Software Design

Data coupling	Best
Stamp coupling	↑
Control coupling	↑
External coupling	↑
Common coupling	↑
Content coupling	Worst

Fig. 7 : The types of module coupling

Given two procedures A & B, we can identify number of ways in which they can be coupled.

Software Design

Data coupling

The dependency between module A and B is said to be data coupled if their dependency is based on the fact they communicate by only passing of data. Other than communicating through data, the two modules are independent.

Stamp coupling

Stamp coupling occurs between module A and B when complete data structure is passed from one module to another.

Software Design

Control coupling

Module A and B are said to be control coupled if they communicate by passing of control information. This is usually accomplished by means of flags that are set by one module and reacted upon by the dependent module.

Common coupling

With common coupling, module A and module B have shared data. Global data areas are commonly found in programming languages. Making a change to the common data means tracing back to all the modules which access that data to evaluate the effect of changes.

Software Design

Content coupling

Content coupling occurs when module A changes data of module B or when control is passed from one module to the middle of another. In Fig. 9, module B branches into D, even though D is supposed to be under the control of C.

Software Design

Functional Cohesion	Best (high)
Sequential Cohesion	
Communicational Cohesion	
Procedural Cohesion	
Temporal Cohesion	
Logical Cohesion	
Coincidental Cohesion	Worst (low)

Fig. 11 : Types of module cohesion

Software Design

Functional Cohesion

➢ A and B are part of a single functional task. This is very good reason for them to be contained in the same procedure.

Sequential Cohesion

➢ Module A outputs some data which forms the input to B. This is the reason for them to be contained in the same procedure.

Software Design

Procedural Cohesion

- Procedural Cohesion occurs in modules whose instructions although accomplish different tasks yet have been combined because there is a specific order in which the tasks are to be completed.

Temporal Cohesion

- Module exhibits temporal cohesion when it contains tasks that are related by the fact that all tasks must be executed in the same time-span.

Software Design

Logical Cohesion

➢ Logical cohesion occurs in modules that contain instructions that appear to be related because they fall into the same logical class of functions.

Coincidental Cohesion

➢ Coincidental cohesion exists in modules that contain instructions that have little or no relationship to one another.