by TapScanner

Join Telegram Group :- @AKTU_Group www.aktutor.in

Que 5.17. **What is post deployment testing ?**

Answer

Post deployment testing :

1. Post deployment testing may reveal those problems which went undetected before deployment of the web application.
2. Despite all the planning and testing carried out before deployment, obtaining user opinion is important for improvement of a website and it ensures that the website adapts to the need of the user.
3. User feedback may come in various forms, ranging from reporting of faults to suggestions for improvement of the website.
4. The effective way to obtain a user's opinion is to get questionnaire or survey filled by the user.
5. The questionnaire / survey can be used to detect trends and may provide valuable information for improvement of the website.

Once the user's opinion is obtained, it is important to identify useful fault reporting, suggestions and recommendations. The following criteria can be used to decide which suggestion needs attention :

1. **Frequency of suggestion :** How many user's have given the same suggestion or recommendation ? If a small number of users are making the same request, then we must think twice before implementing the suggestion.
2. **Source of feedback :** Who is providing the suggestion ? It is vital to make sure that suggestions come from regular users and not accidental users.
3. **Cost of implementing the suggestion :** Is the suggested idea worth implementing ? The correctness of the proposed change and its impact on the cost schedule must be analyzed carefully. The benefits of implementing the suggested idea to the business must be determined.
4. **Impact of implementing the suggestion :** Will implementing the suggestion increase complexity of the website ? Will the change be compatible with the other functionalities of the website ? It is important to obtain the answers to these questions as the results of implementing a change are sometimes unpredictable.

Que 5.18. **Write the difference between GUI testing and usability testing.**

Answer

S.No.	GUI testing	Usability testing
1.	In GUI testing, tester tests the application front end design to see whether it meets the client requirements or not.	In usability testing, tester tests that whether the application is user friendly or not by checking how easily user can access the application.
2.	In GUI testing, we check whether the design and layout of application as per the standards and client requirements or not.	In usability testing, we check whether the design and layout of application is easy to use or not means it is user friendly or not.
3.	GUI testing is more concerned with look and feel of the application means how people react and feel after looking into the application so its testing is done according to that.	Usability testing is more concerned with easiness and user friendliness of the application.
4.	In GUI testing, tester tests the appearance of the software.	In usability testing, tester tests the easiness to use the software.
5.	GUI testing is done to ensure that it meets the design specifications like links, colours, fonts, font sizes, fields etc., are displayed as specified in SRS or as specified in client requirements.	Usability testing is done to ensure that the GUI is well designed and easy to use like links and buttons are easily clickable and leaving any of the mandatory field blank gives the proper message that please enter the xyz in mandatory field.
6.	GUI testing is done by keeping in mind the look and feel of application means how application looks.	Usability testing is done by keeping the end user in mind.
7.	It stands for Graphical User Interface. It is nothing, it only confirm the design specifications with the application.	It is done to ensure that the GUI is well designed and easy to use.
8.	It is done on different platforms to verify the look and feel testing.	It is done to verify how much the application is user friendly to an end user.

Join Telegram Group :- @AKTU_Group Scanned by TapScanner

Join Telegram Group :- @AKTU_Group
www.aktutor.in

9.	In GUI testing, tester test whether the front end design of the system is meeting with project standards or not.	In usability testing, tester tests whether the control flow of the system is convenient for end user or not.
11.	Example includes colours, fonts, font sizes, buttons, links, icons, placement of data labels and fields etc. are displayed as specified or not.	Example includes firstly displayed all mandatory fields, cursor positioning for enter the data into the right field, tab button should work easily etc.
12.	In GUI testing, we only focus on the interface of the application.	Quality of product is depending on usability testing.
13.	In this testing, we test only the front end of the application.	In this testing, we test the overall working of application according to a non-technical user's point of view.

☺☺☺

UNIT 1

Review of Software Engineering and Verification (2 Marks Questions)

Memory Based Questions

1.1. What is s-type (static-type) software ?

Ans. **S-type (static-type) :** This is a software which works strictly according to defined specifications and solutions. The solution and the method to achieve it, both are immediately understood before coding. The s-type software is least subjected to changes, hence this is the simplest of all. For example, calculator program for mathematical computation.

1.2. What is p-type (practical-type) software ?

Ans. **P-type (practical-type) :** This is a software with a collection of procedures. It is defined by exactly what procedures can do. In this software, the specifications can be described but the solution is not obvious instantly. For example, gaming software.

1.3. What is e-type (embedded-type) software ?

Ans. **E-type (embedded-type) :** This software works closely as the requirement of real-world environment. This software has a high degree of evolution as there are various changes in laws, taxes etc. in the real world situations. For example, online trading software.

1.4. What are the stages of a typical SDLC ?

Ans. **Stages of a typical SDLC are :**
Stage 1 : Planning and requirement analysis
Stage 2 : Defining requirements
Stage 3 : Designing the product architecture
Stage 4 : Building or developing the product
Stage 5 : Testing the product
Stage 6 : Deployment in the market and maintenance

1.5. List some SDLC models.

Join Telegram Group :- @AKTU_Group Scanned by TapScanner

Join Telegram Group :- @AKTU_Group www.aktutor.in

Ans. Following are the most important and popular SDLC models used in the industry :

1. Waterfall model
2. Iterative model
3. Spiral model
4. V-model

The other related methodologies are big bang model, agile model, RAD model - Rapid application development and prototyping models etc.

1.6. What is testing process ? List the basic steps of it.

Ans. Testing is a process rather than a single activity. This process starts from test planning, then designing test cases, preparing for execution and evaluating status till the test closure. So, we can divide the activities within the fundamental test process into the following basic steps :

1. Planning and control
2. Analysis and design
3. Implementation and execution
4. Evaluating exit criteria and reporting
5. Test closure activities

1.7. Discuss the term error.

Ans. Error is defined as a human action that produces an incorrect result. In other words, the mistakes made by programmer are known as 'error'. This could happen because of the following reasons :

1. Some confusion in understanding the requirement of the software
2. Some miscalculation of the values
3. Misinterpretation of any value, etc.

1.8. Discuss the terms fault/defect.

Ans. Fault is defined as a manifestation of an 'Error' in software. Faults are also known colloquially as defaults or bugs. The 'Error' introduced by programmer inside the code are known as a 'Defect'. This can happen because of some programmatically 'Error'.

1.9. Explain the term failure.

Ans. Failure is a deviation of the software from its intended purpose. During the testing process, if defect gets executed by the tester then it results into the failure which is known as software failure.

1.10. Explain verification.

Ans. Verification is the system of confirming that software meets its specification. It involves reviews to evaluate documents, plans, code, software requirements, and specifications. It can be fulfill with

checklists to involve issues lists, walkthroughs by people, and inspection meetings.

Verification = "Are we building the project right".

1.11. Explain validation.

Ans. Validation is the system of confirming that software meets the user's requirements. It typically involves absolute testing and takes place after verification is completed. Validation means executing the software to see whether it meets the client requirements or not.

Validation = "Are we building the right project/product".

1.12. What is test case ?

Ans. A test case is a document which has a set of test data, preconditions, expected results and post conditions, developed for a particular test scenario in order to verify compliance against a specific requirement. Test case acts as the starting point for the test execution, and after applying a set of input values ; the application has a definitive outcome and leaves the system at some end point or also known as execution post condition.

1.13. List the standard fields of test case.

Ans. Standard fields of test case are :

1. Test case ID
2. Test priority (low/medium/high)
3. Module name
4. Test designed by
5. Test designed date
6. Test executed by
7. Test execution date
8. Test title/name
9. Test summary/description
10. Pre-condition
11. Dependencies
12. Test steps
13. Test data
14. Expected result
15. Post-condition
16. Actual result
17. Status (pass/fail)
18. Notes/comments/questions

1.14. What is test suite in software testing ?

Ans. In software development, a test suite, less commonly known as a 'validation suite', is a collection of test cases that are intended to be used to test a software program to show that it has some specified set of behaviours. A test suite often contains detailed instructions

Join Telegram Group :- @AKTU_Group Scanned by TapScanner

Join Telegram Group :- @AKTU_Group www.aktutor.in

or goals for each collection of test cases and information on the system configuration to be used during testing. A group of test cases may also contain prerequisite states or steps, and descriptions of the tests.

1.15. What are the levels of verification ?

Ans. There are four levels of verification :

1. **Component testing :** Testing conducted to verify the implementation of the design for one software element (unit, module) or a collection of software elements.
2. **Integration testing :** An orderly progression of testing in which various software elements and/or hardware elements are integrated together and tested. This testing proceeds until the entire system has been integrated.
3. **System testing :** The process of testing an integrated hardware and software system to verify that the system meets its specified requirements.
4. **Acceptance testing :** Formal testing conducted to determine whether or not a system satisfies its acceptance criteria and to enable the customer to determine whether or not to accept the system.

1.16. Explain verifications types.

Ans. There are four types of verification :

1. **Inspection:** Typical techniques include desk checking, walkthroughs, software reviews, technical reviews, and formal inspections.
2. **Analysis :** Mathematical verification of the test item, which can include estimation of execution times and estimation of system resources.
3. **Testing :** It is also known as "white box" or "logic driven" testing. Given input values are traced through the test item to assure that they generate the expected ou put values, with the expected intermediate values along the way. Typical techniques include statement coverage, condition coverage, and decision coverage.
4. **Demonstration :** It is also known as "black box" or "input/output driven" testing. Given input values are entered and the resulting output values are compared against the expected output values. Typical techniques include error guessing, boundary-value analysis, and equivalence partitioning.

1.17. What do you understand by software requirements specification (SRS) ?

Ans. A software requirements specification (SRS) is a detailed description of a software system to be developed with its functional and non-functional requirements. The SRS is developed based on the agreement between customer and contractors. It may include the

use cases of how user is going to interact with software system. A good SRS defines how the software system will interact with all internal modules, hardware, communication with other programs and human user interactions with wide range of real life scenarios.

1.18. What do you understand by formal code review ?

Ans. Formal code review such as a Fagan inspection, involves a careful and detailed process with multiple participants and multiple phases. Formal code reviews are the traditional method of review, in which software developers attend a series of meetings and review code line by line, usually using printed copies of the material. Formal inspections are extremely thorough and have been proven effective at finding defects in the code under review.

1.19. What is lightweight code review ?

Ans. Lightweight code review typically requires less overhead than formal code inspections, though it can be equally effective when done properly. Lightweight reviews are often conducted as part of the normal development process :

1. **Over-the-shoulder :** One developer looks over the author's shoulder as the latter walks through the code.
2. **Email pass-around :** Source code management system emails code to reviewers automatically after check has been made.
3. **Pair programming :** It is a technique in which two authors develop code together at the same workstation.

1.20. What is walkthrough ?

Ans. Walkthrough is the method of conducting informal group/individual review in which a designer or programmer leads members of the development team and other interested parties through a software product, and the participants ask questions and make comments about possible errors, violation of development standards, and other problems or may suggest improvement on the article. Walkthrough can be pre-planned or can be conducted at need basis and generally people working on the work product are involved in the walkthrough process.

1.21. What is inspection ?

Ans. An inspection is a formal, rigorous, in-depth group review designed to identify problems as close to their point of origin as possible. Inspection is a recognized industry best practice to improve the quality of a product and to improve productivity. Inspection is a formal review and generally need is predefined at the start of the product planning.

1.22. What are the goals of inspection ?

Ans. The goals of inspection are :

Join Telegram Group :- @AKTU_Group Scanned by TapScanner

Join Telegram Group :- @AKTU_Group
www.aktutor.in

1. It helps the author to improve the quality of the document under inspection.
2. It removes defects efficiently and as early as possible.
3. It improves product quality.
4. It creates common understanding by exchanging information.
5. It learns from defects found and prevent the occurrence of similar defects.

1.23. Explain functional configuration audit.

Ans. The objective of the functional audit is to provide an independent evaluation of a software product, verifying that its configuration item's actual functionality and performance is consistent with the relevant requirement specification. This audit is held prior to software delivery to verify that all requirements specified in the software requirements specification have been met.

1.24. Explain physical configuration audit.

Ans. The objective of the physical audit is to provide an independent evaluation of a software product's configuration items to confirm that all components in the as-built version map to their specifications. Specifically, this audit is held to verify that the software and its documentation are internally consistent.

☺☺☺

2 UNIT

Functional and Structural Testing (2 Marks Questions)

Memory Based Questions

2.1. Explain functional testing.

Ans. Functional testing is a quality assurance process and a type of black box testing that bases its test cases on the specifications of the software component under test. Functions are tested by feeding them input and examining the output, and internal program structure is rarely. Functional testing usually describes what the system does.

2.2. List some functional techniques.

Ans. Functional testing techniques include :

1. Unit testing
2. Integration testing
3. Smoke testing
4. User acceptance testing
5. Localization testing
6. Interface testing
7. Usability testing
8. System testing
9. Regression testing
10. Globalization testing

2.3. Write the advantages of boundary value analysis.

Ans. **Advantages of boundary value analysis :**

1. Small set of test cases is prepared.
2. It reduces the number of test cases for a tester.
3. Very good at exposing potential user interface/user input problems.
4. Easy to test the boundary values.
5. Provides clear guidelines to prepare the test cases.
6. It is used to find the bugs occur at boundaries rather than exist in the center.
7. It gives the technique in which input domain divided into different class and select each class for testing so we can say that we can easily find the bugs.

2.4. Write the disadvantages of boundary value analysis.

Ans. **Disadvantages of boundary value analysis :**

Join Telegram Group :- @AKTU_Group Scanned by TapScanner

Join Telegram Group :- @AKTU_Group www.aktutor.in

1. Difficult to prepare the test cases if the boundary values are not known correctly.
2. Does not test dependencies between combinations of inputs.
3. Not useful for large amount of values.
4. Not all possible inputs are tested.

2.5. What are the limitations of equivalence class partitioning ?

Ans. Limitations of equivalence class partitioning :

1. ECP usually does not consider the boundary conditions.
2. Sometimes issues at boundaries are ignored.
3. Testers tend to assume that result will be correct for all input data-set.

2.6. What is decision table ?

Ans. A decision table is the method used to build a complete set of test cases without using the internal structure of the program in question. In order to create test cases we use a table to contain the input and output values of a program. Such a table is split up into four sections as shown below in figure.

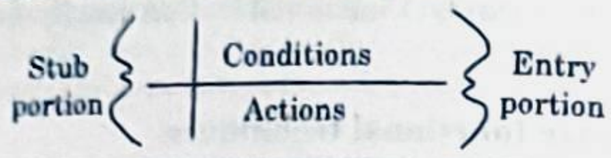


Fig. 2.6.1. The basic structure of a decision table.

2.7 Write the applications of decision tables.

Ans. This technique is useful for applications characterized by any of the following :

a. Prominent if-then-else logic.
b. Logical relationships among input variables.
c. Calculations involving subsets of the input variables.
d. Cause and effect relationships between inputs and outputs.
e. High cyclomatic complexity.

2.8. What is cause-effect graph ?

Ans. A cause-effect graph is a directed graph that maps a set of causes to a set of effects. The causes may be thought of as the input to the program, and the effects may be thought of as the output. Usually the graph shows the nodes representing the causes on the left side and the nodes representing the effects on the right side. There may be intermediate nodes in between that combine inputs using logical operators such as AND and OR. A "Cause" represents a distinct input condition that brings about an internal change in the system. An "Effect" represents an output condition, a system transformation or a state resulting from a combination of causes.

2.9. Why we use cause-effect diagram ?

Ans. The cause-effect diagram can be used under these circumstances :

1. To determine the current problem so that right decision can be taken very fast.
2. To narrate the connections of the system with the factors affecting a particular process or effect.
3. To recognize the probable root causes, the cause for an exact effect, problem, or outcome.

2.10. What are the symbols used in cause-effect graphs ?

Ans. Symbols used in cause-effect graphs :

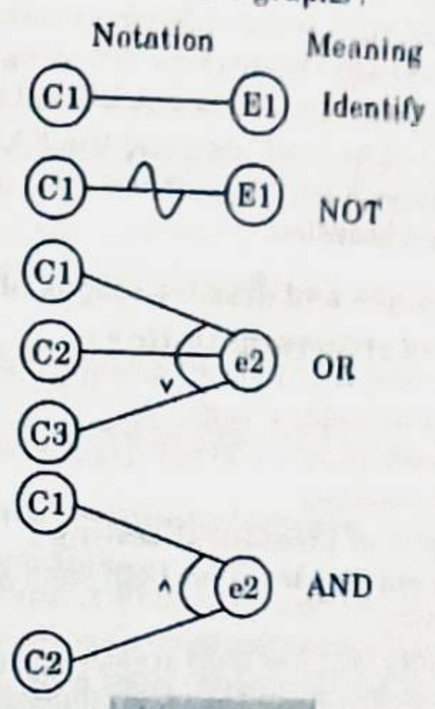


Fig. 2.10.1.

2.11. What are the benefits of cause-effect graphs ?

Ans. Benefits of cause-effect graphing :

1. It helps us to determine the root causes of a problem or quality using a structured approach.
2. It uses an orderly, easy to read format to diagram cause and effect relationships.
3. It indicates possible causes of variation in a process.
4. It identifies areas, where data should be collected for further study.
5. It encourages team participation and utilizes the team knowledge of the process.
6. It increases knowledge of the process by helping everyone to learn more about the factors at work and how they relate.

2.12. Why we do structural testing ?

Ans. Structural testing is done :

Join Telegram Group :- @AKTU_Group Scanned by TapScanner

Join Telegram Group :- @AKTU_Group www.aktutor.in

1. To understand what is missing in our test suite.
2. To complement functional testing.
3. It helps to identify obvious inadequacies.

2.13. Write various categories of structural testing.

Ans. The whole structural testing is categorized into four divisions :

1. **Statement coverage :** It is the weakest form of testing, as it requires that every statement in the code has to be executed at least once.
2. **Branch coverage :** In this each branch condition for the program is tested for its true or false values.
3. **Path coverage :** For path coverage, the path of the program is executed at least once, it test individual path for the program.
4. **Condition coverage :** In this type of testing, it checks all possible combinations of conditions. For conditional branches, we execute the TRUE branch at least once and the FALSE branch, at least, once. Unlike branch coverage, it tests for conditional as well as non- conditional branches.

2.14. Write advantages and disadvantages of structural testing.

Ans. **Advantages of structural testing :**

1. Forces test developer to reason carefully about implementation.
2. Reveals errors in "hidden" code.
3. Spots the dead code or other issues with respect to best programming practices.

Disadvantages of structural testing :

1. Expensive as one has to spend both time and money to perform white box testing.
2. Every possibility that few lines of code are missed accidentally.
3. In-depth knowledge about the programming language is necessary to perform white box testing.

2.15. Write the difference between structural and functional testing.

Ans. Difference between structural and functional testing :

S. No.	Structural Testing	Functional Testing
1.	This testing will uncover error occurred during the coding of the program.	This testing will uncover error occurred during the implementation of requirements and design specifications.
2.	It is concerned about both the result and the process.	It is concerned only about the result but not the processing.
3.	Structural testing is often referred as white box testing. The knowledge of code is very much essential.	Functional testing is often referred as black box testing, no need to know about the coding of the program.

2.16. What is the process of control flow testing ?

Ans. The basic idea behind control flow testing is to select paths and come up with the inputs (input values/test cases) to execute through those paths. It includes the following steps :

Step 1 : From the source code a control flow graph (CFG) is created either manually or automatically.

Step 2 : A coverage target is defined over the CFG example, nodes, paths, branches, edges etc.

Step 3 : Test cases are created using CFG to cover the coverage target.

Step 4 : Execute the test cases.

Step 5 : Analyze the results and determine whether the program is error free or has some bugs.

2.17. What are the notations used in flow graph ?

Ans. **Following notations are used for a flow graph :**

Node : It represents one or more procedural statements.

Edges or links : They represent the flow of control in a program.

Decision node : A node with more than one arrow leaving is called a decision node.

Junction node : A node with more than one arrow entering it is called a junction.

Regions : Areas bounded by edges and nodes are called regions.

2.18. List various path testing techniques.

Ans. **Path testing techniques :**

1. **Control flow graph (CFG) :** The program is converted into flow graphs by representing the code into nodes, regions and edges.
2. **Decision to Decision path (D-D) :** The CFG can be broken into various Decision to Decision paths and then collapsed into individual nodes.
3. **Independent (basis) paths :** Independent path is a path through a DD-path graph which cannot be reproduced from other paths by other methods.

2.19. What notations are used for representing flow graph ?

Ans. Notation for representing control flow are :

Join Telegram Group :- @AKTU_Group Scanned by TapScanner

Join Telegram Group :- @AKTU_Group www.aktutor.in

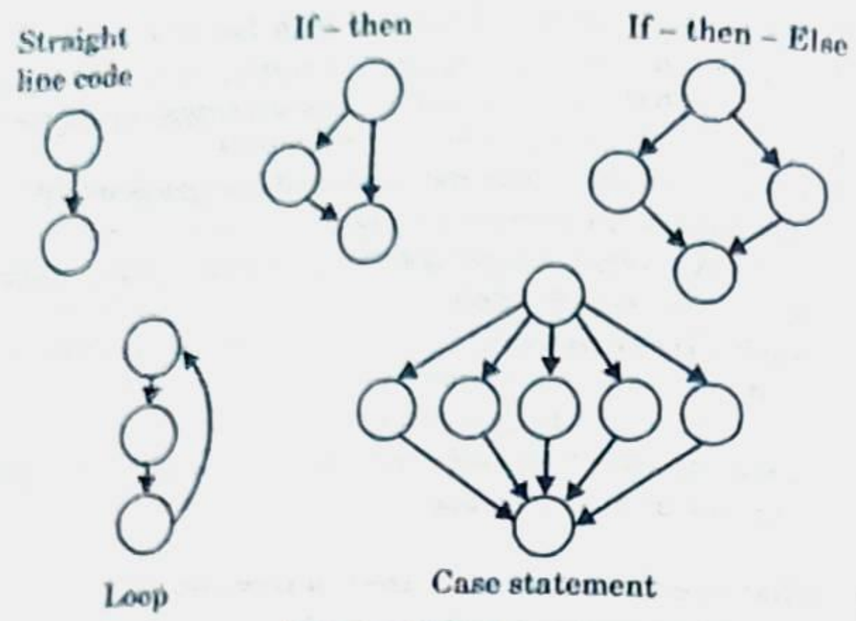


Fig. 2.19.1.

2.20. What steps involved in basis path testing ? Write benefits of basis path testing.

Ans. **The basic steps involved in basis path testing include :**

1. Draw a control graph (to determine different program paths).
2. Calculate cyclomatic complexity (metrics to determine the number of independent paths).
3. Find a basis set of paths.
4. Generate test cases to exercise each path.

Benefits of basis path testing :

1. It helps to reduce the redundant tests.
2. It focuses attention on program logic.
3. It helps and facilitates analytical versus arbitrary case design.
4. Test cases which exercise basis set will execute every statement in program at least once.

2.21. What is independent path ? How to calculate independent path ?

Ans. An independent path is any path through the program that introduces at least one new set of processing statements or a new condition. When stated in terms of a flow graph, an independent path must move along at least one edge that has not been traversed before the path is defined.

There are three equations that you can use to calculate the independent paths.

1. **Independent paths** = Edges – Nodes + 2
2. **Independent paths** = Regions + 1
3. **Independent paths** = Decisions + 1

2.22. Explain cyclomatic complexity.

Ans. Cyclomatic complexity is software metric used to measure the complexity of a program. These metric measures independent paths through program source code. Independent path is defined as a path that has at least one edge which has not been traversed before in any other paths. Cyclomatic complexity can be calculated with respect to functions, modules, methods or classes within a program. Control flow depicts a program as a graph which consists of nodes and edges. In the graph, nodes represent processing tasks while edges represent control flow between the nodes.

The complexity of the program can be defined as :

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

Where,

E = Number of edges , N = Number of nodes

$$V(G) = P + 1$$

Where P = Number of predicate nodes (node that contains condition)

2.23. List the some tools for calculating cyclomatic complexity.

Ans. **Tools for cyclomatic complexity calculation :**

1. OCLint : Static code analyzer for C and related languages
2. devMetrics : Analyzing metrics for C# projects
3. Reflector Add In : Code metrics for .NET assemblies
4. GMetrics : Find metrics in Java related applications
5. NDepends : Metrics in Java applications

2.24. Write advantages of data flow testing.

Ans. **Advantages of data flow testing :**

1. A variable that is declared but never used within the program.
2. A variable that is used but never declared.
3. A variable that is defined multiple times before it is used.
4. Deallocating a variable before it is used.

2.25. What is mutation testing ?

Ans. Mutation testing is a kind of testing in which, the application is tested for the code that was modified after fixing a particular bug/defect. It also helps in finding out which code and which strategy of coding can help in developing the functionality effectively. Mutation testing ensures the accuracy of our unit tests. In some cases our unit tests will be successful and won't detect any bugs or defects in our source code, but that may be not true. In these situations, we will do some alteration in source code (mutating) for making errors and again will run those unit test.

2.26. Explain various types of mutation testing.

Ans. **Mutation testing types :**

1. **Value mutations :** An attempt to change the values to detect errors in the programs. We usually change one value to a much

Join Telegram Group :- @AKTU_Group Scanned by TapScanner

Join Telegram Group :- @AKTU_Group
www.aktutor.in

larger value or one value to a much smaller value. The most common strategy is to change the constants.

2. **Decision mutations :** The decisions/conditions are changed to check for the design errors. Typically, one changes the arithmetic operators to locate the defects and also we can consider mutating all relational operators and logical operators (AND, OR, NOT).
3. **Statement mutations :** Changes done to the statements by deleting or duplicating the line which might arise when a developer is copy pasting the code from somewhere else.

2.27. Write the equation to calculate mutation score.

Ans. The mutation score is defined as the percentage of killed mutants with the total number of mutants.

Mutation Score = (Killed Mutants / Total number of Mutants) * 100

☺☺☺

3 UNIT

Regression Testing and Prioritization Technique (2 Marks Questions)

Memory Based Questions

3.1. What is regression testing ?

Ans. Regression testing is a type of software testing that verifies the software previously developed and tested. It checks whether the software still performs correctly after it was changed or interfaced with other software. During regression testing, new software bugs or regressions may be uncovered. One of the main reasons for regression testing is to determine whether a change in one part of the software affects other parts of the software.

3.2. When to use regression testing ?

Ans. Regression testing is used when :

1. Any new feature is added.
2. Any enhancement is done.
3. Any bug is fixed.
4. Any performance related issue is fixed.

3.3. How to do regression testing ?

Ans. An effective regression testing is made up of the following steps :

1. Performing an initial "smoke" or "sanity" test.
2. Understanding the criteria to select the test cases for regression testing.
3. Prioritization of test cases.
4. Methodology to select test cases.
5. Resetting the test cases for test execution.
6. Concluding the result of a regression test cycle.

3.4. Which types of test cases are selected for regression testing ?

Ans. Effective regression tests can be done by selecting following test cases :

1. Test cases which have frequent defects
2. Functionalities which are more visible to the users

Join Telegram Group :- @AKTU_Group Scanned by TapScanner

Join Telegram Group :- @AKTU_Group www.aktutor.in

3. Test cases which verify core features of the product
4. All integration test cases
5. All complex test cases
6. Boundary value test cases
7. Sample of successful test cases
8. Sample of failure test cases

3.5. Write advantages of regression testing.

Ans. Advantages of regression testing :

1. It helps us to make sure that any changes like bug fixes or any enhancements to the module or application have not impacted the existing tested code.
2. It also detects undesirable side caused by changing the operating environment.
3. It ensures that the bugs found earlier are not creatable.
4. It can be done by using the automation tools.
5. It helps in improving the quality of the product.

3.6. Write disadvantages of regression testing.

Ans. Disadvantages of regression testing are :

1. If regression testing is done without using automated tools then it can be very tedious and time consuming because here we execute the same set of test cases again and again.
2. Automation regression scripts needs skilled software testers.
3. Budget of the product increases.
4. With new functionality added, number of automation scripts also increases and hence needs high maintenance.

3.7. State the techniques of regression testing.

Ans. Regression testing can be carried out using following techniques :

1. **Retest all :**
   This is one of the methods for regression testing in which all the tests in the existing test bucket or suite should be re-executed. This is very expensive as it requires huge time and resources.
2. **Regression test selection :**
   a. Instead of re-executing the entire test suite, it is better to select part of test suite to be run.
   b. Test cases selected can be categorized as : Reusable Test Cases and Obsolete Test Cases.
   c. Re-usable test cases can be used in succeeding regression cycles.
   d. Obsolete test cases can't be used in succeeding cycles.
3. **Prioritization of test cases :** Prioritize the test cases depending on business impact, critical and frequently used functionalities. Selection of test cases based on priority will greatly reduce the regression test suite.

3.8. Explain regression testing tools.

Ans. Following are most important tools used for regression testing :

1. **Quick test professional (QTP) :** HP Quick Test Professional is automated software designed to automate functional and regression test cases. It uses VBScript language for automation. It is a data driven, keyword based tool.
2. **Rational functional tester (RFT) :** IBM's rational functional tester is a java tool used to automate the test cases of software applications.
3. **Selenium :** This is an open source tool used for automating web applications. Selenium can be used for browser based regression testing.

3.9. Write the difference between re-testing and regression testing.

Ans. Re-testing means testing the functionality or bug again to ensure that the code is fixed. If it is not fixed, defect needs to be re-opened. If fixed, defect is closed.

Regression testing means testing your software application when it undergoes a code change to ensure that the new code has not affected other parts of the software.

3.10. What is Prioritization of test cases in regression testing ?

Ans. Prioritizing the test cases depends on the business impact, critical and frequently used functionality. Selection of test cases based on priority will reduce the test suit. The test cases may be classified into three categories :

Priority-0 : These test cases can be called as Sanity test cases which check the basic functionality and are run for accepting the build for further testing. These are also run when a project goes through major changes.

Priority-1 : These test cases deliver high project value to both development teams and customers.

Priority-2 : These test cases deliver moderate project value and are executed as a part of software testing life cycle and selected for regression on need basis.

3.11. What are test case selection criteria in code coverage prioritization technique ?

Ans. The technique is based on version specific test case prioritization where information about changes in the program is known. Hence, prioritization is focused around the changes in the modified program.

3.12. What methodology is used for selection of regression test cases ?

Ans. Once the test cases are prioritized, test cases can be selected. There could be several approaches to regression testing which need to be decided on a case by case basis. For example :

Join Telegram Group :- @AKTU_Group Scanned by TapScanner

Join Telegram Group :- @AKTU_Group
www.aktutor.in

Case 1 : If criticality and impact of the defect fixes are low, then it is enough to select few test cases from Test Case DataBase (TCDB) and execute them. These can fall under any priority (0, 1, or 2).
Case 2 : If the criticality and the impact of the bug fixes are medium, then we need to execute all priority-0 and priority-1 test cases. If bug fixes need additional test cases from priority-2, then those test cases can also selected and used for regression testing.
Case 3 : If the criticality and impact of the bug fixes are high, then we need to execute all priority-0, priority-1 and carefully selected priority-2 test cases.

3.13. What are prioritization guidelines ?

Ans. Test case prioritization is a compromise : the tester selects the situations to execute at the expense of those to omit. The goal of prioritization is to reduce the set of test cases based on some rational, non-arbitrary criteria, while aiming to select the most appropriate tests.
The prioritization schemes basically address these key concepts :
1. What features must be tested ?
2. What are the consequences if some features are not tested ?

3.14. Explain priority category scheme.

Ans. The easiest scheme for categorizing tests is to assign a priority code directly to each test description.
The following three-level priority categorization scheme is as :
Priority-1 : This test must be executed.
Priority-2 : If time permits, execute this test.
Priority-3 : If this test is not executed, the team won't be upset.

3.15. Explain risk analysis.

Ans. All software projects benefit from risk analysis. Standards governing safety critical applications require risk analysis as an integral part of planning and developing software. Using risk analysis at the beginning of a project highlights the potential problem areas whose failures have more serious adverse consequences. This allows developers and product managers to pay special attention when designing the application and consequently to mitigate the risks. The tester uses the results of risk analysis to select the most crucial tests.

3.16. Explain risk matrix.

Ans. A risk matrix allows the tester to evaluate and rank potential problems by giving more weight to the probability or severity value as necessary. The tester uses the risk matrix to assign thresholds that classify the potential problems into priority categories.

☺☺☺

4 UNIT

Software Testing Activities (2 Marks Questions)

Memory Based Questions

4.1. With the help of diagram depict the level of testing.

Ans. **Levels of testing are :**
1. Unit/Component testing
2. Integration testing
3. System testing
4. Acceptance testing

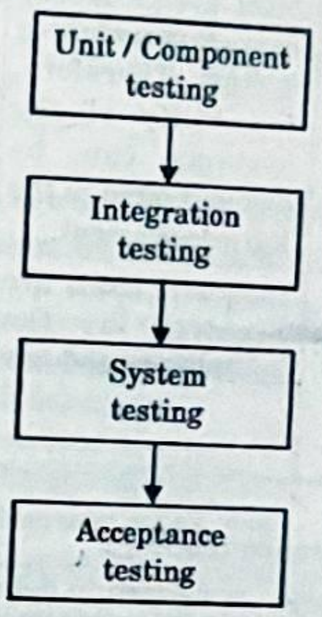


Fig. 4.1.1. Level of testing.

4.2. Write some advantages of unit testing.

Ans.
1. It reduces defects in the newly developed features.
2. It reduces bugs when changing the existing functionality.
3. Reduces cost of testing as defects are captured in very early phase.
4. Improves design and allows better refactoring of code.

4.3. What is integration testing ?

Join Telegram Group :- @AKTU_Group Scanned by TapScanner

Join Telegram Group :- @AKTU_Group www.aktutor.in

Ans. Integration testing is executed to establish whether the components interact with each other consort to the specification or not. It refers to joining all the components resulting in the complete system. For example : checking that a payroll system interacts as required with the human resource system.

4.4. What is the difference between top-down and bottom-up integration ?

Ans.

Issue	Top-down Testing	Bottom-up Testing
Architectural at design	It discovers errors in high-level design, thus detects errors at an early stage.	High-level design is validated at a later stage.
System Demonstration	Since we integrate the modules from top to bottom, the high-level design slowly expands as a working system. Therefore, feasibility of the systems can be demonstrated to the top management.	It may not be possible to show the feasibility of the design. However if some modules are already built as reusable components, then it may be possible to produce some kind of demonstration.
Test Implementation	(nodes-1) stubs are required for the subordinate modules.	(nodes-leaves) test drivers are required for super-ordinate modules to test the lower-level modules.

4.5. What is acceptance testing ?

Ans. A testing technique performed to determine whether or not the software system has met the requirement specifications. There are various forms of acceptance testing :

1. User acceptance testing
2. Business acceptance testing
3. Alpha testing
4. Beta testing

4.6. List the types of system testing.

Ans. Some listed types of system testing are :

1. Usability testing
2. Load testing
3. Regression testing

4. Recovery testing
5. Migration testing
6. Functional testing
7. Hardware/Software testing

4.7. What is regression testing ?

Ans. Regression testing is a type of software testing that verifies the software previously developed and tested. It checks whether the software still performs correctly after it was changed or interfaced with other software.

4.8. What is debugging ?

Ans. On successful culmination of software testing, debugging is performed. Debugging is defined as a process of analyzing and removing the error. It is considered necessary in most of the newly developed software or hardware and in commercial products/ personal application programs. For complex products, debugging is done at all the levels of the testing.

4.9. List commonly used debugging strategies.

Ans. The commonly used debugging strategies are :

1. Brute force
2. Induction strategy
3. Deduction strategy
4. Backtracking strategy
5. Debugging by testing

4.10. What guidelines are used while performing debugging ?

Ans. Some guidelines that are followed while performing debugging are :

1. Debugging is the process of solving a problem. Hence, individuals involved in debugging should understand all the causes of an error before starting with debugging.
2. No experimentation should be done while performing debugging. The experimental changes instead of removing errors often increase the problem by adding new errors in it.
3. If an error is found in one segment of a program, rest of the program should be properly examined for errors.
4. To verify the correctness of a new code and to ensure that no new errors are introduced, regression testing should be performed.

4.11. What is user testing ?

Ans. User testing might be done at any time during development. Some types of user testing such as task analysis look more like joint exploration (involving at least one user and at least one member of your company's testing team) than like testing by one person.

4.12. What is alpha testing ?

Ans. This is a form of internal acceptance testing performed mainly by

Join Telegram Group :- @AKTU_Group Scanned by TapScanner

Join Telegram Group :- @AKTU_Group www.aktutor.in

in-house software QA and testing teams. Alpha testing takes place at the developer's site by the internal teams before release to external customers. This testing is performed without the involvement of the development teams.

4.13. What is beta testing ?

Ans. Beta testing is also known as user testing that takes place at the end user's site by the end users to validate the usability, functionality, compatibility, and reliability testing. Beta testing adds value to the software development life cycle as it allows the "real" customer an opportunity to provide inputs into the design, functionality, and usability of a product.

4.14. Explain risk based testing.

Ans. Risk based testing is basically a testing done for the project based on risks. It uses risk to prioritize and emphasize the appropriate tests during test execution. It also involves testing the functionality which has the highest impact and probability of failure.

4.15. Write advantages and disadvantages of exploratory testing.

Ans. **Advantages of exploratory testing are :**

1. It doesn't require preparation for testing as we don't have documents for testing.
2. In this type of testing, time saves because all tasks are performed simultaneously like testing, designing test scenarios and executing test scenarios.
3. Tester can report many issues due to incomplete requirement or missing requirement document.

Disadvantages of exploratory testing are :

1. Few issues cannot be catch in this type of testing.
2. There is review of test planning & designing of test cases/scenario while testing may cause issues.
3. Testers have to remember the scenario what he is executing.

4.16. What are the benefits of automated test data generation ?

Ans.

1. Fast
2. Accurate
3. Can use built-in algorithms
4. Create both valid and invalid test data
5. Data exportable to multiple formats
6. Create a professional look

4.17. What is exploratory testing ?

Ans. Exploratory testing is about exploring, finding out about the software, what it does, what it doesn't do, what works and what

doesn't work. This is an approach, that is, most useful when there are no or poor specifications and when time is severely limited.

4.18. What are the approaches used for test data generators ?

Ans. **Approaches used for test data generators are :**

1. Random test data generators
2. Pathwise data generators
3. Goal oriented generators
4. Intelligent test data generators

4.19. Explain test planning activities.

Ans.

1. To determine the scope and the risks that need to be tested and that are not to be tested.
2. Documenting test strategy.
3. Making sure that the testing activities have been included.
4. Deciding entry and exit criteria.
5. Evaluating the test estimate.
6. Defining the management information, including the metrics required and defect resolution and risk issues.
7. Ensuring that the test documentation generates repeatable test assets.

☺☺☺

Join Telegram Group :- @AKTU_Group Scanned by TapScanner

Join Telegram Group :- @AKTU_Group
www.aktutor.in

5
UNIT

Object-Oriented Testing (2 Marks Questions)

Memory Based Questions

5.1. List the methods used for performing object-oriented testing.

Ans. 1. State-based testing
2. Fault-based testing
3. Scenario-based testing

5.2. What steps are used to perform state based testing ?

Ans. **Following are the steps :**
1. Derive a new class from an existing class with some additional features which are used to examine and set the state of the object.
2. Next the test driver is written. This test driver contains a main program to create an object, send messages to set the state of the object, send messages to invoke methods of the class that is being tested and send messages to check the final state of the object.
3. Finally, stubs are written. These stubs call the untested methods.

5.3. What points are included in scenario based testing ?

Ans. **Following are the points :**
1. A condition under which the scenario runs.
2. A goal to achieve, which can also be a name of the scenario.
3. A set of steps of actions.
4. An end condition at which the goal is achieved.
5. A possible set of extensions written as scenario fragments.

5.4. What points should be noted while developing test cases in OO environment ?

Ans. **Following are the points :**
1. It should be explicitly specified with each test case which class it should test.
2. Purpose of each test case should be mentioned.
3. External conditions that should exist while conducting a test should be clearly stated with each test case.

4. All the states of object, that is, to be tested should be specified.
5. Instructions to understand and conduct the test cases should be provided with each test case.

5.5. What is web form testing ?

Ans. In the web application testing, the "form testing" is the essential part of testing any website. The main purpose of form is to get the information from user & store into the database. And keep an interaction with them. Below are the test cases which should be considered while doing form testing :
1. First thing to test in the form is the validation on each form fields. Here, two types of validation need to be consider - "client side" and "server side" validations.
2. Check default values are being populated.
3. Check for all mandatory fields. Check if a user not entered a required field showing a mandatory error message.
4. Add and update information using form.
5. Tab orders.
6. Check for the default values of fields.
7. Forms are optimally formatted for better readability.
8. In negative testing, enter invalid inputs to the forms fields.

5.6. List various types of testing which is used in web testing.

Ans. **Types of testing used in web testing are :**

1. Functionality testing	2. Usability testing
3. Compatibility testing	4. Database testing
5. Crowd testing	6. Interface testing
7. Performance testing	8. Security testing

5.7. Explain cookies.

Ans. Cookies are the small text file that gets saved in the user's system. These files are saved in the desired location and are used by the browsers. Basically, they are used to maintain the session mainly login sessions. Instructive information is recorded in the cookie (like sessions) and that can be retrieved for the web pages. User can be able to enable or disable the cookies in browser options.

5.8. What is test HTML and CSS testing ?

Ans. To optimize website in the search engine then testing of HTML and CSS make important role. In this different search engines can crawl the site without any error. It should check for all syntax errors, colour schemas in the readable, check for standard Compliance and the standards are followed like W3C, ISO, ECMA, IETF, WS-I, OASIS.

5.9. What do you understand by link testing ?

Join Telegram Group :- @AKTU_Group Scanned by TapScanner

Join Telegram Group :- @AKTU_Group www.aktutor.in

Ans. Check for all broken links on your web pages & all links are working correctly. Along with, you can check the different links on your web pages :

1. Internal links
2. Outgoing links
3. Mail to links
4. Anchor links

5.10. What is performance testing in web testing ?

Ans. Performance testing is to check the web application that is working under the heavy load. Performance testing is categorized into two parts : web stress testing and web load testing.

Testing activities will include but not limited to :

1. Website application response times at different connection speeds.
2. Load testing, test your web application to determine its behaviour under normal and peak loads.
3. Stress testing test your website to determine its break point when pushed to beyond normal loads at peak time.

5.11. What points should be ensured while doing GUI testing ?

Ans. The following checklist will ensure detailed GUI testing :

1. Check all the GUI elements for size, position, width, length and acceptance of characters or numbers. For instance, you must be able to provide inputs to the input fields.
2. Check error messages are displayed correctly.
3. Check for clear demarcation of different sections on screen.
4. Check font used in application is readable.
5. Check the proper alignment of the text.
6. Check the colour of the font and warning messages is aesthetically pleasing.
7. Check that the images have good clarity.
8. Check that the images are properly aligned.

5.12. What is model based testing ?

Ans. A model is a graphical description of system's behaviour. It helps us to understand and predict the system behaviour. Models help in a generation of efficient test cases using the system requirements. **Following needs are to be considered for this model based testing :**

1. Build the model.
2. Determine inputs for the model.
3. Calculate expected output for the model.
4. Run the tests.
5. Compare the actual output with the expected output.
6. Decision on further action on the model.

5.13. List the methods of usability testing.

Ans. Methods of usability testing are :

1. Hallway testing
2. Remote usability testing
3. Expert review
4. Automated expert review
5. A/B testing

5.14. Explain hallway testing.

Ans. Hallway testing is a quick, cheap method of usability testing in which randomly-selected people, for example, those passing by in the hallway are asked to try using the product or service. This can help designers identify "brick walls", problems so serious that users simply cannot advance, in the early stages of a new design. The idea behind hallway usability testing began as an alternative to hiring trained or certified personnel to test a particular software or technology product. The idea is that you can go out and grab random individuals passing by an office in a hallway and get them to test a product being developed.

5.15. What is split testing or A/B testing ?

Ans. A/B testing or split testing is an experimental approach to web design (especially user experience design), which aims to identify changes to web pages that increase or maximize an outcome of interest (for example, click-through rate for a banner advertisement). As the name implies, two versions (A and B) are compared, which are identical except for one variation that might impact a user's behaviour. Version A might be the one currently used, while version B is modified in some respect. For example, on an e-commerce website the purchase funnel is typically a good candidate for A/B testing, as even marginal improvements in drop-off rates can represent a significant gain in sales.

5.16. Write the goals of usability testing ?

Ans. Goal of usability testing are :

1. Effectiveness of the system
2. Efficiency
3. Accuracy
4. User friendliness

5.17. What are the advantages of usability testing ?

Ans.

1. Usability test can be modified to cover many other types of testing such as functional testing, system integration testing, unit testing, smoke testing etc.
2. Usability testing can be very economical if planned properly, yet highly effective and beneficial.
3. If proper resources (experienced and creative testers) are used, usability test can help in fixing all the problems that user may face

Join Telegram Group :- @AKTU_Group Scanned by TapScanner

Join Telegram Group :- @AKTU_Group www.aktutor.in

even before the system is finally released to the user. This may result in better performance and a standard system.

4. Usability testing can help in discovering potential bugs and potholes in the system which generally are not visible to developers and even escape the other type of testing.

5.18. Write basic security concepts for security testing.

Ans. **The six basic security concepts are :**

1. **Authentication :** The origin of the application and its data is genuine.
2. **Authorization :** Specific users should only get access to authorized functions.
3. **Confidentiality :** Data/information is secure from theft.
4. **Integrity :** The application and its data are not altered in course of time during transmission.
5. **Non repudiation :** Guarantee that sender and receiver of information cannot deny having sent or received the data.
6. **Availability :** Assuring information and communications services will be ready for use when expected. Information must be kept available to authorized persons when they need it.

5.19. Write the attributes of performance testing.

Ans.

1. **Speed :** Determines whether the application responds quickly.
2. **Scalability :** Determines maximum user load, the software application can handle.
3. **Stability :** Determines if the application is stable under varying loads.
4. **Reliability :** Reliability refers to the consistency of a measure.

5.20. What is database testing ?

Ans. Database testing usually consists of a layered process, including the user interface (UI) layer, the business layer, the data access layer and the database itself. The UI layer deals with the interface design of the database. Database testing mainly takes place at this layer and involves testing strategies such as quality control and quality assurance of the product databases.

☺☺☺

Join Telegram Group :- @AKTU_Group Scanned by TapScanner