Component level design.

Coupling and cohesion

Intro

- As soon as the first iteration of architectural design is complete, component-level design takes place.
- The objective of this design is to transform the design model into functional software.
- To achieve this objective, the component-level design represents -the internal data structures and processing details of all the software components (defined during architectural design) at an abstraction level, closer to the actual code.
- In addition, it specifies an interface that may be used to access the functionality of all the software components.

Cont…

- The component-level design can be represented by using different approaches.

- One approach is to use a programming language while other is to use some intermediate design notation such as graphical (DFD, flowchart, or structure chart), tabular (decision table), or text-based (program design language) whichever is easier to be translated into source code.

- The component-level design provides a way to determine whether the defined algorithms, data structures, and interfaces will work properly.

Property of modular design

- **Provide simple interface:**
  - Simple interfaces decrease the number of interactions.
  - Simple interfaces also provide support for reusability of components which reduces the cost to a greater extent.
  - Note that the number of interactions is taken into account while determining whether the software performs the desired function.
- **Ensure information hiding:**
  - The benefits of modularity cannot be achieved merely by decomposing a program into several modules; rather each module should be designed and developed in such a way that the information hiding is ensured.

Coupling

- Coupling measures the degree of interdependence among the modules.
- Influence the strength of coupling between two modules.
- For better interface and well-structured system, the modules should be loosely coupled in order to minimize the 'ripple effect' in which modifications in one module results in errors in other modules.

Coupling categorization

❖**No direct coupling:**

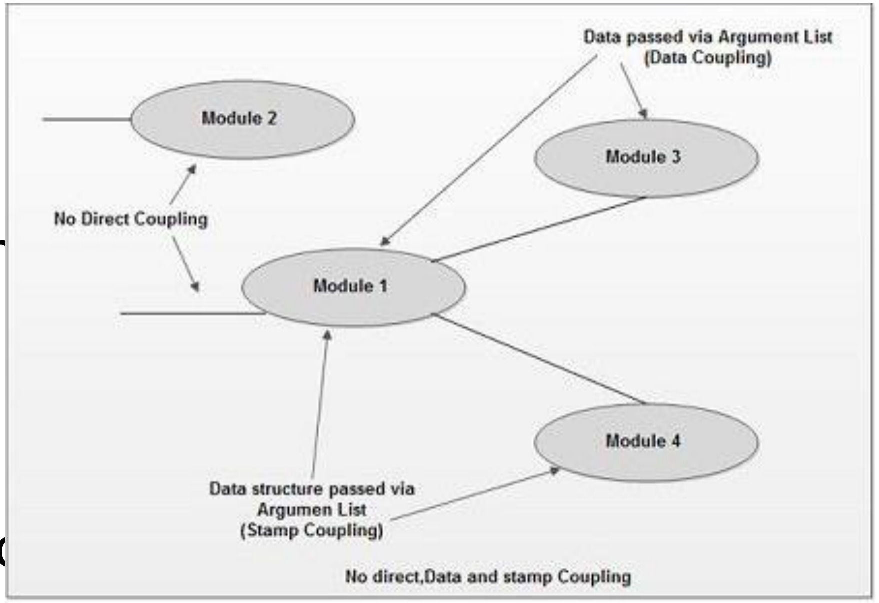
➢Two modules are said to be 'no direct coupled' if they are independent of each other.

❖**Data coupling:**

➢Two modules are said to be 'data coupled' if th data items for communication.

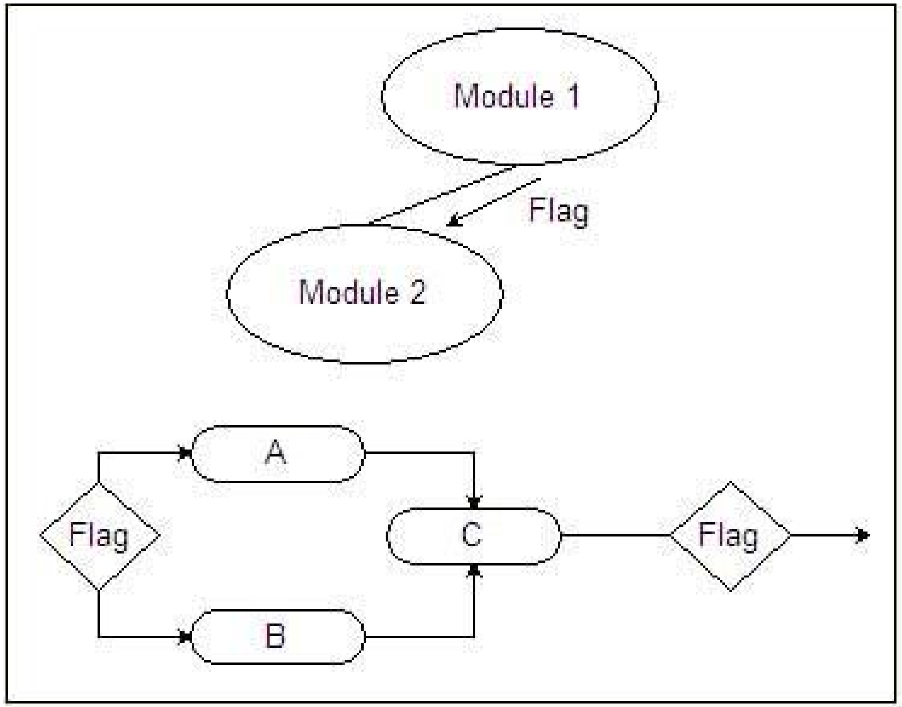
❖**Stamp coupling:**

➢Two modules are said to be 'stamp coupled' if data structure that stores additional informatic perform their functions.

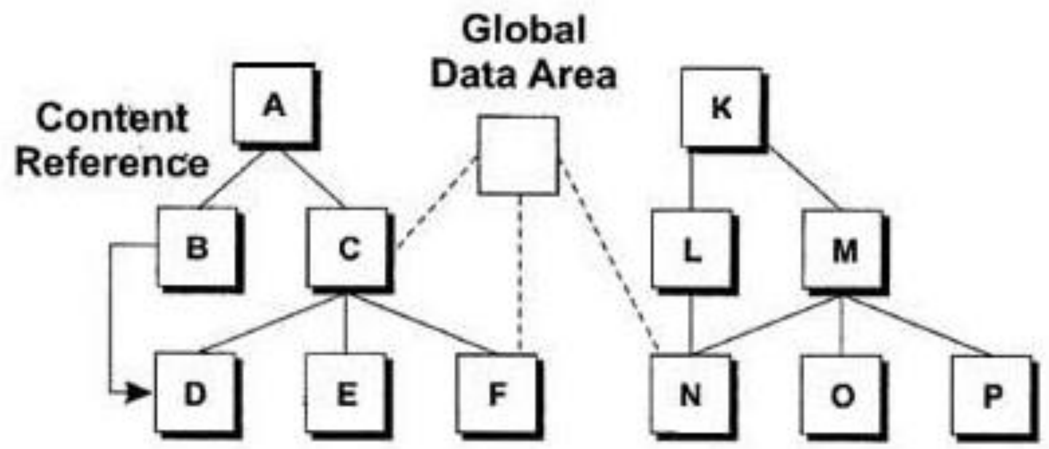


Cont…

- **Control coupling:**
  - Two modules are said to be 'control coupled' if they communicate (pass a piece of information intended to control the internal logic) using at least one 'control flag'.
  - The control flag is a variable whose value is used by the dependent modules to make decisions.



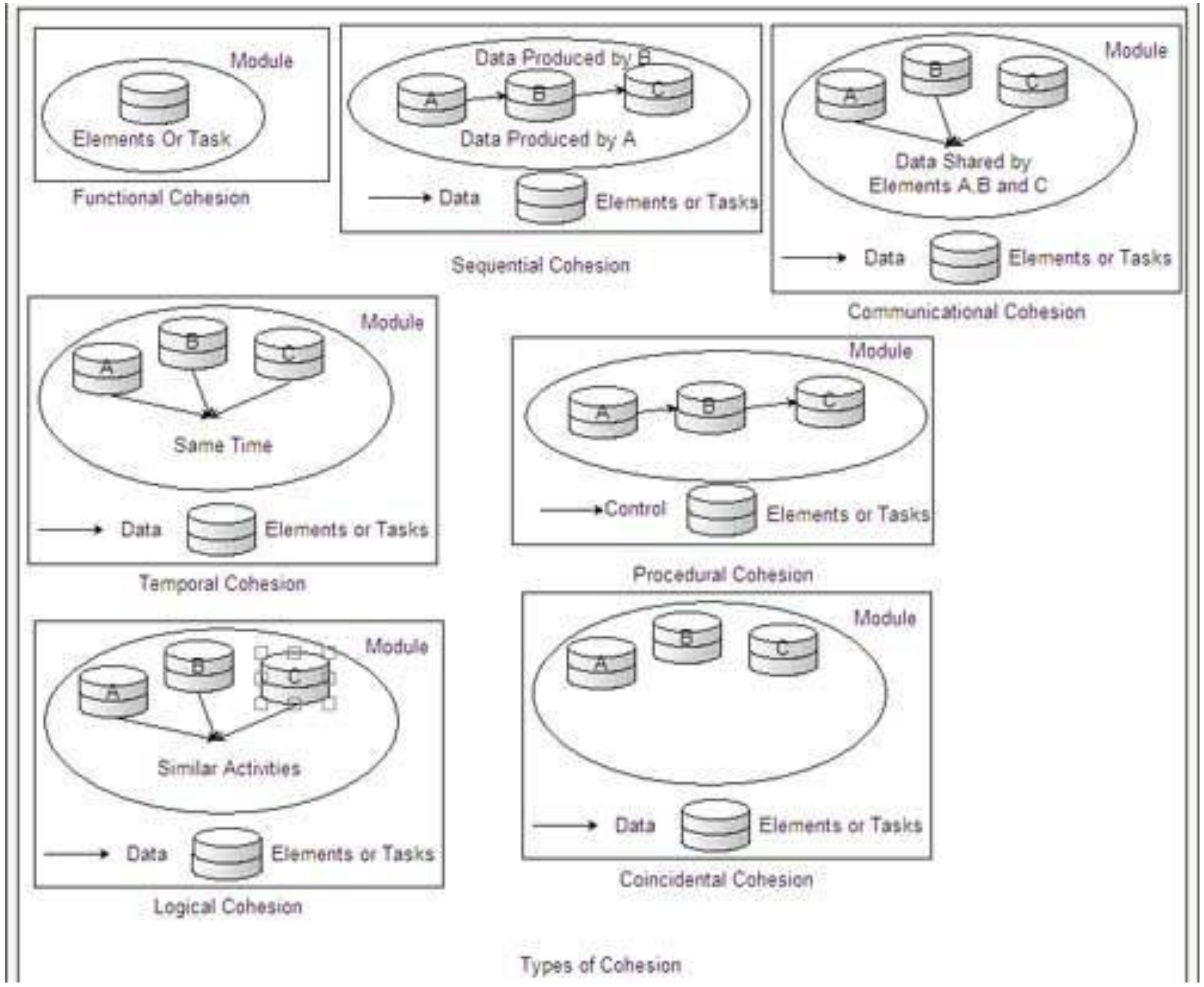
- **Content coupling:**
  - Two modules are said to be 'content coupled' if one module modifies data of some other module or one module is under the control of another module or one module branches into the middle of another module.



Cohesion

- Cohesion measures the relative functional strength of a module.
- It represents the strength of bond between the internal elements of the modules.
- The tighter the elements are bound to each other, the higher will be the cohesion of a module.
- In practice, designers should avoid a low level of cohesion when designing a module.
- Generally, low coupling results in high cohesion and vice versa.

Types of cohesion



Types of cohesion

- **Functional cohesion**
  - In this, the elements within the modules are concerned with the execution of a single function.
- **Communicational cohesion**
  - In this, the elements within the modules perform different functions, yet each function references the same input or output information.
- **Procedural cohesion**
  - In this, the elements within the modules are involved in different and possibly unrelated activities.

Cont.

- **Temporal cohesion**
  - In this, the elements within the modules contain unrelated activities that can be carried out at the same time.
- **Logical cohesion**
  - In this, the elements within the modules perform similar activities, which are executed from outside the module
- **Coincidental cohesion**
  - In this, the elements within the modules perform activities with no meaningful relationship to one another.