# Cohesion

When function of one module cooperates with another to perform single objective then module is said to be good cohesion. The primary characteristics of neat module decomposition are high cohesion and low coupling. Cohesion is a measure of functional strength of a module. A module having high cohesion and low coupling is said to be functionally independent of other modules. By the term functional independence, we mean that a cohesive module performs a single task or function. A functionally independent module has minimal interaction with other modules. High cohesion has features of reliability, robustness and reusability where as low cohesion has difficult to test, reuse even difficult to understand and maintenance.

Classification of cohesion

The different classes of cohesion that a module may possess are described as follows from lower cohesion to higher where coincidental cohesion worst type and functional is best type.

Coincidental Low
Logical
Temporal
Procedural
Communicational
Sequential
Functional High

Coincidental cohesion: A module is said to have coincidental cohesion, if it performs a set of tasks that relate to each other very loosely. In this case, the module contains a random collection of functions. It is likely that the functions have been put in the module out of pure coincidence without any thought or design or parts of the module grouped arbitrarily. Different functions in module carry out different activities. For example, in a transaction processing system (TPS), the get-input, print-error, and summarize-members functions are grouped into one module. The grouping does not have any relevance to the structure of the problem.

Logical cohesion: A module is said to be logically cohesive, if all elements of the module perform similar operations, e.g. error handling, data input, data output, etc. An example of logical cohesion is the case where a set of print functions generating different output reports are arranged into a single module, such as module contain set of print functions to generate various type of output report such as salary, grade sheet, manual report etc. Here parts of the modules are logically categorized to do same thing even if they are different by nature (e.g grouping all mouse and keyboards input handling routines).

Temporal cohesion: When a module contains functions that are related with each other and all the functions must be executed in the same time span, then module is said to exhibit temporal cohesion. For example when computer booted several functions need to be performed at the same time one by one such as initialization of memory, devices, loading OS etc. and The set of functions in a single module responsible for initialization, start-up, shutdown of some process, etc. exhibit temporal cohesion.

Another example we can consider as a functions which called after catching an exception which closes opened files, give notification to user, create error log etc.
Procedural cohesion: A module is said to possess procedural cohesion, if the set of functions of the module are all part of a procedure (algorithm)in which certain sequence of steps have to be carried out for achieving an objective, e.g. the algorithm for decoding a message. As in the above, the parts/functions of the modules always follow certain sequence of execution. Another example is function check file permission and then opened. Plus we can consider the activities included in the trading house can contain functions login (), place-order (), print-bill (),…log-out() etc which are operate on different data but flow sequentially.

Communicational cohesion: Communicational cohesion is also called informational cohesion. Parts/functions of the modules grouped together to operate on same data. A module is said to have communicational cohesion, if all functions of the module refer to or update the same data structure, e.g. the set of functions defined on an array or a stack.
Another example of informational cohesion is let us consider a module name student which contain various functions like admit student, enter marks, print grade sheet etc and all the data stored in an array name student record that is defined with in a module.
Sequential cohesion: A module is said to possess sequential cohesion, if the elements of a module form the parts of sequence, where the output of one function is input to the next. For example, in a Transaction Processing System, the get-input (), validate-input (), sort-input () functions are grouped into one module.
Another example in an online store system when customer request for some item then functions involved are follow as sequence of: create-order(), check-avail() and then order ().

Functional cohesion: Functional cohesion is said to exist, if different elements/parts/functions of a module cooperate to achieve a single task. For example, a module containing all the functions required to manage employees' pay-roll exhibits functional cohesion such as compute-overtime(), compute-workhr(), computededuction() etc . It is considered one of the most superior cohesion in the module but in certain cases not achievable and instead communicational cohesion overcome the solution.

# Coupling

Coupling between two modules/classes/components is a measure of the degree of interdependence or interaction between the two modules or mutual interdependence between modules. A module having high cohesion and low coupling is said to be functionally independent of other modules.

Two modules are said to be highly coupled when:
-function calls between two modules involve & shared large chunk of data.
-interaction occurs through shared data.

If two modules interchange large amounts of data, then they are highly interdependent. The degree of coupling between two modules depends on their interface complexity. The interface complexity is determined based on the parameter, no. of parameter etc. Module with low coupling is better.

Classification of Coupling

Classification of the different types of coupling will help to quantitatively estimate the degree of coupling between two modules. Five types of coupling can occur between any two modules. These are shown below according to flow of lower coupling to higher, indicated by an arrow.

Data Stamp Control Common Content
Low ->-> ->-> High

Data coupling: Two modules are data coupled, if they communicate by using an elementary data item e.g. an integer, a float, a character, etc such as passing any float value through function that computes square root. This data item should be problem related and not used for the control purpose.

Stamp coupling: Two modules are stamp coupled, if they communicate using a composite data item such as a record in PASCAL or a structure in C and used part of these data.

Control coupling: Control coupling exists between two modules, if data from one module is used in another module to direct the order of instructions execution. An example of control coupling is a flag set in one module and tested in another module.

Common coupling: Two modules are common coupled, if they share data through some global data items. It is also known as global coupling. Example, sharing global variable. Changing global variable leads changes in all modules.

Content coupling: Also known as pathological coupling. Content coupling exists between two modules, if they share code, e.g. a branch from one module into another module or if one module changes on internal working of another module (e.g accessing

local data of another module).

No coupling exists when modules do not communicate with each other.

From the above degree of coupling increases from the data coupling (that has less interdependency, coordination and data flow) to content coupling(that has high interdependency, coordination and information flow).

High coupling among modules makes design solution difficult to understand and maintenance, increase development effort, harder to reuse and test particular module, assembling of module may require more effort, change in one module usually leads a ripple effect and difficult to develop modules in independent way, these are the disadvantages. High cohesion is contrasted with high coupling and somehow it correlates with loose coupling and vice versa.