8- Functional Independence

The concept of functional independence is a direct extension of:

- Modularity
- Abstraction,
- And information hiding.

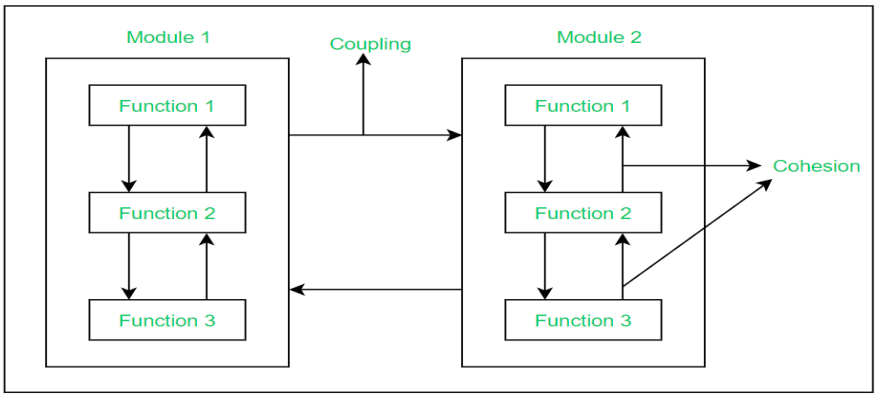
You should design software so that each module addresses a specific subset of requirements and has a simple interface when viewed from other parts of the program structure.

It is fair to ask why independence is important.?

Software with effective modularity, that is, independent modules, is easier to develop because functions can be divided and interfaces are simplified. Independent modules are easier to maintain and test because effects caused by design or code modification are limited, error propagation is reduced, and reusable modules are possible. To summarize, functional independence *is a key to good design,* and design is *the key to software quality*.

Independence is measured using two qualitative criteria:

cohesion and coupling.



A- Cohesion

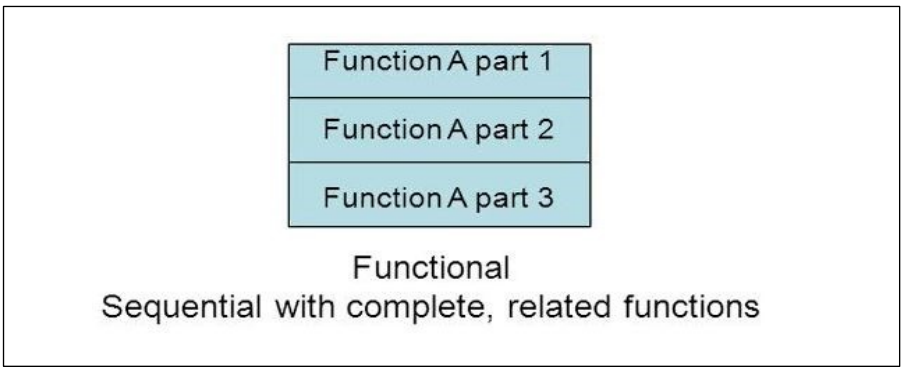
Is an indication of the relative functional strength of a module. Cohesion is a natural extension of the information-hiding concept. A cohesive module performs a single task within a software procedure, requiring little interaction with procedures being performed in other parts of a program. Stated simply, a cohesive module should do just one thing.

The systems having high cohesion will have elements such as instructions, groups of instructions, and the definition of data strongly connected to each other. This helps in improving the focus on a given task and thus, high cohesion is preferred.

There are various types of cohesion:

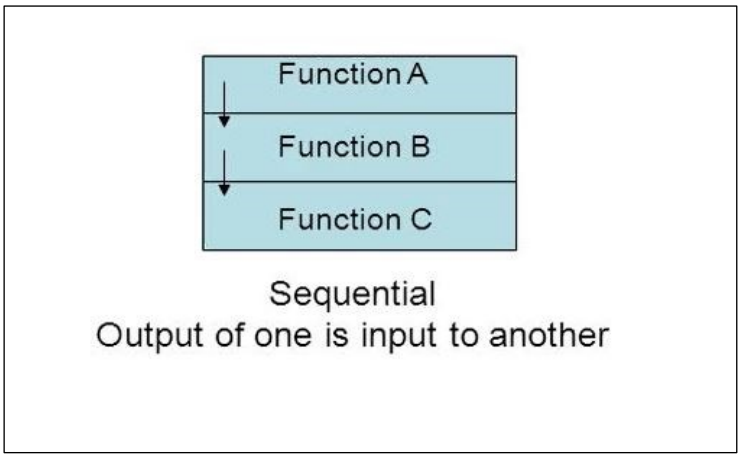
1. Functional cohesion

In this type of cohesion, the elements inside a module work together to achieve one common goal. The elements work in coordination with the main aim of staying focused on the task assigned. They perform only the activities which are necessary for the assigned work.



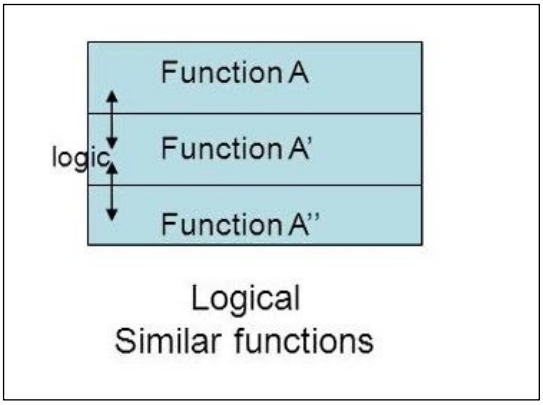
2. Sequential cohesion

In this type of cohesion, the output of a particular activity acts as an input to another. This helps in easy maintenance.



3. Logical Cohesion

Logical Cohesion is a type of cohesion in which all the elements within a module perform similar operations such as error handling.



B- Coupling

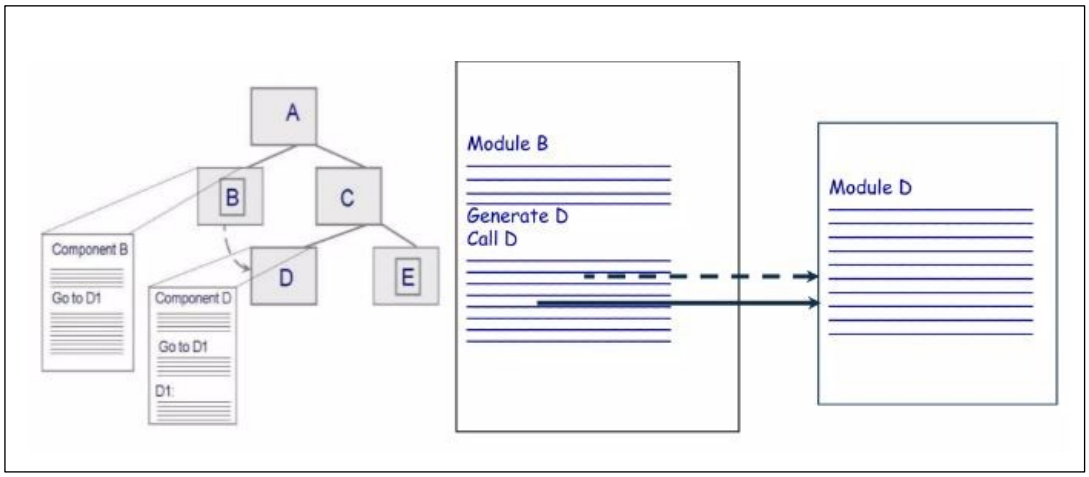
Coupling is defined as the degree to which the two modules are dependent on each other. It measures the strength of relationships between modules. Coupling depends on the interface complexity between modules, the point at which entry is made to a module, and what data pass across the interface. In software design, the low coupling is preferred.

There are various types of coupling:

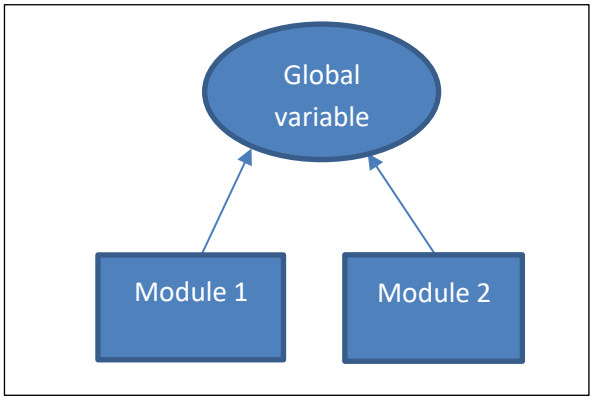
1. Content coupling

In this type of coupling, the interacting modules share code with each other. Basically, here one module depends on the implementation of the

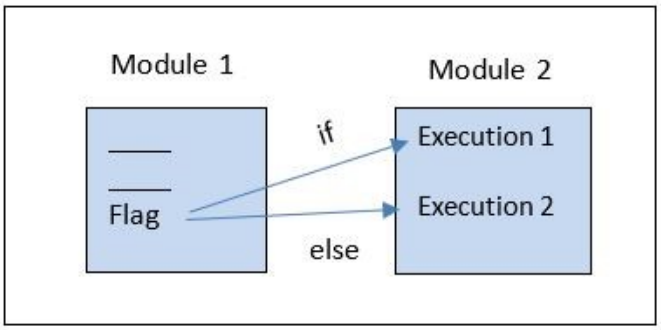
other module.



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



3. Control Coupling:
   Exists among two modules if data from one module is used to direct the instruction execution in another. It occurs using flag or command.



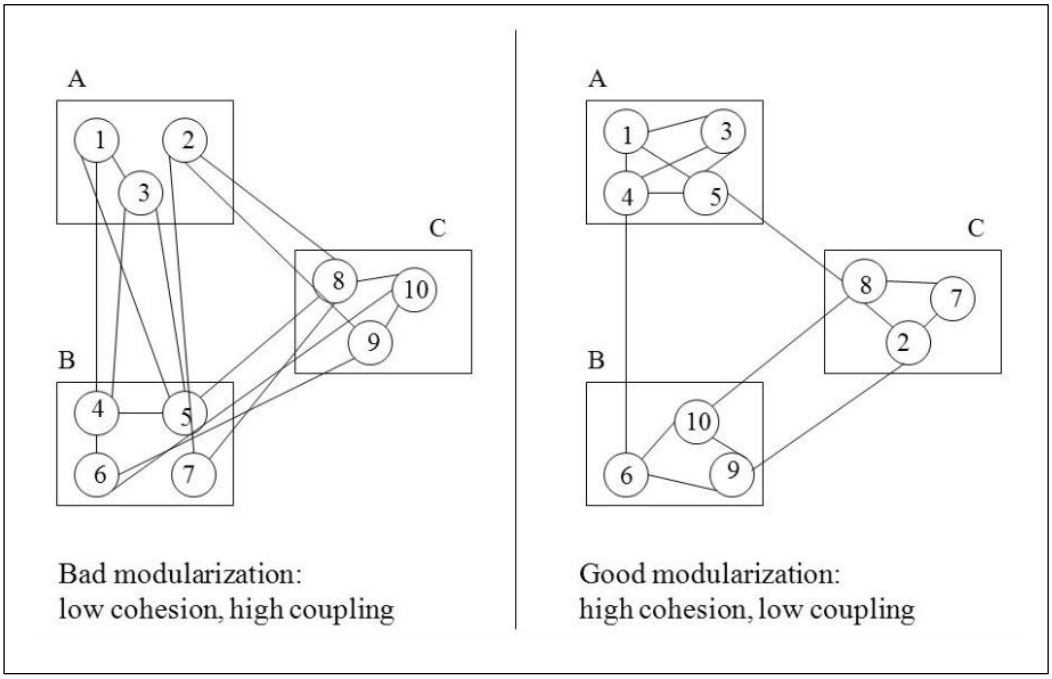
4. Stamp Coupling:

   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.

5. Data coupling

   In this type of coupling, one or more than one data item is passed between the interacting modules as parameter. The data shared can be of integer or float data types.

   The designer aims for high cohesion and low coupling, i.e., a cohesive component (module) focuses on a single function with little interaction with other modules of the system.



From all of the above, the differences between cohesion and coupling can be explained in the following way:

	Cohesion	Coupling
1	Cohesion is defined as the degree of relationship between elements of the same module.	Coupling is defined as the degree of interdependence between the modules.
2	High cohesion is preferred due to improved focus on a particular task.	Low Coupling is preferred as it results in less dependency between the modules.
3	Cohesion is used to indicate a module's relative functional strength.	Coupling is used to indicate the relative independence among the modules.
4	In cohesion, the module focuses on a particular task.	In coupling, a particular module is connected to other modules.
5	Cohesion is also known by the name 'Intra- module Binding'.	Coupling is also known by the name 'Inter-module binding'.