Q2: What do you mean by the terms cohesion & coupling in the context of software design?

Ans:-

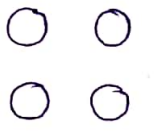
Coupling & Cohesion :-

→ Effective problem decomposition is an important characteristic of a good design. Good module decomposition is indicated through high cohesion of the individual modules & low coupling of the modules with each other.

Module coupling :-

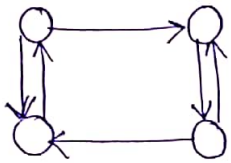
In software engineering, the coupling is the degree of interdependence between software modules. Two modules that are tightly coupled are strongly dependent on each other. However, two modules that are loosely coupled aren't dependent on each other. Uncoupled modules have no interdependence at all within them.

→ The various types of coupling techniques are shown in fig :-



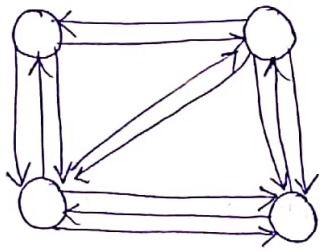
Uncoupled: no dependencies

(a)



Loosely Coupled: Some dependencies

(b)



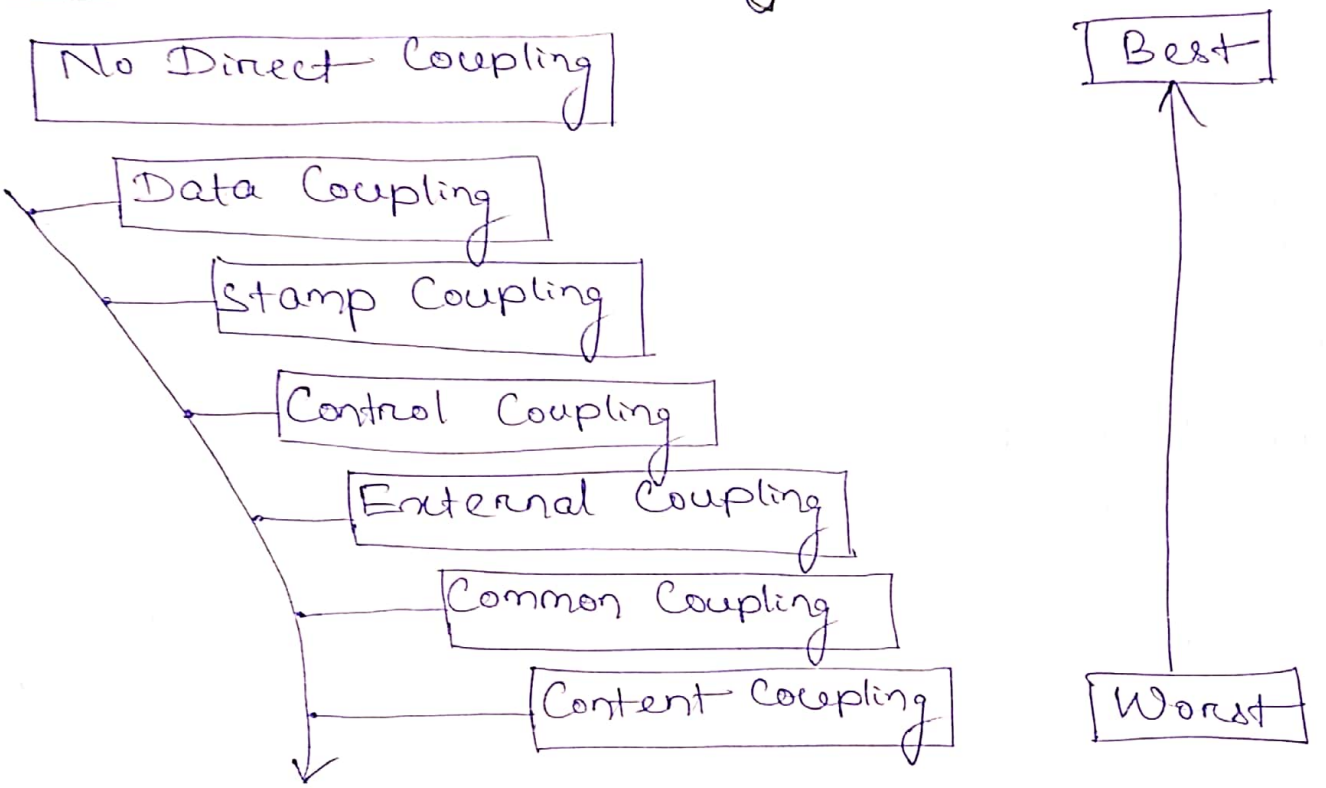
Highly Coupled: many dependencies

(c)

→ A good design is the one that has low coupling. Coupling is measured by the no. of relations between the modules. That is, the coupling increases as the no. of calls between modules increase or the

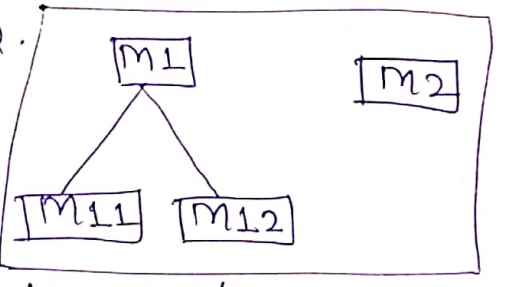
amount of shared data is large. Thus, it can be said that a design with high coupling will have more errors.

Types of module coupling :—

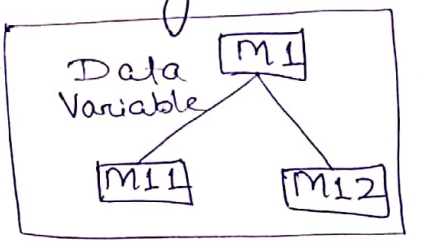


1. **No Direct Coupling :—** There is no direct coupling between M1 & M2.

→ In this case, modules are subordinates to different modules. Therefore, no direct coupling.



2. **Data Coupling :—** When data of one module is passed to another module, this is called data coupling.



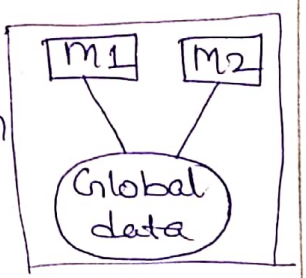
3. **Stamp Coupling :—** Two modules are stamp coupled if they communicate using composite data items such as structure, objects, etc. When the module passes non-global data structure or entire structure

to another module, they are said to be stamp coupled. For example, passing structure variable in C or object in C++ language to a module.

4. Control Coupling :— Control coupling exists among two modules if data from one module is used to direct the structure of instruction execution in another.

5. External Coupling :— External coupling arises when two modules share an externally imposed data format, communication protocols, or device interface. This is related to communication to external tools & devices.

6. Common Coupling :— Two modules are common coupled if they share information through some global data items.

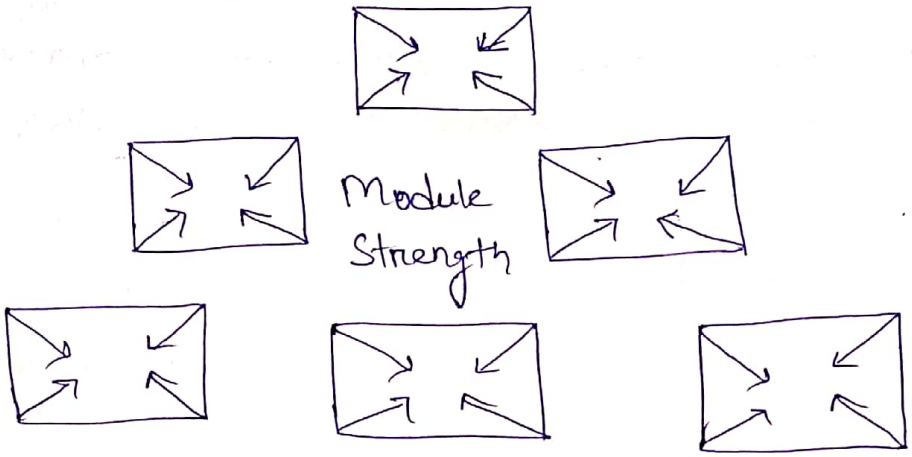


7. Content Coupling :— Content coupling exists among two modules if they share code, e.g., a branch from one module into another module.

Module Cohesion :—

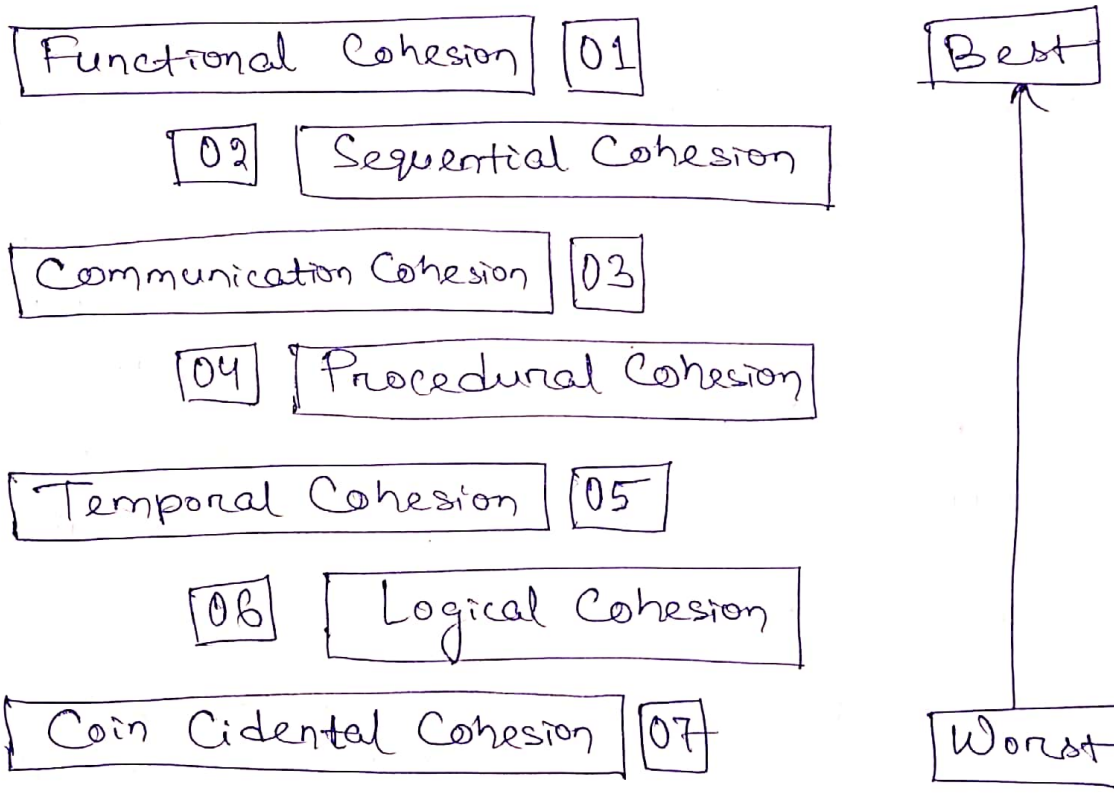
In computer programming, cohesion defines to the degree to which the elements of a module belong together. Thus, cohesion measures the strength of relationships between pieces of functionality within a given module. For example, in highly cohesive systems, functionality is strongly related.

→ Cohesion is ~~or~~ an ordinal type of measurement & is generally described as "high cohesion" or "low coupling".



(Cohesion = Strength of relations within Modules)

Types of Modules Cohesion :-



1. Functional Cohesion :-

Functional cohesion is said to exist if the different elements of a module, cooperate to achieve a single function.

2. Sequential Cohesion :— A module is said to possess sequential cohesion if the element of a module form the components of the sequence, where the output from one component of the sequence is input to the next.

3. Communicational Cohesion :— A module is said to have communicational cohesion, if all tasks of the module refer to or update the same data structure, e.g., the set of functions defined on an array or a stack.

4. Procedural Cohesion :— A module is said to be procedural cohesion if the set of purpose of the module are all parts of a procedure in which particular sequence of steps has to be carried out for achieving a goal, e.g., the algorithm for decoding a message.

5. Temporal Cohesion :— When a module includes functions that are associated by the fact that all the methods must be executed in the same time, the module is said to exhibit temporal cohesion.

6. Logical Cohesion :— A module is said to be logically cohesive if all the elements of the module perform a similar operation. For example, error handling, data input & data output, etc..

7. Coincidental Cohesion :— A module is said to have coincidental cohesion if it performs a set of tasks that are associated with each other very loosely, if at all.

Q3:- Discuss the SEI CMM-based quality assessment?

Ans:- SEI Capability Maturity Model (SEI CMM) helped organizations to improve the quality of the software they develop & therefore adoption of SEI CMM model has significant business benifits.

→ SEI CMM can be used two ways : capability evaluation & software process assessment. Capability evaluation & Software process assessment differ in motivation, objective & the final use of the result.

→ Capability evaluation provides a way to assess the software process capability of an organization.

→ The result of capability evaluation indicates the likely contractor preformance, if the contractor is awarded a work. Therefore, the results of software process capability assessment can be used to select a contractor.

→ On the otherhand, s/w process assessment is used by an organization with the objective to improve its process capability. Thus, this type of assessment is for purely internal use.

→ SEI CMM classifies software development industries into the following five maturity levels. The different levels of SEI CMM have been designed so that it is easy for an organization to slowly build its quality system starting from scratch.

Level 1: Initial -

→ A s/w development organization at this level is characterized by ad hoc activities. Very few

or no processes are defined & followed. Since, s/w production processes are not defined, different engineers follow their own process & as a result development efforts become chaotic. Therefore, it is also called chaotic level.

→ The success of projects depends on individual efforts & heroics. When engineers leave, the successors have great difficulty in understanding the process followed & the work completed. Since, formal project management practices are not followed, under time pressure short cuts are tried out leading to low quality.

Level 2: Repeatable :—

At this level, the basic project management practices such as tracking cost & schedule are established. Size & cost estimation techniques like function point analysis, COCOMO, etc. are used.

→ The necessary process discipline is in place to repeat earlier success on projects with similar applications. Please remember that opportunity to repeat a process exists only when a company produces a family of products.

Level-3 : Defined :—

→ At this level, the ~~basic~~ process both project management practices ~~such as tracking~~ & development activities are defined & documented. There is a common organization-wide understanding of activities, roles & responsibilities. The processes though defined, the process & product qualities are not measured. ISO 9000 aims at achieving this level.

Level 4: Managed:-

→ At this level, the focus is on software metrics. Two types of metrics are collected. Product metrics measure the characteristics of the product being developed, such as its size, reliability, time complexity, understandability, etc.

→ Process metrics reflect the effectiveness of the process being used, such as average defect correction time, productivity, average number of defects found per hour inspection, average number of failures detected during testing per LOC, etc. Quantitative quality goals are set for the products.

→ The s/w process & product quality are measured & quantitative quality requirements for the product are met. Various tools like Pareto charts, fishbone diagrams, etc. are used to measure the product & process quality.

→ The process metrics are used to check if a project performed satisfactorily. Thus, the results of process measurements are used to evaluate project performance rather than improve the process.

Level 5: Optimizing:-

→ At this stage, process & product metrics are collected. Process & product measurement data are analyzed for continuous process improvement. For example, if from an analysis of the process measurement results, it was found that the code

reviews were not very effective & a large no. of process measurement errors were detected only during the unit testing, then the process may be fine-tuned to make the review more effective. Also, the lessons learned from specific projects are incorporated into the process.

→ Such an organization identifies the best software engineering practices & innovations which may be tools, methods, or processes. These best practices are transferred throughout the organization.

Key process areas(KPA) of a software organization:—

CMM Level	Focus	Key Process Areas
1. Initial	Competent people	
2. Repeatable	Project Management	Slw project planning Slw configuration management
3. Defined	Definition of Processes	Process definition Training Program Peer Reviews
4. Managed	Product & Process Quality	Quantitative Process metrics Slw quality management
5. Optimizing	Continuous Process improvement	Defect Prevention Process change management Technology change management