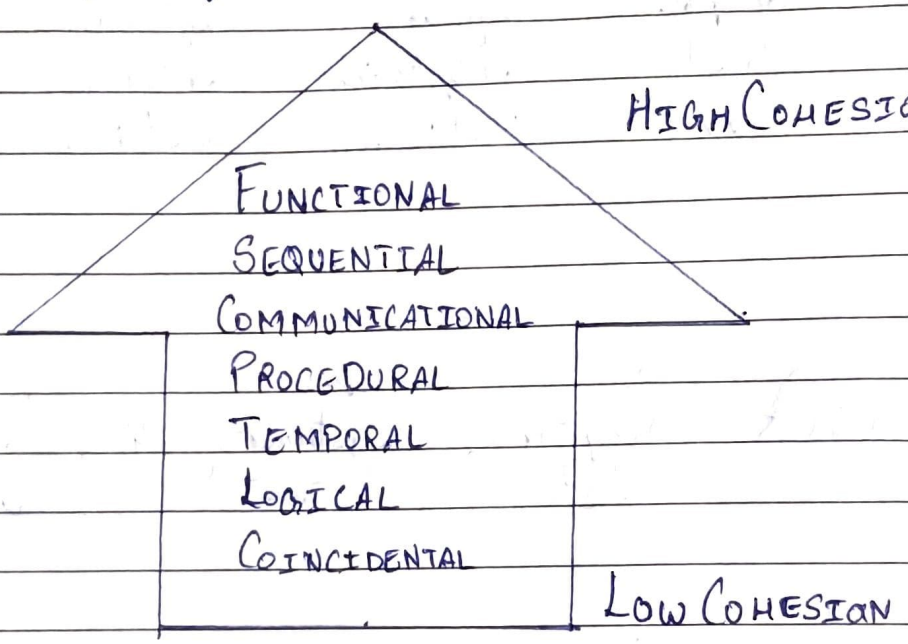
Varun Khadayate A016 08-09-2020

S.E.

Assignment - IV

Q1. Modular Cohesion

In computer programming, cohesion defines to the degree to which the elements of a module belong together. Thus, cohesion measures the strength of relationship between pieces of functionality within a given module. Eg: In highly cohesive system, functionality is strongly related.

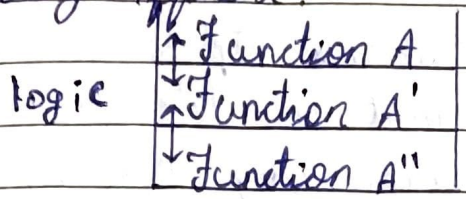


1. Coincidental Cohesion : The elements are not related. The elements have no conceptual relationship other than location in source code. It is accidental and the worst form of cohesion. Eg: Print next line and reverse the characters of a string in a single component.

Functions A	
Function B	Function C
Function D	Function E

Camlin Page
Date / /

2. Logical Cohesion: The elements are logically related and not functionally. Several logically related elements are in the same component and one of the element is selected by the client component. Eg: A component reads inputs from tape, disk and network. All the code for these functions is the same component present is. Operations are related, but the functions are significantly different.



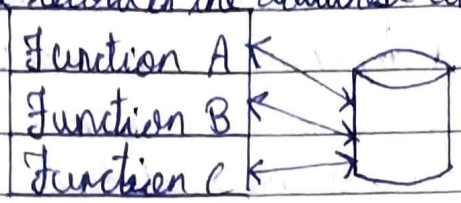
3. Temporal Cohesion: The elements are related by their timing involved. A module connected with temporal cohesion all the tasks must be executed in the same time-span. This cohesion contains the code for initializing all the parts of the system. Lots of different activities occur, all at in time.

Time t_0
Time t_0+X
Time t_0+2X

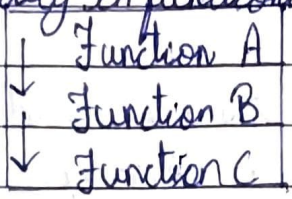
4. Procedural Cohesion: Elements of procedural cohesion ensure the order of execution. Actions are still weakly connected and unlikely to be reusable. Eg: Calculate student GPA, print student record, Calculate cumilative GPA, print cumilative GPA.

Function A
Function B
Function C

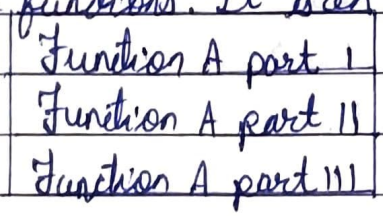
5. Communicational Cohesion: Two elements operate on the same input data or contribute towards the same output data.
Eg: Update record in the database and send it to the printer.



6. Sequential Cohesion: An elements outputs some data that becomes the input for other element, i.e, data flow between the parts. It occurs naturally in functional programming languages.



7. 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.

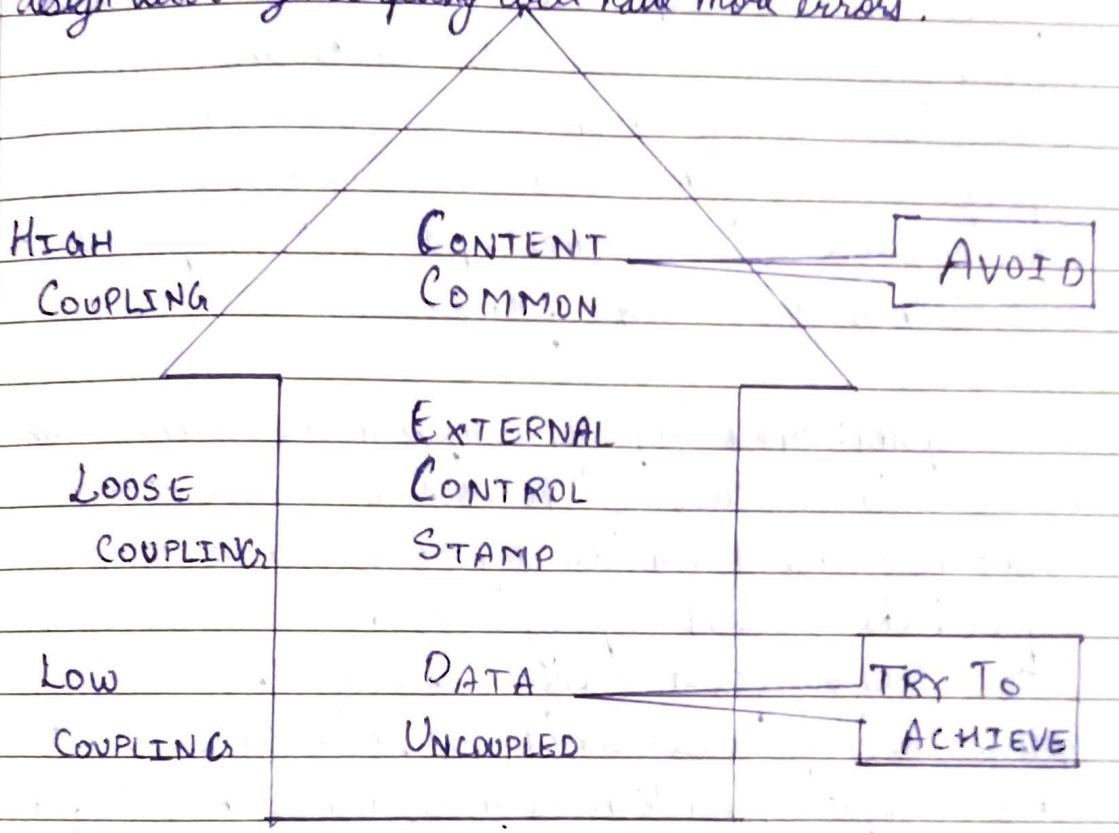


Object Oriented Software Development Cohesion

In object-oriented programming, if the methods that serve a class tend to be similar in many aspects, then the class is said to have high cohesion. In a highly cohesive system, code readability and reusability is increased, while complexity is kept manageable.

Q2. Modular Coupling

In software engineering, the coupling is the degree of interdependent interdependence between software modules. Two modules that are tightly coupled are strongly dependent on eachother. Uncoupled modules have no interdependence at all within them. A good design is the one that has low coupling. Coupling is measured by the number of relations between the modules. That is, coupling increases as the number of calls between modules increases or amount of shared data is large. Thus, it can be said that a design with high coupling will have more errors.



1. Uncoupled Coupling: Completely uncoupled components are not systems. Systems are made of interacting components.

2. Data Coupling: Data Coupling is simply coupling of data, i.e. interaction between data when they are passed through parameters using or when modules share data through parameters. When data of one module is shared with other modules or passed through other

modules, this condition is said to be data coupling. Eg: The print routine takes the customer name, address, and billing information as arguments.

3. Stamp Coupling: Stamp Coupling simply means the sharing of composite data structures between modules. If the modules interact or communicate by sharing or passing the structure that contains more information than required to perform their action, then these modules are said to be stamp coupled. Therefore, it involves stamp data. It may be necessary due to efficiency factors - this choice made by the insightful designer, not a lazy programmer. Eg: Customer Billing System, the print routine of the customer billing accepts customer data structure as an argument, parses it, and prints the name, address and billing information.

4. Control Coupling: Control Coupling simply means to control data sharing between modules. If the modules interact or connect by sharing controlled data, then they are said to be controled coupled. The controlled coupling means that one module controls the flow of data or information by other modules by them the information about what to do. Eg: Sort that takes a comparison function as an argument. The sort function is clearly defined: return a list in sorted order, where sorted is is determined by a parameter.

5. External Coupling: The external coupling means the sharing of data structures or format that are imposed externally between the modules. External coupling is very important but there should be a limit also. It should be limited to a smaller number of modules with structures. Eg: External file, Device interface, Protocol transfer of data and information.

6. Common Coupling: Common Coupling simply means the sharing of common data or global data between several modules. If 2 modules share the information through global data items or interact by sharing common data, then they are said to be commonly coupled. Eg: Process control component maintains current data about state of operation. Gets data from multiple sources. Supplies data to multiple sinks. Each source process writes directly to global data store. Each sink process reads directly from global data store.

7. Content Coupling: Content coupling simply means using of data or control information maintained in other module by one module. This coupling is also known as pathological coupling. In these coupling, one module relies or depends upon the internal working of another module. Therefore, if any changes have to be done in the inner working of a module then this will lead to the need for change in the dependent module. Eg: Part of program handles lookup for customer. When customer not found, component adds customer by directly modifying the content of data structure containing customer data.

Camlin | Page
Date / /

Object Oriented Software Development Coupling
In OOP coupling it is of 2 types:
1. Sub Class Coupling: Describes the relationship between the child and its parent. The child is connected to parent, but the parent is not connected to the child.

2. Temporal Coupling: It is when 2 actions are bundled together into one module ~~to~~ just because they happen to occur at the same time.

By reducing coupling, on the face of it the programmer is going to merge unrelated units of code, which ~~cohesion~~ would also reduce Cohesion. Likewise, removing unrelated functions from a class will introduce ~~to~~ another class on which the original will need to depend, increasing coupling.

Q4. There are namely 6 non-functional requirement metrics which are:
1. Speed: Used to measure the processed transaction per second.
   User response time.
   Screen refresh time.

2. Size: Used to measure the size of packets delivered during the process.
   No. of ~~from~~ ROM chips.

3. Ease of Use: Training time and No. of help frames available

4. Reliability : Used to measure the mean time taken for failure to occur.
Probability of an unavailability.
Rate of failure occurance.
Availability.

5. Robustness : Used to measure the time required to restart after the failure is occured and resolved.
Percentage of events that caused failure.
Probability of the data that can be corrupted due to failure.

6. Portability : Used to measure the Percentage of data which has target dependent statements.
No. of target systems.

Q3. Coupling metrics measures the dependencies between a given entity and other entities of the program. The goal of these metrics is to ~~determine~~ estimate the stability of the whole program considered as a collection of entities.

The metrics used to measure code-coupling are:-
- DSQI - Robert Cecil "Software Package Metrics"

~~DSQI~~ DSQI

Camlin | Page
Date / /

DSQI [Design Structure Quality Index]
It is used for evaluating Object-Oriented software packages. However, it is meant to be used in case of an extreme programming framework.
Using this software code coupling metrics, the metric is useful as the metrics calculation is comparatively transparent. This also allows the developer to build the software that follows these constraints and gets better metrics on their code.