UNIT- III

Explain Coupling and Cohesion

- Coupling: Coupling is the measure of the degree of interdependence between the modules. A good software will have low coupling.

Types of Coupling:

- Data Coupling: If the dependency between the modules is based on the fact that they communicate by passing only data, then the modules are said to be data coupled. In data coupling, the components are independent of each other and communic-ate through data. Module communications don't contain tramp data. Example - customer billing system.
- Stamp Coupling: In stamp coupling, the complete data structure is passed from one module to another module. Therefore, it involves tramp data. It may be necessary due to efficiency factors - this choice was made by the insight-ful designer, not a lazy programmer.
- Control Coupling: If the modules communi-cate by passing control information, then they are said to be control coupled. It can be bad if parameters indicate compl-etely different behaviour and good if param-eters allow factoring and reuse of functio

nality. Example - sort function that takes comparison function as an argument.

- External Coupling: In external coupling, the modules depend on other modules, external to the software being developed or to a particular type of hardware. Ex- protocol, external file, device format, etc.
- Common Coupling: The modules have shared data such as global data structures. The changes in global data mean tracing back to all modules which access that data to evaluate the effect of the change. So it has got disadvantages like difficulty in reusing modules, reducing ability to control data accesses, and reduced maintainability.
- Content Coupling: In a content coupling, one module can modify the data of another module, or control flow is passed from one module to the other module. This is the worst form of coupling and should be avoided.
- Cohesion: Cohesion is a measure of the degree to which the elements of the module are functionally related. It is the degree to which all elements directed towards performing a single task are contained in the component. Basically,

cohesion is the internal glue that keeps the module together. A good software design will have high cohesion.

Types of Cohesion:

- Functional Cohesion: Every essential element for a single computation is contained in the component. A functional cohesion performs the task and functions. It is an ideal situation.
- Sequential Cohesion: An element outputs some data that becomes the input for other elements, i.e., data flow between the parts. It occurs naturally in functional programming languages
- Communicational Cohesion: Two elements operate on the same input data or contribute towards the same output data. Example- update record in the database and send it to the printer.
- Procedural Cohesion: Elements of procedural cohesion ensure the order of execution. Actions are still weakly connected and unlikely to be reusable. Ex- calculate student GPA, print student record, calculate cumulative GPA, print cumulative GPA
- Temporal Cohesion: The elements are related by their timing involved. A module connected with temporal cohesion all the

tasks must be executed in the same time span. This cohesion contains the code for initializing all the parts of the system. Lots of different activities occur, all at unit time.

- Logical Cohesion: The elements are logically related and not functionally. Ex - A component reads inputs from tape, disk, and network. All the code for these functions is the same component. Operations are related, but the functions are significantly different.
- Coincidental Cohesion: The elements are not related (unrelated). The elements have no conceptual relationship other than location in source code. It is accidental and the worst form of cohesion. Ex - print next line and reverse the characters of a string in a single component.

Why software Testing is necessary in software development?

Software Testing is a method to check whether the actual software product matches expected requirements and to ensure that software product is Defect free. It involves execution of software/ system components using

manual or automated tools to evaluate one or more properties of interest. The purpose of software testing is to identify errors, gaps or missing requirements in contrast to actual requirements. Software Testing means the Verification of Application Under Test (AUT). This Software Testing course introduces testing software to the audience and justifies the importance of software testing. Software Testing is important because if there are any bugs or errors in the software, it can be identified early and can be solved before delivery of the software product. Properly tested software product ensures reliability, security and high performance which further results in time saving, cost effectiveness and customer satisfaction. Testing is important because software bugs could be expensive or even dangerous. Software bugs can potentially cause monetary and human loss, and history is full of such examples.

- In April 2015, Bloomberg terminal in London crashed due to software glitch affected more than 300,000 traders on financial markets.

Here are the benefits of using software testing:

- Cost-Effective: It is one of the important advantages of software testing. Testing any IT project on time helps you to save your money for the long term. In case if the bugs caught in the earlier stage of software testing, it costs less to fix.
- Security: It is the most vulnerable and sensitive benefit of software testing. People are looking for trusted products. It helps in removing risks and problems earlier.
- Product quality: It is an essential requirement of any software product. Testing ensures a quality product is delivered to customers.
- Customer Satisfaction: The main aim of any product is to give satisfaction to their customers. UI/UX Testing ensures the best user experience.

Write Short notes on:-

a, Static Testing: Static Testing is a type of a Software Testing method which is performed to check the defects in software without actually

executing the code of the software applic-ation. Whereas in Dynamic testing checks the code is executed to detect the defects. Static testing is performed in early stage of development to avoid errors as it is easier to find sources as failures and it can be fixed easily. The errors that can't be found using Dynamic Testing, can be easily found by Static Testing

b, Dynamic Testing:- Dynamic Testing is a type of Software Testing which is performed to analyze the dynamic behav-ior of the code. It includes the testing of the software for the input values and output values that are analyzed. Dynamic Testing is basically performed to describe the dynamic behavior of code. It refers to the observation of the physical respo-nse from the system to variables that are not constant and change with time. To perform dynamic testing the software should be compiled and run. It includ-es working with the software by giving input values and checking if the output is an expected by executing particular test cases which can be done with either manually or with automation

process.
In 2V's i.e, Verification and Validation, Validation is Dynamic Testing.
Levels of Dynamic Testing:
There are various levels of Dynamic Testing. They are:

- Unit Testing
- Integration Testing
- System Testing
- Acceptance Testing.

c, Black Box Testing: Black Box Testing method in which the functionalities of software applications are tested without having knowledge of internal code structure, implementation details and internal paths. Black box Testing mainly focuses on input and output of software applications and it is entirely based on software requirements and specifications. It is also known as Behavioral Testing. Operating system like Windows, a website like Google, a database like Oracle or even your own custom application black box testing can test all these applications by just focusing on the inputs and outputs without knowing their internal code implementation

d. White Box Testing: White Box Testing technique in which internal structure, design and coding of software are tested to verify flow of input-output and to improve design, usability and security. In white box testing, "open box" code is visible to testers so it is also called clear box testing, Open box testing, Transparent box testing, Code-based testing and Glass box testing. White box testing is software engineering is based on the inner workings of an application and revolves around internal testing. The term 'White Box' was used because of the see through box concept. The clear box or White Box name symbolizes the ability to see through the software's outer shell (or 'box') into its inner workings.

e, Structural Testing: Structural Testing also known as glass box testing or white box testing is an approach where the tests are derived from the knowledge of the Software's structure or internal implementation. The other names of structural testing includes clear box testing, open box testing, logic driven testing or path driven testing.

Structural Testing Techniques:

- Statement Coverage: This technique is aimed at exercising all programming statements with minimal tests.
- Branch Coverage: This technique is running a series of tests to ensure that all branches are tested at least once.
- Path Coverage: This technique corresponds to testing all possible paths which means that each statement and branch are covered.

Explain Software Quality Assurance.

Software Quality Assurance (SQA) is simply a way to assure quality in the software. It is the set of activities which ensure processes, procedures as well as standards are suitable for the project and implemented correctly. Software Quality Assurance is a process which works parallel to development of software. It focuses on improving the process of development of software so that problems can be prevented before they become a major issue. Software Quality Assurance is a kind of Umbrella activity that is applied throughout the software process.

Software Quality Assurance has:

1. A quality management approach.
2. Formal technical reviews.
3. Multi testing strategy
4. Effective software engineering technology
5. Measurement and reporting mechanism.

Major Software Quality Assurance Activities:

1. SQA Management Plan: Make a plan for how you will carry out the sqa through out the project. Think about which set of software engineering activities are the best for project check level of sqa team skills.
2. Set the Check Points:
SQA team should set checkpoints. Evaluate the performance of the project on the basis of collected data on different check points.
3. Multi testing Strategy:
Do not depend on a single testing approach When you have a lot of testing approaches available use them.
4. Measure Change Impact: The changes for making the correction of an error sometimes re introduces more errors keep the measure of impact of change on project. Reset the new change to change the check the compatibility of this fix

with whole project.

5. Manage Good Relations: In the working environment managing good relations with other teams involved in the project development is mandatory. Bad relation of sqa team with programmess team will impact directly and badly on proje-ct. Don't play politics.

- Benefits of Software Quality Assurance (SQA):

1. SQA produces high quality software.
2. High quality application saves time and cost.
3. SQA is beneficial for better reliability.
4. SQA is beneficial in the condition of no maintenance for a long time.
5. High quality commercial software increase market share of company.
6. Improving the process of creating softw-are.
7. Improves the quality of the software.

What are the McCall's Quality factors? Explain them briefly.

McCall software quality model was intro-duced in 1977. This model is incorporated with many attributes termed as softw-are factors, which influence a software.

The model distinguishes between two levels of quality attributes:

1. Quality Factors:- The higher level quality attributes which can be assessed directly are called quality factors. These attributes are external attributes. The attributes in this level are given more importance by the users and managers.
2. Quality Criteria:- The lower or second level quality attributes which can be assessed either subjectively or objectively are called Quality Criteria. These attributes are internal attributes. Each quality factor has many second level of quality attrib-utes or quality criteria.

Example-
Usability quality factor is divided into operability, training, communicativeness, input/output volume, input/output rate.
This model classifies all software quality factors. The 11 factors are organised into three product quality revision, and product transition factors.
The following are the product quality factors-

1. Product Operation: It includes five softw-are quality factors, which are related with the requirements that directly

affect the operation of the software such as operational performance, convenience, ease of usage and its correctness. These factors help in providing a better user experience.

- Correctness - The extent to which a software meets its requirements specification.
- Efficiency - The amount of hardware resources and code the software, needs to perform a function.
- Integrity - The extent to which the software can control an unauthorized person from the accessing the data or software.
- Reliability - The extent to which a software performs its intended functions without failure.
- Usability - The extent of effort required to learn, operate and understand the functions of the software.

2. Product Revision: It includes three software quality factors, which are required for testing and maintenance of the software. They provide ease of maintenance, flexibility and testing effort to support the software to be functional according to the needs and requirements of the user in the

future.

- Maintainability - The effort required to detect and correct an error during maintenance phase.
- Flexibility - The effort needed to improve an operational software program.
- Testability - The effort required to verify a software to ensure that it meets the specified requirements.

3. Product Transition: It includes three software quality factors, that allows the software to adapt to the change of environments in the new platform or technology from the previous
   - Portability - The effort required to transfer a program from one platform to another.
   - Re-usability - The extent to which the program's code can be reused in other applications.
   - Interoperability - The effort required to integrate two systems with one another.