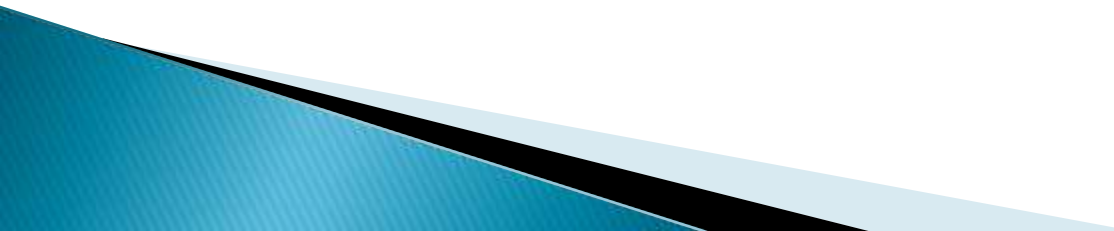



OCS551 Software Engineering

Coupling and Cohesion

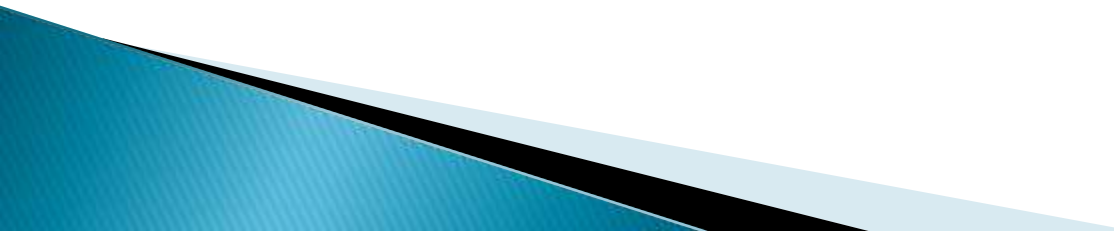
What is Software Design?

- ▶ Software design is a process to transform user requirements into some suitable form, which helps the programmer in software coding and implementation.
 - ▶ The activity that leads from requirements to implementation.
- 


Software Design Levels

- ▶ Software design yields three levels of results:
 - ▶ **Architectural Design** – The architectural design is the highest abstract version of the system. It identifies the software as a system with many components interacting with each other. At this level, the designers get the idea of proposed solution domain.
 - ▶ **High-level Design**– High-level design focuses on how the system along with all of its components can be implemented in forms of modules. It recognizes modular structure of each sub-system and their relation and interaction among each other.
 - ▶ **Detailed Design**– Detailed design deals with the implementation part of what is seen as a system and its sub-systems in the previous two designs. It is more detailed towards modules and their implementations.
- 

Modularization

- ▶ Modularization is a technique to divide a software system into multiple discrete and independent modules, which are expected to be capable of carrying out task(s) independently. Designers tend to design modules such that they can be executed and/or compiled separately and independently.
- 

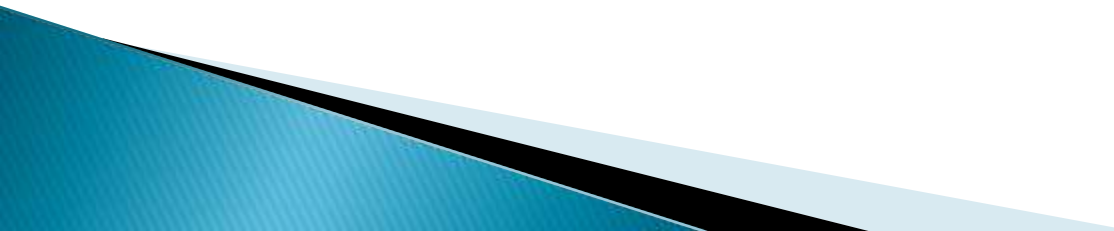
Advantages of modularization:

- ▶ Smaller components are easier to maintain
 - ▶ Program can be divided based on functional aspects
 - ▶ Desired level of abstraction can be brought in the program
 - ▶ Components with high cohesion can be re-used again
 - ▶ Concurrent execution can be made possible
- 

Coupling and Cohesion

- ▶ When a software program is modularized, its tasks are divided into several modules based on some characteristics. There are measures by which the quality of a design of modules and their interaction among them can be measured. These **measures are called coupling and cohesion.**

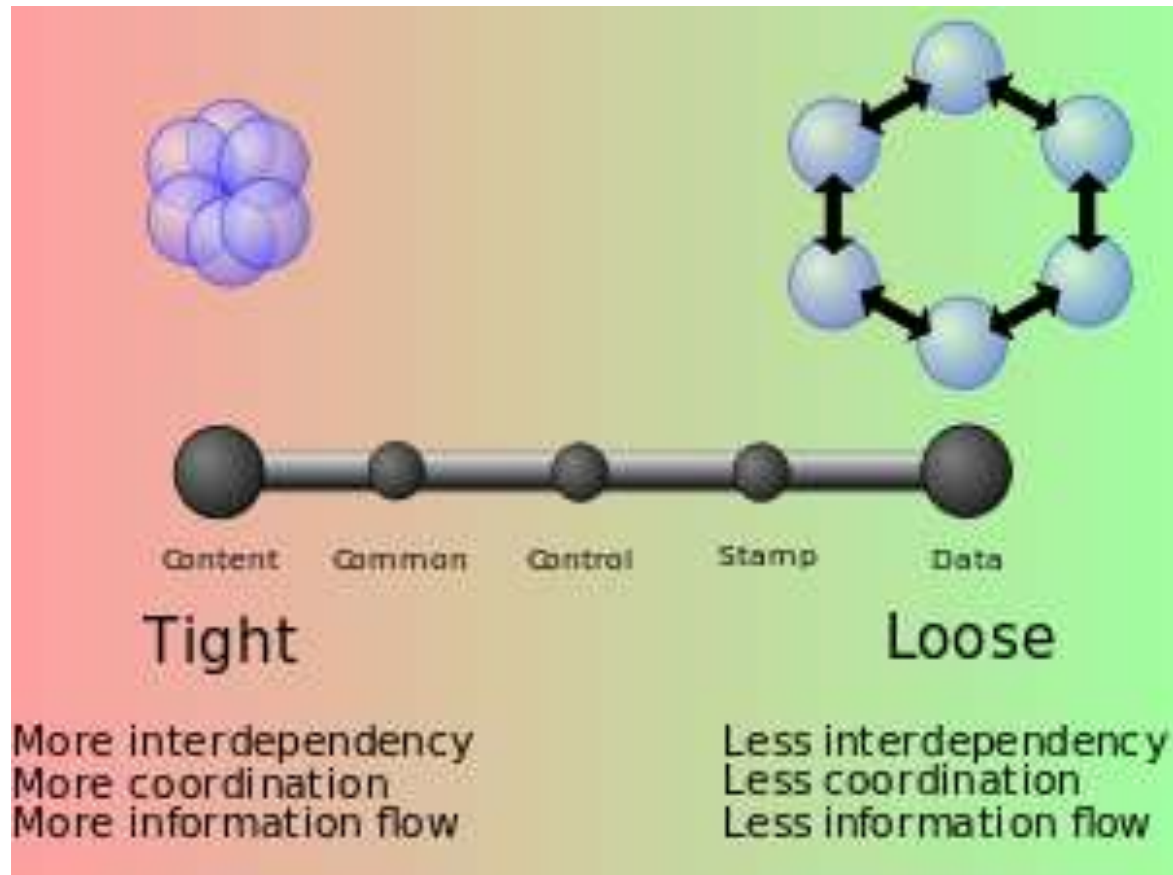
Coupling and Cohesion

- ▶ Coupling – concerns relationships between modules
 - ▶ Cohesion – concerns relationships within a module
 - Goal: To have loosely coupled modules with high internal cohesion
- 

Coupling

- ▶ Coupling is defined as the extent to which a system, subsystem, method or module connects with (depends on) others. In other words, it measures interdependency. The lower the coupling, the better the program.
- ▶ Coupling increases as the number of calls between modules increase or the amount of shared data is large

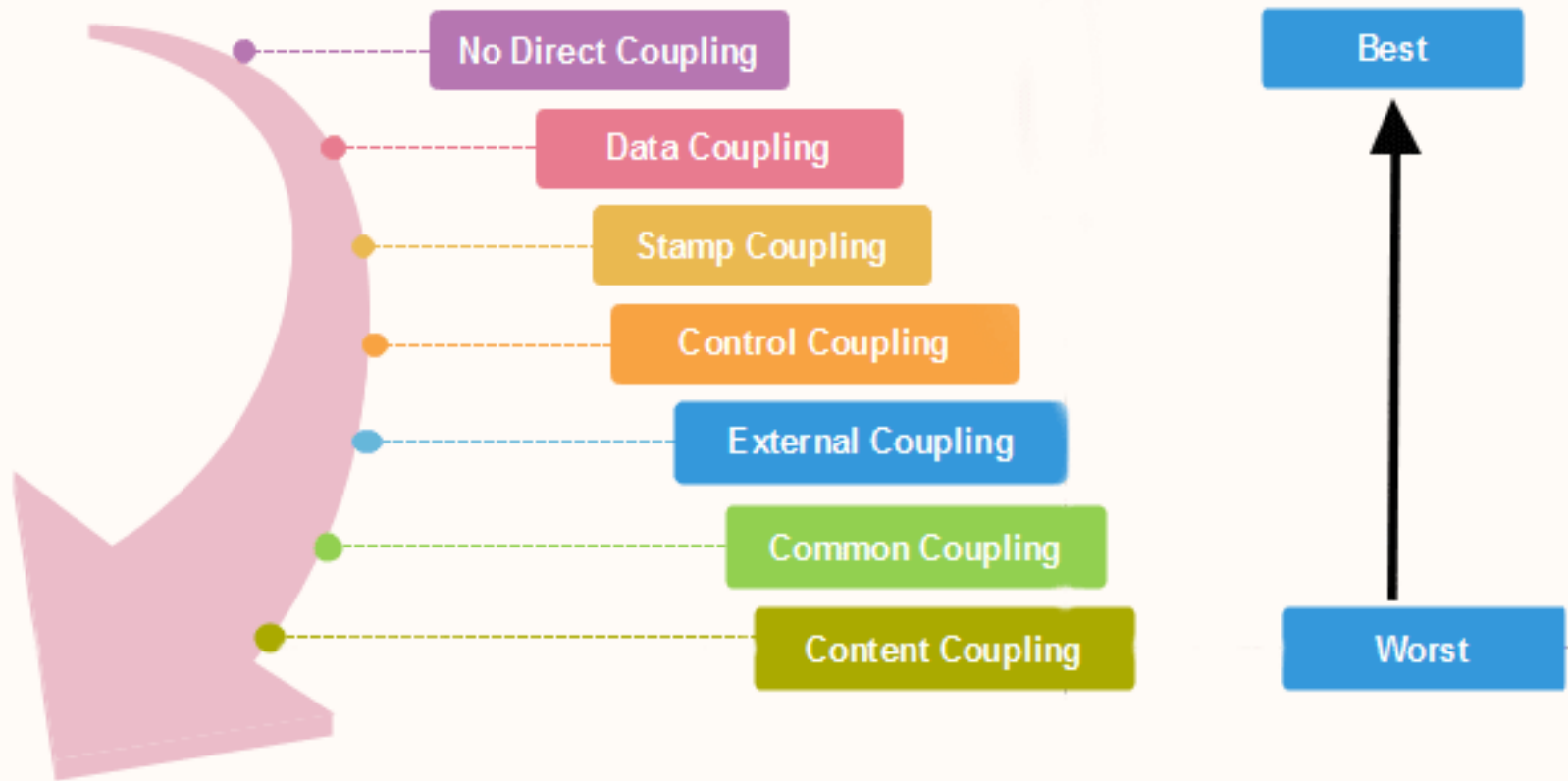
Coupling



Coupling

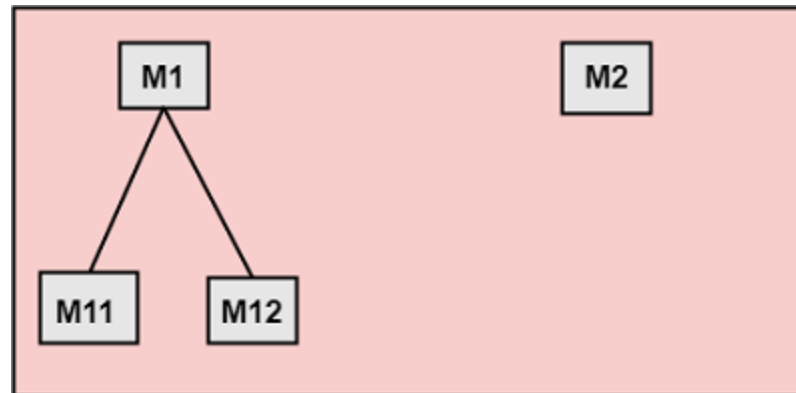
Types of Modules Coupling

There are various types of module Coupling are as follows:



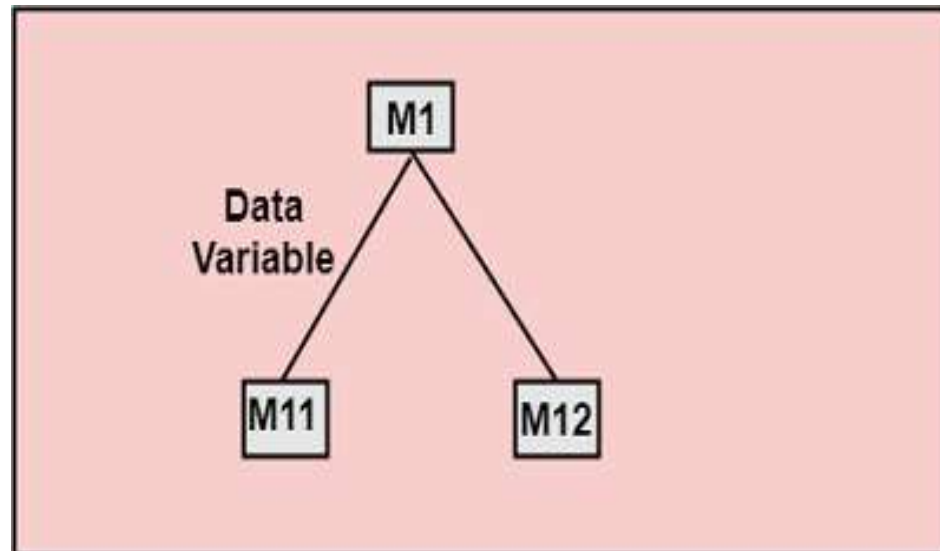
1. No Direct Coupling

- ▶ There is no direct coupling between M1 and M2. Modules are subordinates to different modules. Therefore, no direct coupling.

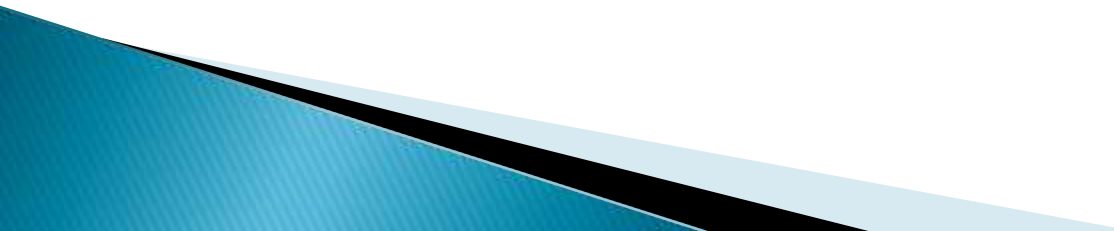


2. Data Coupling

- ▶ When data of one module is passed to another module, this is called data coupling.
- ▶ Example–customer billing system.



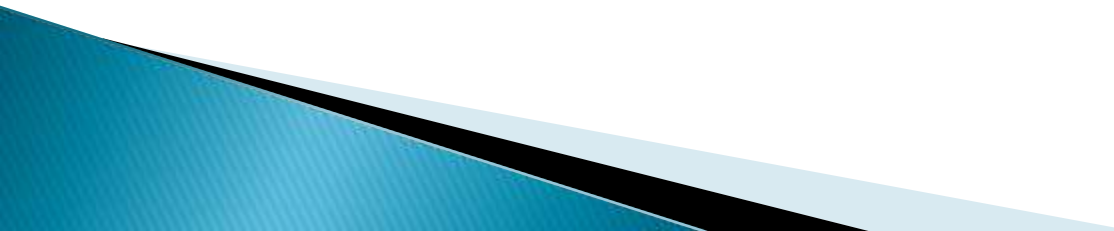
3. Stamp Coupling

- ▶ 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.
- 

4. Control Coupling

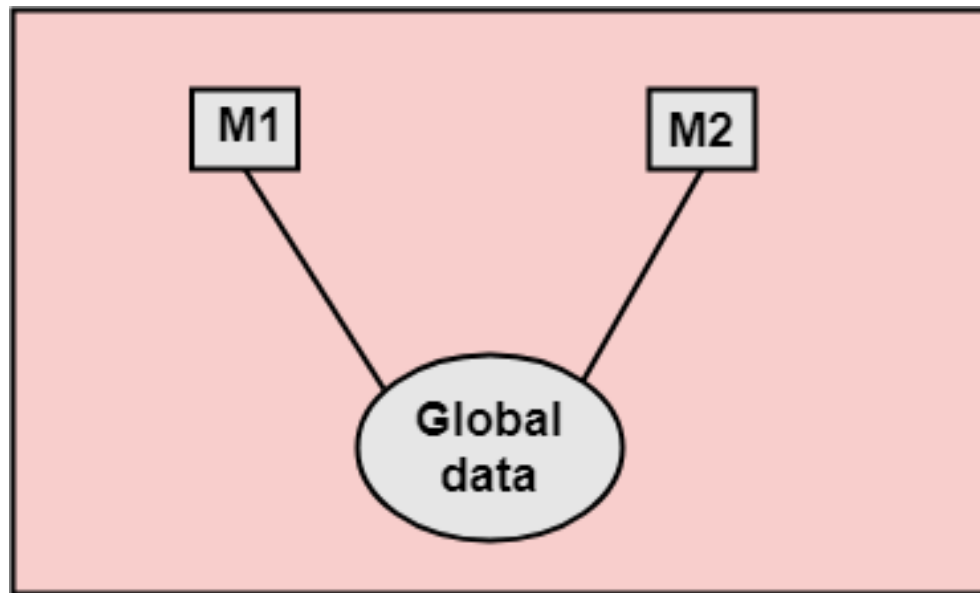
- ▶ Control Coupling exists among two modules if data from one module is used to direct the structure of instruction execution in another.

5. External Coupling

- ▶ External Coupling arises when two modules share an externally imposed data format, communication protocols, or device interface. This is related to communication to external tools and devices.
 - ▶ Ex- protocol, external file, device format, etc.
- 

6. Common Coupling


- ▶ Two modules are common coupled if they share information through some global data items.



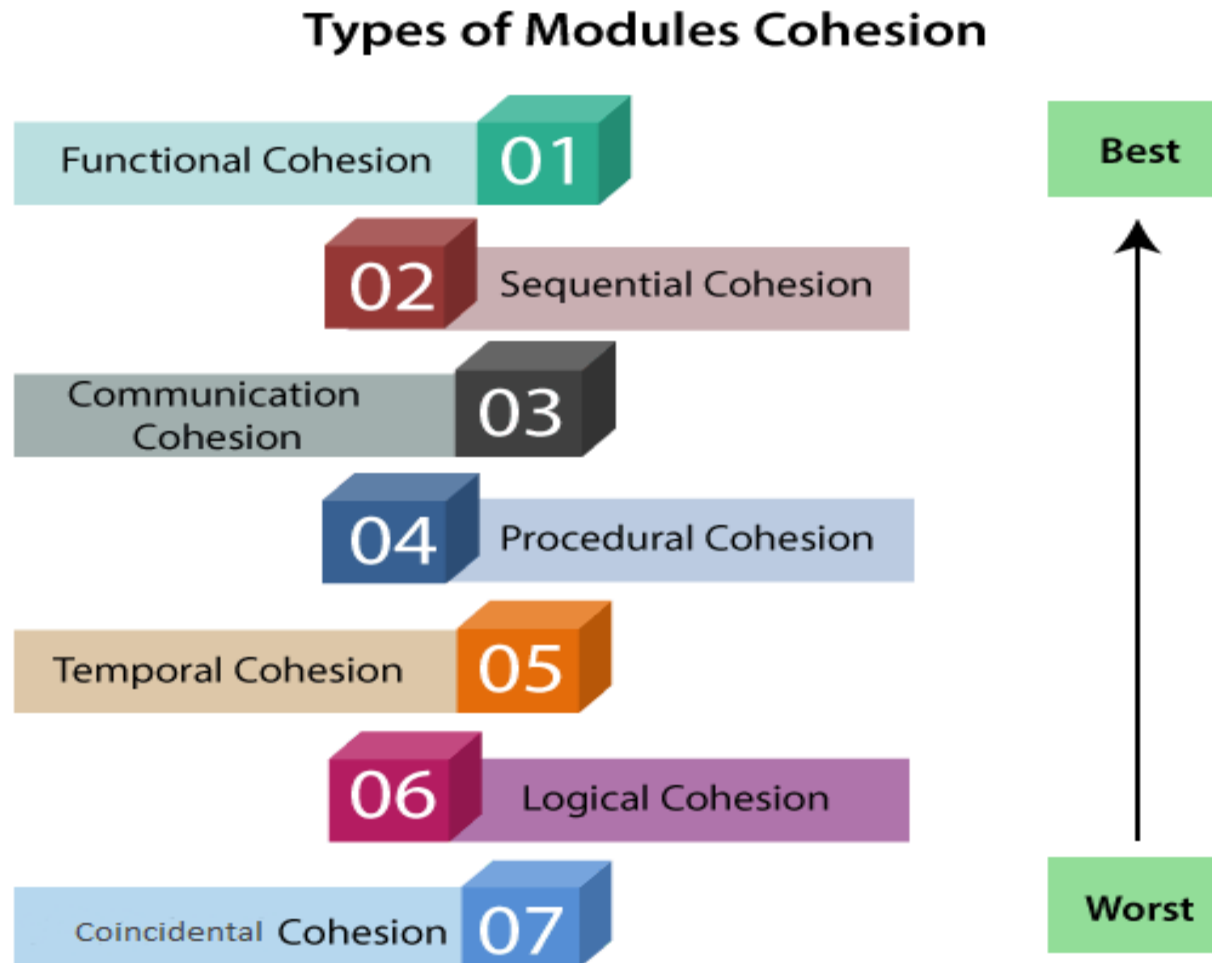
7. Content Coupling

- ▶ Content Coupling exists among two modules if they share code, e.g., a branch from one module into another module.

Cohesion

- ▶ Cohesion defines to the degree to which the elements of a module belong together. Thus, cohesion measures the strength of relationships between pieces of functionality within a given module. For example, in highly cohesive systems, functionality is strongly related. A good software design will have high cohesion.
- 

Types of Modules Cohesion



1. Functional Cohesion

- ▶ Functional Cohesion is said to exist if the different elements of a module, cooperate to achieve a single function.

2. Sequential Cohesion

- ▶ A module is said to possess sequential cohesion if the elements of a module form the components of the sequence, where the output from one component of the sequence is input to the next.

3. Communicational Cohesion

- ▶ A module is said to have communicational cohesion, if all tasks of the module refer to or update the same data structure, e.g., the set of functions defined on an array or a stack.

4. Procedural Cohesion

- ▶ A module is said to be procedural cohesion if the set of purpose of the module are all parts of a procedure in which particular sequence of steps has to be carried out for achieving a goal, e.g., the algorithm for decoding a message.

5. Temporal Cohesion

- ▶ When a module includes functions that are associated by the fact that all the methods must be executed in the same time, the module is said to exhibit temporal cohesion.



6. Logical Cohesion

- ▶ A module is said to be logically cohesive if all the elements of the module perform a similar operation. For example Error handling, data input and data output, etc.

7. Coincidental Cohesion

- ▶ A module is said to have coincidental cohesion if it performs a set of tasks that are associated with each other very loosely, if at all.

Differentiate between Coupling and Cohesion

S.No	Coupling	Cohesion
1	Cohesion is the concept of intra module.	Coupling is the concept of inter module.
2	Cohesion represents the relationship within module.	Coupling represents the relationships between modules.
3	Increasing in cohesion is good for software.	Increasing in coupling is avoided for software.
4	Cohesion represents the functional strength of modules.	Coupling represents the independence among modules.
5	High cohesion gives the best software.	Loose coupling gives the best software.
6	In cohesion, module focuses on the single thing.	In coupling, modules are connected to the other modules.