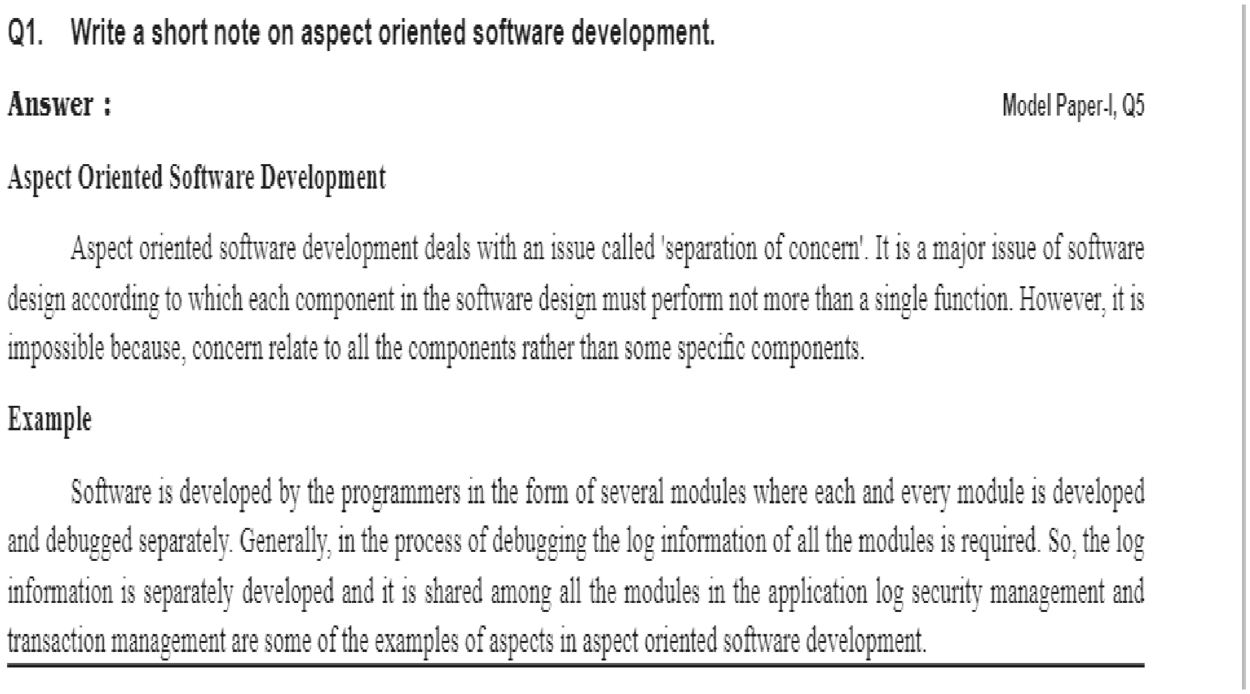
UNIT-3

1. write a short note on aspect oriented software development



Q1. Write a short note on aspect oriented software development.

Answer : Model Paper-I, Q5

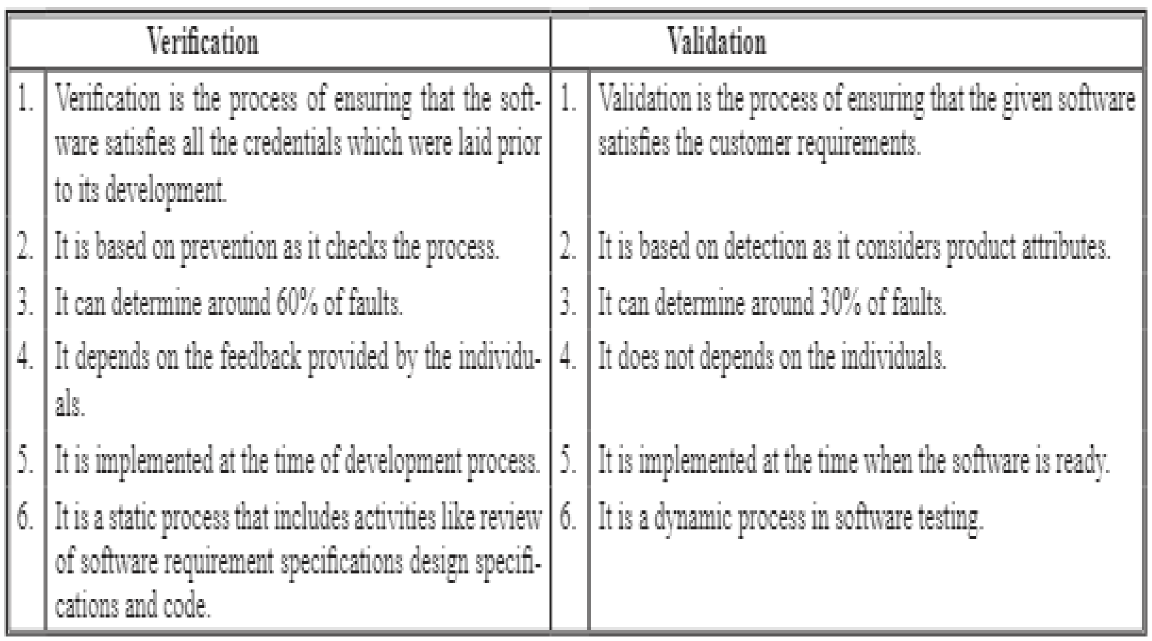
Aspect Oriented Software Development

Aspect oriented software development deals with an issue called 'separation of concern'. It is a major issue of software design according to which each component in the software design must perform not more than a single function. However, it is impossible because, concern relate to all the components rather than some specific components.

Example

Software is developed by the programmers in the form of several modules where each and every module is developed and debugged separately. Generally, in the process of debugging the log information of all the modules is required. So, the log information is separately developed and it is shared among all the modules in the application log security management and transaction management are some of the examples of aspects in aspect oriented software development.

4. Listout the difference between verification and validation



	Verification		Validation
1.	Verification is the process of ensuring that the software satisfies all the credentials which were laid prior to its development.	1.	Validation is the process of ensuring that the given software satisfies the customer requirements.
2.	It is based on prevention as it checks the process.	2.	It is based on detection as it considers product attributes.
3.	It can determine around 60% of faults.	3.	It can determine around 30% of faults.
4.	It depends on the feedback provided by the individuals.	4.	It does not depends on the individuals.
5.	It is implemented at the time of development process.	5.	It is implemented at the time when the software is ready.
6.	It is a static process that includes activities like review of software requirement specifications design specifications and code.	6.	It is a dynamic process in software testing.

Q5. What are the advantages to perform unit testing?

The advantages of performing unit testing are given below,

- It reduces the burden and overhead incurred by the developer.
- It facilitates for faster detection and prevention of bugs because only a part of the system is taken into consideration rather than the complete system.
- It provides a way for developing and testing the modules independently.

Q6. What are the methods used in the component testing?

Answer :

The developer uses two methods for testing the program in the component testing. They are,

1. Drivers

Driver is the module where the inputs of other modules are provided by writing a main program with desired inputs which may be hard-coded or feeded by the user when the provider module is unavailable.

In other words, it can be defined as a software module which is responsible for the following,

- Invoking the module to be tested.
- Providing the inputs to the testing module.
- Controlling and supervising execution.
- Reporting test outputs.

In addition to this, it also facilitates the unit under testing as it,

(i) Initializes the required environment for testing.

(ii) Provides the desired simulated input format to the testing unit.

2. Stubs

Stubs can be defined as a software module that acts similar to a module which is called by the module under test. The stubs are more simple compared to the actual module. It pretends to be a subordinate unit and service with the minimum desired functionality for that unit.

Q7. Define the terms,

(i) Error

(ii) Fault/Bug/Defect.

Answer :

(i) **Error**

The term "error" can be defined in two different ways,

(a) It is difference (discrepancy) between a computed, observed or measured value and the true, specified or theoretically correct value i.e., the difference that exist between the output of a software and the theoretically correct value.

(b) In general, the human action due to which there can be a defect in software is referred to as an error.

(ii) **Fault/Bug/Defect**

The situation that leads to system failure while executing a particular function is referred to as a fault. It forms the main reason for software malfunctioning.

Q8. Define failure.

Answer :

Failure is defined as the inability of a system or component to perform as per the system specifications. The occurrence of failures may be due to the functional or performance factors. A failure can be recognized by the behaviour of the software that is different from the specified behaviour. Some times it happens that the failure occur but are not detected.

The terms error, fault and failures are interrelated to each other. Since the presence of error indicates that some failure has occurred, and also ensures that some fault is present within the system.

Q9. List and explain the classification of bug.

Answer :

The bugs are classified into three types. They are,

1. **Critical Bugs**

   Critical bugs are the bugs which may bring great financial loss to the organization if not attended immediately.

   Example

   Miscalculation of fares in railways e-TRS may end up in high financial loss.

2. **Moderate Bugs**

   Moderate bugs are the bugs which bring temporary financial loss to the organization. They are less severe in damage.

3. **Simple Bugs**

   Simple bugs are the bugs which does not affect the functionality i.e., behavior or continuity of the software.

Q10. What are the types of process used in software product development?

Answer :

The following are the types of processes used for the software product developments.

1. **Informal Process**

   Here, standard process are not followed because the development team creates their own processes. Even though, it is said as informal, the steps adopted are same as that of formal procedure such as black box testing. This distinction is generally drawn out by the team using general purpose team.

2. **Managed Processes**

   Here a well established process model is adopted by the organization in order to perform software development activity. It consists of well-defined procedures, with sequential execution and relationship between multiple processes. Also, it makes use of tools to carryout configuration and project management.

3. **Methodical Process**

   Here, a well-defined methods are adopted along with these supporting standards. For example, UML diagrams used in object oriented Analysis and design, this sustained by automated tools.

4. **Improving Process**

   The organization implements unique processes for enhacing the product with in specified budgets. Here, a specific quantitative improvement measurement model is used so as to determine the changes occurred on the productivity of people.

Q11. What is software development? What are the key objectives of good software development methods and how these objectives can be achieved?

Answer :

Software Development

Software development is a phase of software engineering where designing and coding/programming of applications are performed.

Key Objectives of Good Software Development Methods

The key objectives of good software development methods are,

1. Developing a code that is easy to understand.
2. Developing a code that can be reused.

The above objectives can be achieved by following different approaches such as use of good coding practices, code reusability and concept reusability. Good coding practices makes the developed software easy to maintain and reuse.

These practices increases level of granularity for reusing the developed software. Code reusability can be supported at object and function level through the use of object-oriented software design. However, this method offers low-level reusability. To increase the level of reusability, methods like product lines approach are preferred. This method makes the software available as a product or component that can be reused directly or with a little modification. Most of the modern applications such as e-TRS follows this approach.

The level of reusability can also be increased by making the reusable design patterns instead of just coding.

Q12. Why good coding practices are used? Discuss some of the good coding practices.

Answer :

Good Coding Practices

Good coding practices are used in almost all the programming languages. The usage of these coding practices maximizes the software understandability and maintainability. As a result, the overall operational and maintenance cost is minimized.

The following are some of the good coding practices that are used in programming languages,

(i) Meaningful Variable Names

Meaningful names must be given to the variables which are used in the functions to easily understand their purpose, data type and parameter type.

Example

In e-TRS(ticketing system), a variable can be declared as Int_In_Train number in interface class Journey Type. The variable name itself indicates the following,

(a) Data type - Integer(Int)

(b) Parameter - Input(In)

(c) Trainnumber

(ii) Display Correct Messages

Displaying the correct messages is important, because if a user submits a form with errors, it is necessary to display the erroneous field and the correct error type.

(iii) Exceptions and Error Handling

Error handling method should return short error messages. Such messages do not provide enough information to hackers and hence, they fail to identify the reason and function which is generating the error.

(iv) Detailed Documentation

The documentation must include detailed information so that the developers who were not involved in coding can also understand the use of each and every operation including its functionality and input-output parameters.

(v) Parameter Driven Approach

The parameters that are to be used in computation are not fixed and they can be changed. Hence, this calls the need for addressing it in the configuration along with their values. This approach is called parameter driven approach and its usage increases the code readability and decreases the need for changing the code.

Example

In e-TRS(ticketing system) the fare is calculated using the formula,

Fare = Basefare + Distance * Rate/Kilometer

Here, Basefare is expected to change, so it must be stored in configuration file.

(vi) Consistent Formatting

Consistent formatting refers to aligning of code using tabs, punctuations, blank lines, comments. So that the readability of code can be increased.

(vii) Create Separate Folders

Separate folders must be created to store source code, header files, object code, executables, build utilities, documentation and scripts, inorder to maintain and administrate source code easily.

(vii) Simple and Effective Code

The source code should be simple and effective that does not include complex programming constructs like pointers, bits operations, heavy structures, deep if then else conditions and deep hierarchy. Such constructs makes it difficult to understand and maintain over the long term.

Q13. Write short notes on code reuse.

Answer :

Code Reuse

Code reusability in software engineering is a software development methodology. It helps in increasing the amount of reuse for existing software units. These software units can differ in their sizes, where each unit can be an application, component (or) object and function.

Code reusability is used in the following levels,

(i) Application Level

Code reuse at application level can be performed using the following approaches,

(a) In the first approach, the application is configured according to various customers during the time of deployment. This approach is used when there is no need to modify application to be reused.

(b) In the second approach, a group of applications with similar architectural design are developed and they are customized according to the customers.

This approach is subdivided into the following.

1. Commercial off the shelf (COTS) based application development.
2. Software product line application development.

Example

Consider COTS (Commercial off the shelf) product which helps in the implementation of functionality quickly without involving increased cost. In such a product application reusability can be performed as shown in the figure.

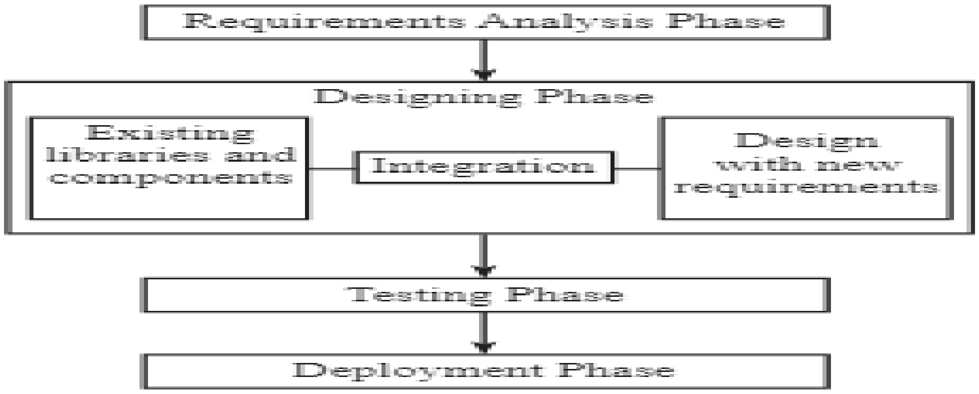

Requirements Analysis Phase
Designing Phase
Existing libraries and components
Integration
Design with new requirements
Testing Phase
Deployment Phase

Figure: Phases in Application Reusability

Suppose, if a microsoft module on Human Resource has to be used, then it may require customization. This customization might involve customizing the caste based Roster management and Income Tax according to the Indian Rules. So, it is necessary to check the ease of customization. Which generally involves the inclusion of additional functionalities into COTS products.

(ii) Component Level

Code reuse at component level includes the reuse of all the components from highest level to the lowest possible level (i.e., from subsystem to an individual object) in an application.

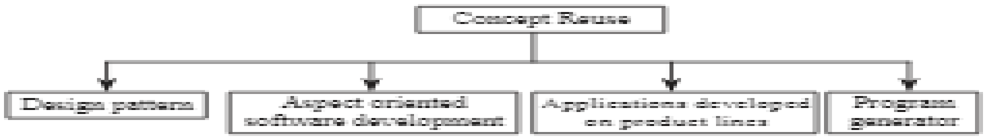
(iii) Object Level

Code reuse at object level deals with the reuse of components that perform a specific function like any mathematical function or object classes. Here, the creation of standard libraries of classes and functions associated with various application and OS platforms are performed. These libraries can be used by the programmers by simply linking them with their application codes. Mathematical libraries, graphic libraries, engineering libraries are some of the example of standard libraries. Developing all these libraries requires good knowledge and skills as it involves development of functions, objects and classes. Thus, it is highly useful and can be reused at higher level.

Q14. Discuss the concept of reuse methods.

Answer :

Concept reuse is one of the alternative form of reuse. It uses abstraction pattern instead of concrete elements. The pattern offers well defined and reusable solution for the set of problems that usually occur. It consists of problem description and the nature of solutions to those problems. These solutions are considered as a result obtained from expertise and experience. Concept reuse can be represented in four ways,


Concept Reuse
Design pattern
Aspect oriented software development
Applications developed on product lines
Program generator

(i) Design Pattern

Design pattern describes abstract objects, concrete objects and interaction among these objects. One of the examples of such patterns is application framework which is defined as a collection of classes (i.e., both abstract and concrete). These classes are then subjected to design application systems.

(ii) Aspect Oriented Software Development

Aspect oriented software development method is used to build certain components in advance so that they can be shared at all parts of the development. Log information, security and transaction management are some of the examples of aspects or features that can be shared.

(iii) Applications Developed on Product Lines

Applications are developed as a common architecture using product lines approach. Such an application can be customized accordingly based on different user requirements.

(iv) Program Generator

Program generator is defined as a software program which allows user to create their own program. It is an easy method of reuse because a user need not have to write code. The program generator can create either the entire application or a partial application which can be developed further. The programmers just need to fill specific details inorder to develop complete application.

Q15. Discuss the use of design patterns in reusability. Also discuss the implementation of factory design pattern.

Answer :

Use of Design Pattern

Reusability can be increased by using some of design patterns like strategy design pattern and factory design pattern. Use of design pattern for reusability motivates the concept reuse.

Abstract classes specifies the procedure to develop an application for some particular domain. The concrete classes are inturn developed from abstract classes. Moreover, the sub classes are also derived from base abstract class.

Example

Consider that an application need to be developed in such a way that it can be implemented at all places of India i.e., all states and union territories. To develop such an application, an abstract class framework must be created. The functionality which is common in all the states must be developed centrally while the functionality which is specific to a particular state can be developed in that state only. The latter one can be obtained by organizing the details into subclasses. These subclasses are derived from base abstract class.

The concept of factory design pattern and strategy design pattern can be illustrated using web resources, coding practices of microsoft and other independent contributors.

Factory Design Pattern

Factory design pattern consists of the following components which are used for building products.

(i) Product

It acts as an interface used by all the other products.

(ii) Creator

It acts as an abstract class.

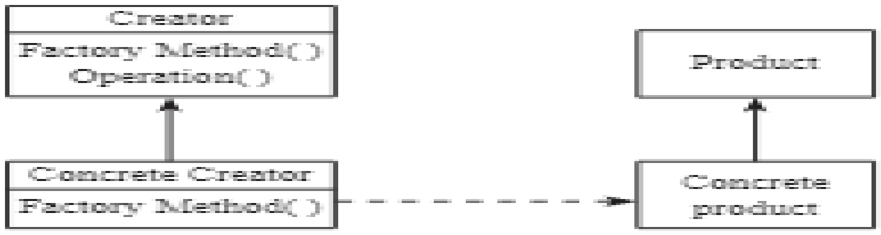


Figure: Implementation of Factory Design Pattern

In this pattern, the concrete product is obtained from the implementation of the product interface. Moreover, the create class carries factory method that does not contain any implementation. Concrete creator implements factory method by inheriting it.

Q16. Write a brief note on strategy design pattern.

Answer :

Strategy Design Pattern

The strategy design pattern is used when a client needs to choose a single algorithm from a group of algorithms. This pattern consists of a strategy interface. This interface is common for all other and hence it is shared among all the components. Strategy interface can also be called as 'compositor'.

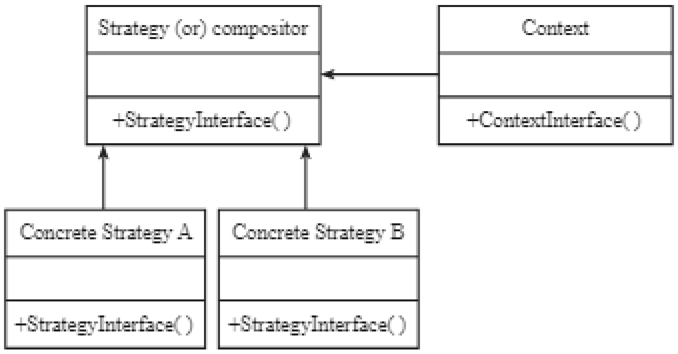


Figure: Implementation of Strategy Design Pattern

The 'compositor' class is common for both 'Concrete with GST' and 'ConcreteWithoutGst' classes. Generally, strategy design pattern considers HAS-A relationship which is better than IS-A relationship which makes its composition better than inheritance. Let 'strategy parent' be the common strategy interface used in context class. This interface also handles the object referencing.

The following program shows the implementation of strategy design pattern.

```
namespace BehavioralPattern.StrategyPattern
{
public abstract class StrategyParent
{
Public abstract long compute(int actualprice, int Gst);
}
public class ConcreteWithGst : StrategyParent
{
public override long compute(int actualprice, int Gst)
{
return actualprice + Gst;
}
```

What are Agile Methods? List and explain the principles of agile methods.

Answer :

Agile Methods

Agile methods refer to the iterative approach used for gathering software specifications, developing and delivering the software.

Agile Principles

Agile methods follows a common set of principles which are as follows,

(i) Stakeholders engagement

(ii) Iterative and incremental delivery

(iii) Flexible and open to changes

(iv) Concentrate on people

(v) Follow simple design.

(i) Stakeholders Engagement

Important stakeholders are involved during the development so that they can provide new requirements and features and give priorities to them. Prioritizing is not an easy task because each and every stakeholders give different priorities to different requirements.

(ii) Iterative and Incremental Delivery

The software is developed and delivered on incrementally. In every increment, the customers suggest to include specific requirements.

(iii) Flexible and Open to Changes

The developed software must be flexible and open to changes. So that the system can be modified accordingly.

(iv) Concentrate on People

The members of development team must be given freedom, so that they can work in their own way instead of following a specific process. With this, their skills can be identified and utilized in the software development.

(v) Follow Simple Design

Simple design principles and practices must be followed during software development. The complexities (if any)

Discuss the extreme programming practices that fit in principles of agile methods.

Answer :

Extreme Programming Practices

Extreme programming practices that can fit in principles of agile methods are as follows,

(i) Step by Step Planning

The user and system requirements are gathered and written in story cards. Using these story cards, the development is planned in a step-by-step manner in order to develop the product on time, so that they can be included in the latest released.

(ii) Frequent Releases

The changes are included in different releases where the first release includes minimum functionality. This release focusses on business value of the organization. Later on, additional functionalities are included with frequent releases. These frequent releases supports incremental development.

(iii) Simple Design

The developed design must be simple enough to meet only the necessary requirements planned for current release.

(iv) Develop Test Cases First

The developers need to write text cases for new functionality. Prior to the developmental activities.

(v) Refactoring

The developers use refactoring technique to make the code simple even when modifications are performed. Refactoring is performed whenever scope for improving the product is found.

(vi) Pair Programming

Two developers are involved in developing the software programs. Both these developers simultaneously work at a single workstation. They review their work and support each other to enhance the product quality.

(vii) Collective Ownership

The pair of programmers work on any module and they are allowed to change the code at any point to improve its quality.

What is software prototyping?

Answer :

Software Prototyping

A prototype is considered as a sample version (i.e., initial version) of a software.

It is generally used for the following purposes,

(i) Describing the concept.

(ii) Trying various design options.

(iii) Obtaining more information about specifications.

(iv) Discovering all the solutions that are possible for a given problem.

A prototype must be developed with repitative and quick construction so that it can also help in the following,

(a) Controlling the Cost.

(b) Enabling the stakeholders to perform trials on λ prototype.

The software prototype is used in the following phases of software development,

(i) Requirements Engineering Phase

The IT engineer analyses the prototype and performs the following activities,

- Validates the requirements that are included in the system.
- Gathers additional requirements from the business stakeholders.

(ii) Design Phase

The prototype can be used in design phase in the following ways,

(a) It can be used for software evaluation.

(b) It is much better in representing user requirements when compared to textual content and diagrams representing user requirements.

(c) It can be used to include graphical user interface for the system while involving the users during its development.

(d) It can help the designers so that they can finalize the user interface design.

(iii) Testing Phase

The software prototype can be used in testing phase in the following ways,

(a) Prototype is used to execute test cases of each and every stakeholder and ensure their satisfaction with respect to software quality.

(b) It identifies the faults in the system by comparing the results of full fledged system and prototype.

(c) It evaluates the reasons behind the occurrence of fault.

(iv) Implementation Phase

The software prototype can be used in implementation phase in the following ways,

(a) Prototype is used to train the end users

(b) It is used for developing user documentation.

Stages in Prototype Development

The stages in prototype development are as follows,

(i) Specify Goals and Objectives

The goals and objectives of prototype must be specified as a first step. This is because, the end users may sometimes misinterpret the prototype functionality and fail to gain full benefits if the goals and objectives of prototype are not specified clearly.

(ii) Understand Prototype Functionality

The functionality of the prototype must be understood to identify the things to be included and excluded with in the prototype. Some functionalities are removed to control the cost and to deliver the product quickly. Moreover, some requirements are also ignored which include response time, memory utilization, error handling and management can be ignored. A part from this, the requirements such as reliability standards and program quality can be minimised.

Discuss in detail about the verification and validation.

Answer : Model Paper-III, Q11(a)

"Verification" and "validation" are two important aspects of the software testing process. Though these two phrases seems to have similar definitions, there exists huge difference between them. Verification is the process of ensuring that the given software satisfies almost all the credentials which were laid prior to it's (software's) development and validation is the process of ensuring that the given software satisfies the customer requirements. Hence, in short we can refer both of them as,

Verification

It checks whether a product is being built in the right way.

Validation

It checks whether the right product is being developed.

Hence, in order to ensure that the given software is correct in all aspects, then it has to satisfy the verification and validation process successfully.

Verification and Validation (or) V and V activities for developing a software include,

(i) Formal Technical Reviews (FTR)

(ii) Performance monitoring

(iii) Quality and configuration audits

(iv) Feasibility study

(v) Simulation

(vi) Algorithm analysis

(vii) Database review

(viii) Documentation review

(ix) Usability, qualification and installation testing.

The equation depicting the relation between verification and validation and testing is,

$$T = V_e + V_a$$

Where,

T – Testing process

V_e – Verification

V_a – Validation.

However, testing is highly validation oriented.

Explain in detail about the faults that arise in program inspection process.

Answer :

The commonly occuring faults in the program inspection process are as follows,

(i) Faults of Data Initialization

This type of fault arises when the value of the declared variable is not initialized properly. It must also verify the proper definition of the upper bound on array for storing the elements in the array. Moreover, it must describe the size of the variable in order to prevent from the issues related to buffer overflow.

if-then, else, while or for loop are not defined or used inappropriately with in the program. In such situations, the faults must be rectified by checking that they are correctly terminated. In case of switch statement, it must be checked that, whether the cases are properly defined or not and uses break statement at the suitable places.

(iii) Faults Related to Reading Input and Computing Output Variable

This type of fault arises when there exist inefficiencies in reading of input variable or computing the output variables. In this case, types and values of the input variables to be read need to be interrelated with one another. Moreover, the output variables should also be examined to know whether they compute and assign accurate values for displaying on the screen or writing in the database.

(iv) Faults Related to Interface Parameters

This type of fault arises when the calling and called function parameters are different in terms of number, datatype and order. Also, it must be verified that there exist common data structure of shared variables that are used in all the programs. This can be accomplished by defining a data structure in the header file.

(v) Faults in Memory Allocation or De-allocation

This type of fault, verifies whether the space allocated in the dynamic memory is accurate or not. Once the purpose of allocating dynamic memory is completed then the memory is deallocated.

These type of faults are resolved automatically by acquiring and releasing dynamic memory using Java.

(iv) Faulty Exception Handling

This type of fault arises when the overall conditions of error are not specified. This leads to the generation of unanticipated errors resulting in program failure.

developed program.

Q35. Define cleanroom process and discuss its five main strategies.

Answer :

Clean Room Testing

Clean room testing is a manual verification technique which was introduced by Mills Dyer and linger in IBM. The main goals of clean room development are,

(i) To prevent errors

(ii) To decrease costs

(iii) To increase reliability.

time taken to conduct code walkthroughs code inspection and formal verification is too long.

Strategies in the Cleanroom Development

The following are the important strategies of cleanroom development method.

(i) Formal Specifications

This strategy uses state transition model for representing software specifications in the formal form. It also indicates responses of the system to stimuli and events.

(ii) Incremental Development

This strategy splits the software into increments which are initially defined along with the customer input. Basically, this method is used for delivering critical functionality in the initial stage so that it becomes less complex in the later stages. Also, it permits the customer to take a trial before releasing the complete software. Moreover, it facilitates the customers to give feedback to the development team based on critical increments. As a result, the development team can focus on making the software perfect with respect to the critical functionality with every release.

(iii) Structured Programming

This strategy uses a specific amount of control constructs and data abstraction constructs so as to divide the specifications into detailed system models until it develops an executable program.

(iv) Static Verification

This strategy verifies the software after its development using software inspection. It does not include code testing. Apart from this, each stage develops a detailed system model representation by implementing well defined transformations in order to assure the accuracy of the individual stages. Moreover, it checks these individual representations and develops mathematical arguments that describes the consistency of generated output with the input supplied.

Q36. Discuss the goals and objectives of testing process.

Answer :

Goals and Objectives of Testing Process

Software testing is performed inorder to identify and describe errors generated by lack of human skills or caused by bugs in the system program such as compilers, languages, databases and operating system. Apart from this, there exist few errors which are not found during reviews are easy to identify at the time of testing.

Moreover the testing is carried out so that the customers after receiving the product should not encounter any faults. Since testing is performed at initial levels it minimizes the rework time, such as post release debugging.

The two goals of software testing process are as follows,

(i) **Demonstrating that Software Meets Requirements**

The requirements included in user and system requirements document must be tested atleast once in case of customized software. However, the entire features of the system must be tested in case of generic software products. An explicit acceptance testing phase is specified in the e-governance application for appointing the users from line ministry wherein they formally verify the specifications assured by the system.

Q37. Explain testing strategy.

Answer :

Testing Strategy

Software testing strategy provides the guidelines that give an outline of the following,

- What steps to follow in order to undergo a test?
- When to plan and workout the steps?
- How much time, input and resources must be utilized?

Hence, software testing strategy must include a proper planning for the test, a design for the test case, test execution and resultant data collection and evaluation. It must be able to promote a customized testing approach and also encourage planning and management tracking along with the increase in project progress.

Software development begins at system engineering, defines the role of the software, and ends at coding. Testing begins at coding phase and ends at system engineering phase. At each level of testing, a different testing method is adopted. These methods are,

(i) **Unit Testing - At Coding Phase**

As soon as the code is developed, each unit or component in it is tested individually. Such testing is called unit testing. Unit testing ensures that maximum errors are detected and entire code is tested, by following certain paths in the control structure of a component (unit).

(ii) Integration Testing - At Design Phase

The units in the developed code are combined or integrated to result in a single package. Integration testing thus tests whether the integration results in any more bugs or not. It concentrates on software architecture's design and construction. Maximum code coverage is ensured by following certain control paths in the program.

(iii) Release Testing

Once these testings are completed, the developed system is given to the user so that they can perform acceptance testing. During this process, the domain experts help the users testing team to develop test cases, perform system testing and store the results of the test records. If at all, the bugs are encountered while executing the system, those bugs are reported to the software development team. The software is tested repeatedly by the testing team until all the bugs are fixed and no bug remains in the system. This process is known as Release Testing.

Q38. Discuss various types of system testing.

Answer :

Some of the different types of software testing are as follows,

(i) System testing

(ii) Alpha and Beta testing

(iii) Acceptance testing.

(i) System Testing

System testing exposes bugs that are not resulted from the components or from the inconsistencies between the components. It is a black box functional testing used to test the entire software system. It is performed to show the behavior of the system. System testing is done either on the whole system that is integrated or only on the important parts of a system. During the test design, it ensures the systems's behavior as per the requirements specification. Testing is performed for the applications like accountability, security, performance, sensitivity, configuration, start-up and recovery.

(ii) Acceptance Testing

It can be defined as the process of testing which is responsible for the assurance of the following,

- Fulfillment of acceptance criteria with respect to system.
- Acceptance of the system with respect to the buyers.
- Users confidence about the system.
- Fitness of the software for the users.

The purpose of this testing is to get comments from the end-user to the development team about whether or not the software has met their specified requirements. It is the most essential testing applied to a product since it involves the user satisfaction otherwise, the software may not be accepted.

(iii) Alpha and Beta Testing

Acceptance testing proves to be an efficient form of testing when the product is developed for a single user. However, if there are more than one user, then other types of testing – alpha and beta testing must be used.

(a) Alpha Testing

During the alpha testing, several users are called to the developer's site and are exposed to the product. Under the guidance of the software engineer, the user uses the product, in the mean while, the software engineer notes all the possibilities like, the problems faced by the users during normal operations, the instructions that are difficult to understand, the operations generating bugs, etc.

(b) Beta Testing

During the beta testing, the product is delivered to the users. Here, in the absence of the software engineers, the product is tested and a report is prepared by the user. The report contains details of the difficulties faced by the users during the product operation. After analyzing the report, the software engineer makes suitable modifications to the software and then supplies it back to the users.

Q39. What is unit testing? Also, discuss the principles involved in it.

Answer :

Unit Testing

A unit is the smallest piece of source code that can be tested. It is also known as a module which consists of several lines of code that are processed by a single programmer. The main purpose of performing unit testing is to reveal that a particular unit doesn't fulfill the specified functional requirements and also to show that the structural implementation is not similar to the expected structure designed.

Unit tests can be both static tests and dynamic tests. At first, static tests are performed followed by the dynamic tests to check the test paths, boundaries and branches. Most of the unit tests are dynamic white box structural tests. These tests require either the execution of the software as a whole or parts of software. If a bug is revealed while performing the unit test, it is referred to as a unit bug.

The following are the principles that has to be included in the unit testing.

(i) Module Interface

This interface is responsible for passing the data in and out from the module. It ensures that caller and called functions do not face any problems that are associated with data types, data size etc.

(ii) Data Structure

The local variable information that is stored with in the function must not be distributed while executing the function.

(iii) Boundary Value Condition

Sometimes, the modules may not function properly at the point of boundary values, restricted values and extreme limits. These values should include data values such as minimum values, maximum value and value more than minimum value, value less than maximum value and the value that is in between minimum and maximum value.

(iv) Robust Testing

If additional input sets are designed in the boundary value testing. Then it turns out to be Robust testing. The additional input sets could be,

- ❖ Value more than minimum value.
- ❖ Value more than maximum value.

As a result, the number of test cases in robust testing are (6n + 1) wherein n = The number of input variables.

Q40. Explain in detail about white box testing.

Answer :

White Box Testing

White box testing deals with the internal logic and structure of the program code. It is also known as glass-box, structural or open-box testing. In glass-box testing, test cases are derived based on the knowledge of software structure and its implementation. The tester can analyze the code and make use of the knowledge about the structure to derive test data. Using white box testing, the tester can detect which module or unit is not functioning properly. This test is performed depending on the knowledge of "how" the system is implemented. White box tests can analyze data flow, control flow, information flow, exception and error handling techniques. These techniques are used to test the behavior of software. White box testing is done to check if the implementation is in accordance with the planned design. It is also used to check the implementation of security functions and to explore the risks.

This technique is used to ensure that independent path and the statements are executed for a minimum of one time in a module.

Error Handling Path

These paths are used in cases where the business logic generates error condition or error messages.

Commonly Encountered Errors

The errors that can be easily encountered while executing the path are incorrect declaration of arithmetic precedence, initialization of incorrect values, specification of incorrect precision and rounded off errors, comparison of various categories of Data types, no termination of loops, violations of boundary values, no invocation of error handlers and so on.

In comparison to black box testing, white box testing is more complicated. Due to its complexity, the testing team members need to perform a detailed study on the internal structure, algorithm and implementation details of a program

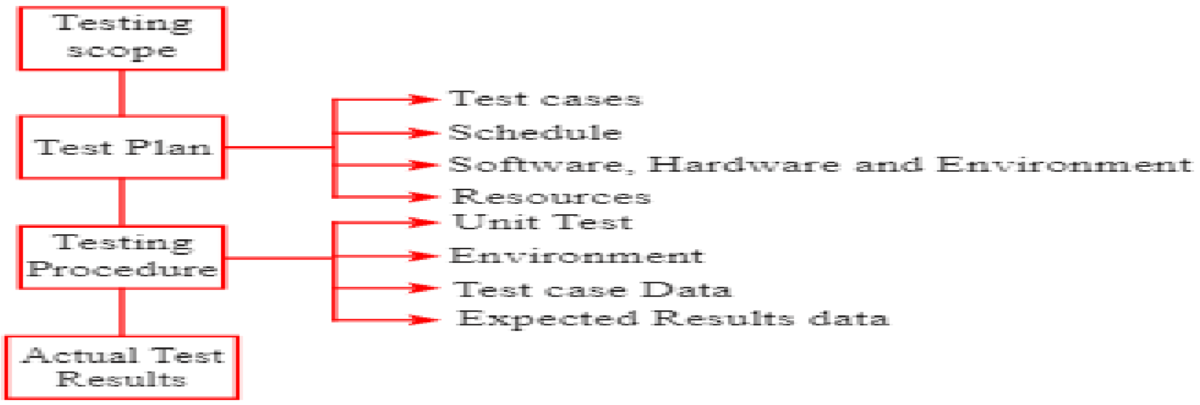
Q41. Discuss in detail about Integration Testing.

Answer :

Integration Testing

Integration testing integrates (assembles) all the modules of a system and realizes them as a complete system. Then it performs a test on the module interfaces during the parameter passing in the module and ensures that there are no errors. A test is performed after each successful integration of module. The test plan for integration testing is given by the module dependency graph.

The following figure illustrates the integration testing plan.



Strategy for Integration Testing

The integration testing strategy can be implemented in two ways,

1. Non-incremental integration testing
2. Incremental integration testing.

1. Non-incremental Integration Testing

Non-incremental integration testing method integrates all the modules of a system together and performs a test on

Types of Incremental Integration Testing

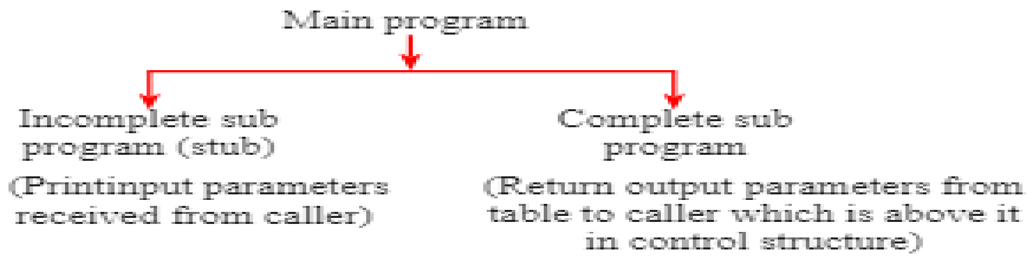
The incremental integration testing has been categorized into two types,

(i) Top-down integration testing

(ii) Bottom-up integration testing.

(i) Top-down Integration Testing

In this method, the modules are integrated from top to bottom i.e., from high-level modules to low-level modules in the design hierarchy.


Main program
Incomplete sub program (stub)
(Printinput parameters received from caller)
Complete sub program
(Return output parameters from table to caller which is above it in control structure)

The subordination of modules to the high-level modules is performed in either of the following ways,

(a) Depth first integration

(b) Breadth first integration.

(a) Depth First Integration

Here, the subordination (integration) of the modules is done by considering the longest control path of the design hierarchy from top to bottom. For instance consider the design hierarchy,

consider the design hierarchy,

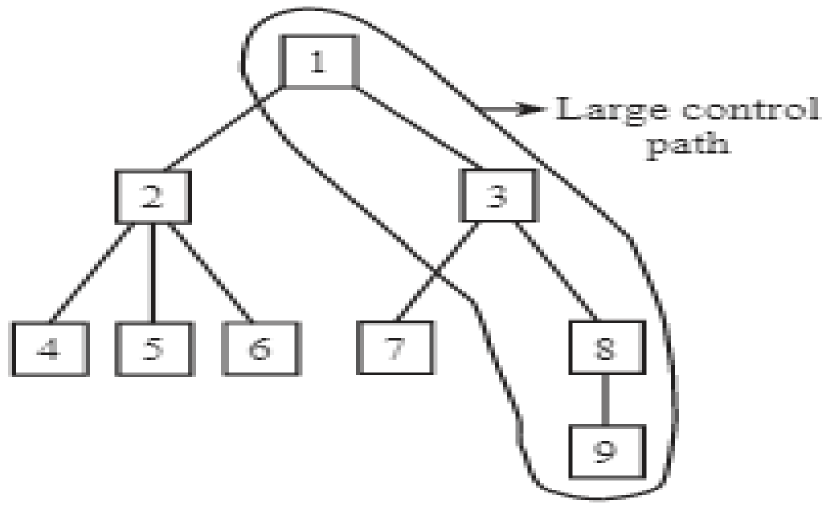


Figure: Depth First Integration

Q42. Write notes on,

(i) Regression testing

(ii) Release testing.

Answer :

(i) Regression Testing

Regression testing is a type of testing that performs an old test suite after each modification on the program or after each fix of the bug. The result of this ensures an error free program with all the bugs fixed with no addition of new bug. It is not necessary to perform the entire test suite when only few modifications are done. Only those functions that are expected to be affected are tested. The test cases pertaining to those functions are only conducted.

The following are the different test case classes which are found in the testing suite of regression.

Q43. Explain various types of system testing.

Answer :

System Testing

The various types of system tests include the following,

1. Recovery testing
2. Security testing
3. Performance testing
4. Stress testing.

1. Recovery Testing

It can be defined as a process of testing which ensures recovery of system from failures. For this purpose, the system is subjected to forced failures of different types. Some of the forced failures are given below,

1. Forcebly restarting the computer when the application is running. This helps in capturing how long the data is integrated.
2. Forcebly unplugging the cable while the application is receiving data from the network. This helps in examining the capability of application in resuming the data receiving operation.
3. Forcebly restarting when the browser is running multiple sessions. This helps in examining the capability of browser to resume its operation after rebooting.

Recovery of a system can be accomplished by employing certain ways like use of,

- Automated recovery system,
- Checkpointing
- Using Number of CPUs or number of device instances and failure detecting mechanisms.

The checkpointing system logs the information periodically regarding transactions and their states.

2. Security Testing

It can be defined as the process of testing which ensures the correctness of system security. Typically, security is associated with vulnerabilities and risks that exist in the system.

The security team should perform certain operations to handle the software security risk/vulnerabilities. Some of them are,

- ❖ Employ static analysis tools.
- ❖ Conduct security test.
- ❖ Conduct architectural risk analysis.
- ❖ Construct risk-based security test plans.
- ❖ Develop usecases for security misuse.
- ❖ Record security requirements.

Security Testing Guidelines

The security testing team must employ a risk-based strategy in both the architectural reality and the attacker's mindset aspects.

Concentrate on the parts of code which can be an attacked easily. For this purpose, findout the risk and potential loss linked with that risk within the system and then generate the tests.

Thus, the risk analysis for the security testing team acts as a tool in finding potential security problems and their affects.

The security testing must perform the following types of tests.

- ❖ Testing security mechanisms so as to guarantee that they have been developed correctly.
- ❖ Testing risk-based security so as to guarantee that the attackers cannot harm the security of the system.

3. Performance Testing

It can be defined as the process of testing where in run-time performance of the system is checked in accordance to the different performance factors. The realtime embedded system necessarily requires performance testing as they involve critical performance requirements.

This testing offer clear, precise and quantifies specification of performance requirements in the SRS and system test plans. Quantity specification for performance requirement is very much essential. For instance, the amount of time needed for a system to provide response to a query cannot be specified as "Desired time" rather, the time must be quantified.

The performance test plan at the time of requirements development stage records the performance requirements. These requirements should include the following details,

- ❖ The time needed for any or all backend batch processes.
- ❖ Description of performance test scope along with the inclusion and exclusion of the subsystems, interfaces, components etc., in the scope of performance testing.
- ❖ The appearance of software system such as configuration associated with network and server devices.
- ❖ Number of simultaneous users expected for the user interfaces, present in the system.

It can be defined as the process of testing where the system under test in forceably broken by flooding its resources so as to determine critical situations that leads the system to crash. The following are the parts of the system which may suffer from such situations.

- ❖ Disk space
- ❖ Output
- ❖ Input operation
- ❖ User interactions
- ❖ Communication.

However, stress testing is essential for the real time systems incase of unexpected occurrence of events, leading to input loads reaching beyond the values mentioned in specification and to failure in system due to such load on resources. Thus care must be taken to note the complete threshold and system units for the real time systems. Stress testing is then applied on every single value.

Stress testing requires specific time for the test preparation as well as time specific amount of resources utilized during the test execution. Thus stress testing damage must be measured so as to prevent from the loss that can not be determined. For instance, the real time defences system need to work upon warfare conditions continuously. In such situations the real time defence system must be stress tested so as to prevent from loss of system as well as the life.

Q44. Discuss in detail about object oriented testing strategies and its types.

Answer :

Object Oriented Testing Strategy

Softwares developed using object-oriented approach are significantly different than their procedural counterparts (or non-object oriented applications). This is because, object oriented technique comprises of features such as inheritance, polymorphism, encapsulation, abstraction etc., which do not allow the usual software testing techniques to work efficiently. Therefore, it is important to employ an object oriented testing method which will provide special testing support for manipulation that are casual to the object-oriented features.

Uses of Drivers

1. Drivers perform testing of operations at lowest level.
2. They perform testing of entire group of classes.
3. They can be used for replacing the user interface inorder to perform testing on system functionality before implementing the interface.

Use of Stubs

Stubs are used when there is a need to collaborate classes provided that these classes are partially implemented.

3.2.3 Test Cases, Equivalence Partitioning (Black Box Testing), Art of Debugging

Q45. What are test cases? Also, discuss the guidelines for test case design.

Answer : Model Paper-I, Q11(b)

Test Cases

A set of test inputs and conditions based on which a software is tested is referred to as test case. A test case contains two types of information.

1. **Set of Inputs**

   These are of two types,

   (i) Precondition and

   (ii) Actual input.

2. **Expected Outputs**

   These are of two types,

   (i) Post condition and

   (ii) Actual outputs.

The test data which is specified in the input can be used for verifying the system. This test data can be created automatically. But, the test cases which are to be used cannot be automatically created. Their creation needs experience and knowledgeable domain people as they need to obtain expecting the output of the system.

Guidelines for Test Case Design

Answer :

Equivalence Partitioning

Equivalence partitioning can be defined as the process used to determine the test cases using classes of input condition called equivalence classes. These classes are identified in such a way that each member present in the class carry out the same processing and provide the same output. This method eliminates the need for testing the entire input domain rather it tests one test case from each equivalence class. This itself provides the list of bugs (or) errors. It works in such a way that if one test case is successful in providing the bugs then the remaining test cases will also have the ability to reveal the bugs that are present. Therefore by using minimum number of test cases only the tester will be able to track down all the bugs. This improves the hit rate.

Goals

The goals of equivalence partitioning are,

(i) **Completeness**

Equivalence partitioning aims at obtaining completeness in testing the input domain.

(ii) **Non-redundancy**

Another goal of equivalence partitioning is to reduce the number of redundant test cases. If such cases are present then they lead to wastage of time, space and resources. This method can be used by performing the following two steps,

(a) Identification of the equivalence classes

(b) Designing the test cases.

Q47. Explain the guidelines defined for forming equivalence classes.

Answer :

Equivalence Classes

The equivalence classes can be formed based on the behavioural patterns of the classes present in the input domain. For instance, consider a module that gives the absolute value of integers so this module consists of the specification that consist of different behavioural patterns for positive and negative integers. Hence in this case there are two classes. One for positive integers and the other for negative integers. There are two types of equivalence classes,

(i) **Valid Equivalence Classes**

These are the classes that accept valid input to program.

(ii) **Invalid Equivalence Classes**

These are capable of even considering invalid input to program.

Input

Q50. With the help of a diagram, explain the process of debugging.

Answer :

Debugging

Once testing is successfully carried out and bugs are uncovered, the next step is the removal of these bugs and is named as "debugging". The process of debugging is depicted below,

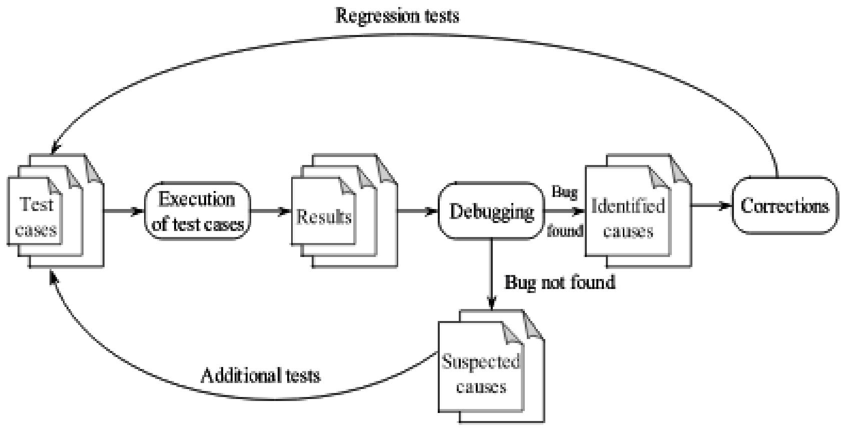


Figure: Debugging Process

The test cases are executed and the results are gathered. These results are compared with the expected results. Any variation between the two means a bug. The cause of the bug may be either identified or it still remains hidden. In the second case the professional performing debugging may suspect the cause, design test cases for it and repeatedly execute these test cases till the bug is corrected. In the above figure, the two outcomes from debugging represents the above two cases. On successfully identifying the causes, corrections are made and regression testing is carried out, ensuring that everything works well after modifications.

The process of debugging continues till all errors are fixed. There is no specific format for debugging, rather it is an art and largely depends upon the psychology of the professional performing debugging. The professional may not be willing to accept the errors as debugging may increase difficulties, ask for frustrating problem solving and brain teasers, etc.

3.3.1 Introduction, ISO 9126 Quality Standards, Quality Management Models

Q52. Discuss the three steps required to achieve quality in software product and also mention the quality characteristics of ISO 9126 quality standards.

Answer : Model Paper-II, Q11(b)

The quality in software product can be achieved in three steps,

(a) Quality planning

(b) Management

(c) Measurement.

(a) Quality Planning

Quality planning refers to the set of quality related activities that a project is supposed to do in order to achieve the appropriate quality goal. Setting a quality goal is much difficult task than scheduling and effort estimation. Quality goals are usually specified in terms of acceptance criteria, which suggests that the delivered software must work according to acceptance criteria. Moreover, the quality can also be specified in terms of number of defects found during acceptance testing.

Based on the size and type of the product, there exist different types of quality attributes.

Analyzing the Quality Attributes

Quality attributes are considered for performing architectural design assessment. Some of these attributes include reliability, performance, security, maintainability, flexibility, testability, portability, reuseability and interoperability. Each of these attributes are analyzed in an isolated manner.

However, the product demands complex algorithm so as to attain efficiency, therefore due to this, incorporation of all these attributes with equal weightage becomes difficult in the software product. Aside this, determine the quality attributes such as maintainability or robustness by emphasizing on process definition. Subsequently, the maintainability can be determined by procedure parameters such as lines of code, error messages, length of user manuals and product metrics.

ISO 9126 Quality Standards

The attributes such as functionality, usability, reliability, performance and supportability are often referred to as software quality attributes. These were initially introduced by Hewlett-Packard which remains as target values to be satisfied

1. **Functionality**

In order to determine functionality, usually the efficiency of the program and also its maturity in performing various functions is assessed and finally its security aspects are analyzed deeply.

It has five sub characteristics given below,

(i) **Suitability**

It indicates that the operations of the software are correct and relate the specifications.

(ii) **Accurateness**

It signifies that the functions are accurate and correct.

Example

Check the cost of the ticket while booking.

(iii) **Interoperability**

It signifies that no software component or system can operate alone. They are interdependent and has to interact with other components.

(iv) **Compliance**

It signifies the compatibility of software with government set laws and guidelines such as SOX.

(v) **Security**

It checks the unauthorized user of software functions.

2. **Reliability**

Its given program depends on following factors.

- ❖ The mean time to failure.
- ❖ The potentiality to escape errors and the ability to recover from failures.
- ❖ Frequency and severity of failure of a program.
- ❖ The predictability of the program etc.

It has three subattributes.

(i) **Maturity**

It is concerned with the frequency of software failure cause because of the development of a bug with in the software.

(ii) **Fault Tolerance**

It is concerned with the method of delivering the performance of the software even in the event of failures.

(iii) **Recoverability**

It consists of three subattributes.

(i) **Understandability**

It is concerned with the level of understanding or which ease the functionality and man machine interface is understood by the user.

(ii) **Learnability**

It is concerned with the type of training essential for different types of users novice, expert, casual inorder to get informed about the software.

(iii) **Operability**

It is concerned with how much ease the software is put into practice.

4. **Efficiency**

The ability of a software to use system resources in an optimal way. Time behaviour and resource behaviour are the sub-attributes of efficiency.

it contains two attributes.

(i) **Time Behavior**

It is concerned with the total amount of transaction occurred with in a minute. It is most important attribute to measure efficiency.

(ii) **Resource Behavior**

It is concerned with the total no. of system software licenses and total count of computer resources such as memory, disk, CPU, network.

5. **Maintainability**

Supportability or maintainability of a given program can be computed based on following attributes.

- ❖ Configurability
- ❖ Compatibility
- ❖ Testability
- ❖ Extensibility
- ❖ Serviceability
- ❖ Adaptability
- ❖ The simplicity in separating the defects
- ❖ The simplicity of installation of a program etc.

It is further sub-divided into four attributes.

(i) **Analyzability**

It is concerned with how much ease the reason for faults is detected.

(ii) **Changeability**

It is concerned with the hardwork required for altering

Capability Maturity Models of Process Improvement

The software development organizations emphasizes upon product improvement by making advancements in the process steps. Because of the involvement of multiple activities in each step, the entire process becomes complicated. Every process contain certain characteristics and incorporation of one attribute may prevent the inclusion of other. For instance, if the software development has rapidity then visibility factor would be ignored.

Some of the process characteristics include,

1. **Visibility**

   It is concerned with the productive output generated through process activities and it should be externally visible to the users.

2. **Supportability**

   It is concerned with tools substantiated by the process activities.

3. **Acceptability**

   It is concerned with acceptability of process. In essence, whether or not the process is used by the software engineers carrying out the product development.

4. **Reliability**

   It is concerned with the process. In simple terms, it checks whether the process can avoid errors or detect them prior to the development of the product.

5. **Robustness**

   It is concerned with the functioning of a process even in extreme conditions like unexpected failure or any occurrence of problems.

6. **Maintainability**

   It is concerned with the adherence of all upcoming requirements in the organization.

7. **Rapidity**

   It is concerned with the fast working of system for particular specifications.

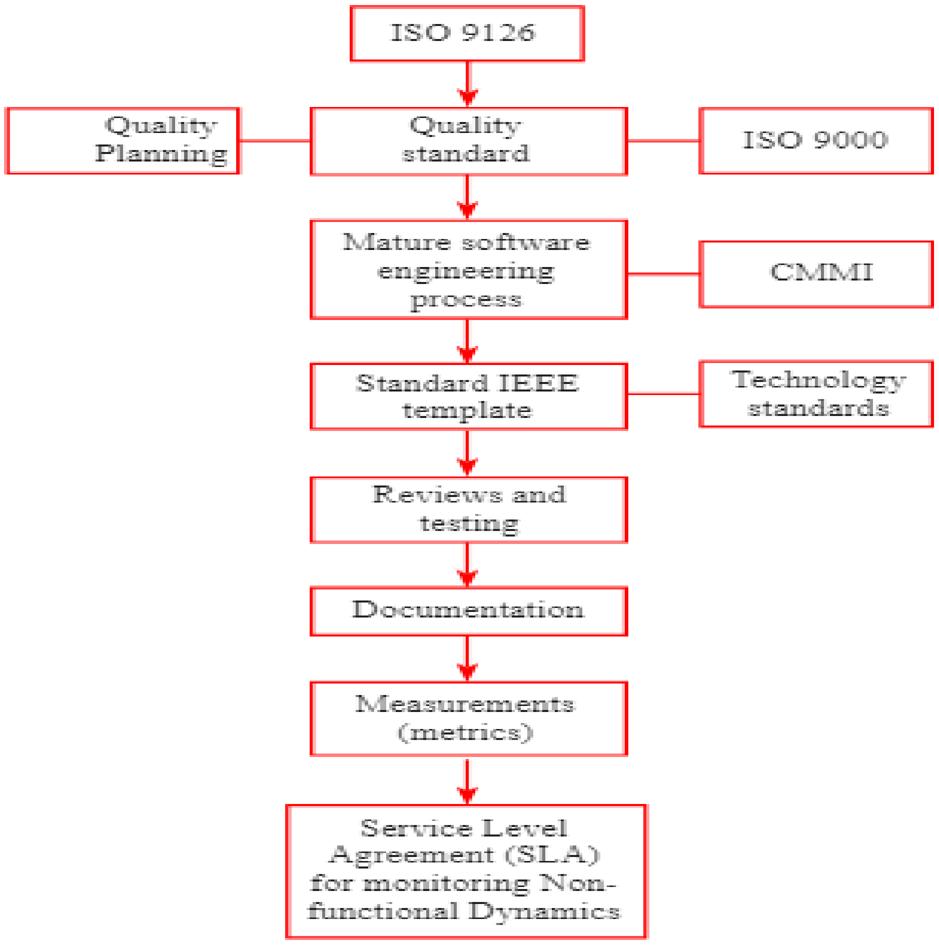


Figure: Software Quality Assurance

3.3.4 Class Coupling Metric – Coupling Between Objects, Monitoring Dynamic Quality Attributes (Visible Externally) of a Software

Q62. Write short notes on,

(i) Class Coupling

(ii) Class cohension metrics.

Answer : Model Paper-III, Q11(b)

(i) **Class Coupling**

If one class makes use of another class as an instance then that class is called class coupling. It is rarely used because, its implementation reduces reusability factor.

By definition, the coupling between the objects is nothing but the total number of inherited classes acquired by the class.

Example

Class A inheriting the attributes from two classes class *B* and class C. Then Class *A* takes single object from class *B* and another object from class C. Thus, generating a CBO pertaining to Class *A* is 2.

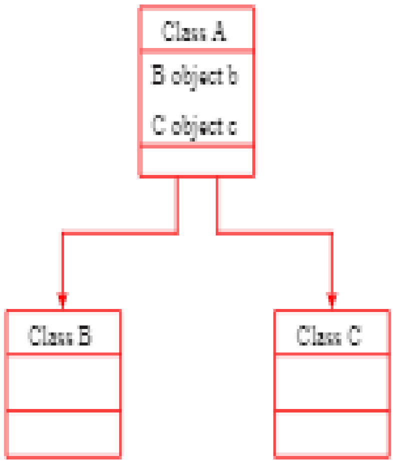


Figure: Coupling Between Objects

(ii) **Class Cohesion Metric**

The class cohesion metric property is responsible for incorporating data hiding feature in object oriented system. Here class A has high cohesiveness while a class having less cohesiveness stays intricate and full of defects. This state is referred to as lack of cohesion in methods (LCOM).

The existence of different or non-identical methods in a class is represented by LCOM metric. This is found after comparing similar attributes in two methods.

For instance, class having three methods-method 1, method 2 and method 3.

Method 1 = (K_1, K_2, K_3)

Method 2 = (K_1, K_3)

Method 3 = (K_3)

Method 1 $\prod$ method 2 = 1

Method 1 $\prod$ method 3 = 1

Method 2 $\prod$ method 3 = 0

Here, total number of methods with identical attributes is nothing but total methods with intersection as 1 is 2.

Total number of methods with non-identical attributes is nothing but the methods with intersection as 0 is 1.

Then LCOM = 1 – 2 = – 1

When LCOM < 0 then cohesion is 0, therefore LCOM = 0.

When LCOM > 0 and high then class cohesion becomes very high.