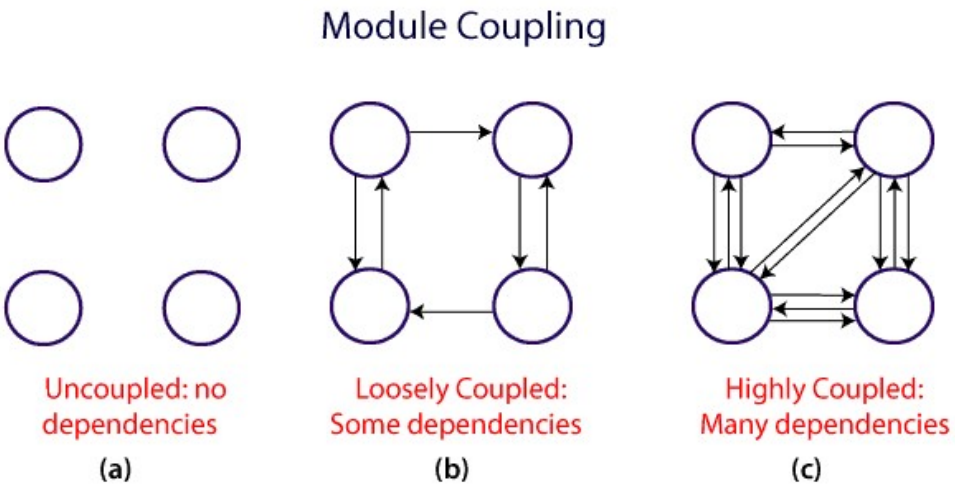
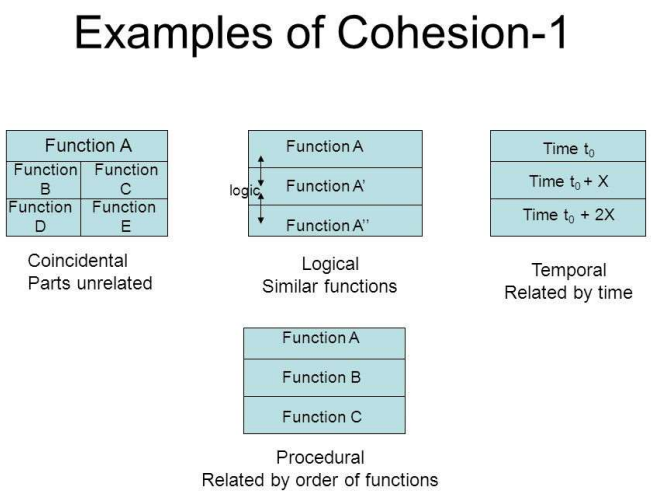
Unit-3 Design
3.3 Coupling and Cohesion





COHESION AND COUPLING

Cohesion is a measure of the functional strength of a module, whereas the coupling between two modules is a measure of the degree of interaction (or interdependence) between the two modules.

Coupling: Two modules are said to be highly coupled, if either of the following two situations arise:

1. If the function calls between two modules involve passing large chunks of shared data, the modules are tightly coupled.
2. If the interactions occur through some shared data, then also we say that they are highly coupled.

- If two modules either do not interact with each other at all or at best interact by passing no data or only a few primitive data items, they are said to have low coupling.

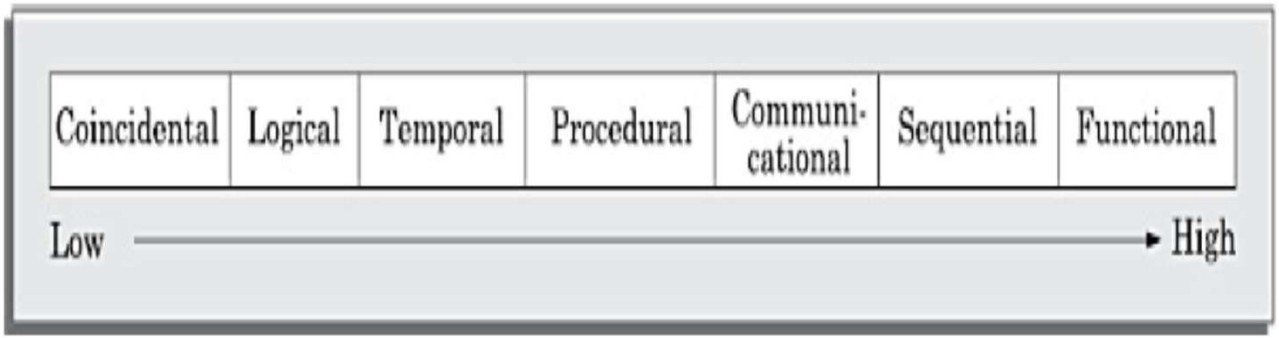
Cohesion

- When the functions of the module co-operate with each other for performing a single objective, then the module has good cohesion.
- If the functions of the module do very different things and do not co-operate with each other to perform a single piece of work, then the module has very poor cohesion.
- **Functional independence:** By the term functional independence, we mean that a module performs a single task and needs very little interaction with other modules.
- A module that is highly cohesive and also has low coupling with other modules is said to be functionally independent of the other modules.
- Functional independence is a key to any good design primarily due to the following advantages it offers:

1. **Error isolation**
2. **Scope of reuse**
3. **Understandability**

Classification of Cohesiveness

- The cohesiveness increases from coincidental to functional cohesion. That is, coincidental is the worst type of cohesion and functional is the best cohesion possible. These different classes of cohesion are elaborated below.



Module Name:
Random–Operations

Function:
Issue–book
Create–member
Compute–vendor–credit
Request–librarian–leave

(a) An example of coincidental cohesion

Module Name:
Managing–Book–Lending

Function:
Issue–book
Return–book
Query–book
Find–borrower

(b) An example of functional cohesion

Coincidental cohesion: A module is said to have coincidental cohesion, if it performs a set of tasks that relate to each other very loosely, if at all. In this case, we can say that the module contains a random collection of functions.

Logical cohesion: A module is said to be logically cohesive, if all elements of the module perform similar operations, such as error handling, data input, data output, etc.

Temporal cohesion: When a module contains functions that are related by the fact that these functions are executed in the same time span

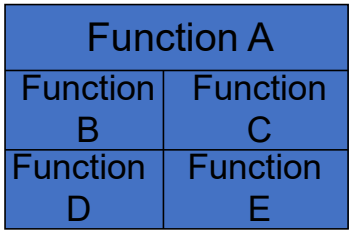
Procedural cohesion: A module is said to possess procedural cohesion, if the set of functions of the module are executed one after the other, though these functions may work towards entirely different purposes and operate on very different data.

Communicational cohesion: A module is said to have communicational cohesion, if all functions of the module refer to or update the same data structure.

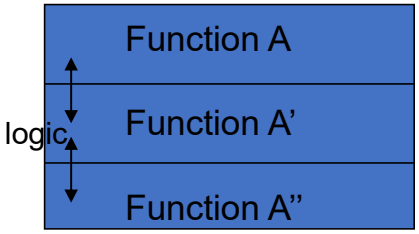
Sequential cohesion: A module is said to possess sequential cohesion, if the different functions of the module execute in a sequence, and the output from one function is input to the next in the sequence.

Functional cohesion: A module is said to possess functional cohesion, if different functions of the module co-operate to complete a single task.

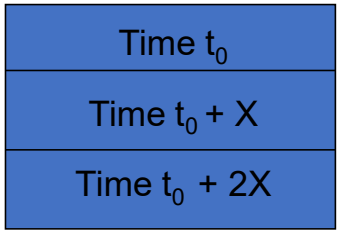
Examples of Cohesion-1



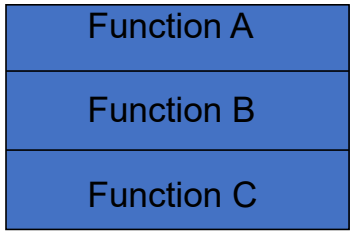
Coincidental
Parts unrelated



Logical
Similar functions

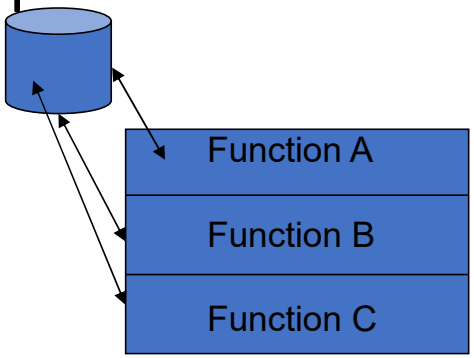


Temporal
Related by time

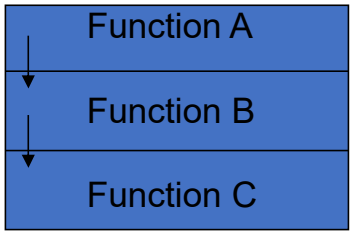


Procedural
Related by order of functions

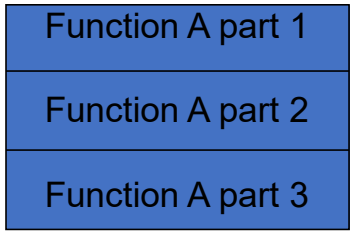
Examples of Cohesion-2



Communicational
Access same data



Sequential
Output of one is input to another



Functional
Sequential with complete, related functions

Classification of Coupling

The coupling between two modules indicates the degree of interdependence between them.

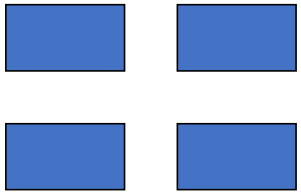
Intuitively, if two modules interchange large amounts of data, then they are highly interdependent or coupled. We can alternately state this concept as follows.

The degree of coupling between two modules depends on their interface complexity.

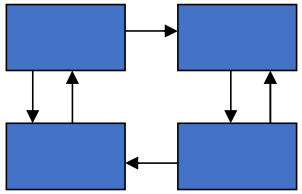
Between any two interacting modules, any of the following five different types of coupling can exist.

The interface complexity is determined based on the number of parameters and the complexity of the parameters that are interchanged while one module invokes the functions of the other module.

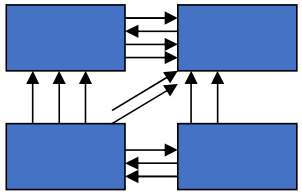
Coupling: Degree of dependence among components



No dependencies



Loosely coupled-some dependencies



Highly coupled-many dependencies

High coupling makes modifying parts of the system difficult, e.g., modifying a component affects all the components to which the component is connected.

Data coupling: Two modules are data coupled, if they communicate using an elementary data item that is passed as a parameter between the two.

Stamp coupling: Two modules are stamp coupled, if they communicate using a composite data item such as a structure in C.

Control coupling: Control coupling exists between two modules, if data from one module is used to direct the order of instruction execution in another. 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 some global data items.

Content coupling: Content coupling exists between two modules, if they share code. That is, a jump from one module into the code of another module can occur.