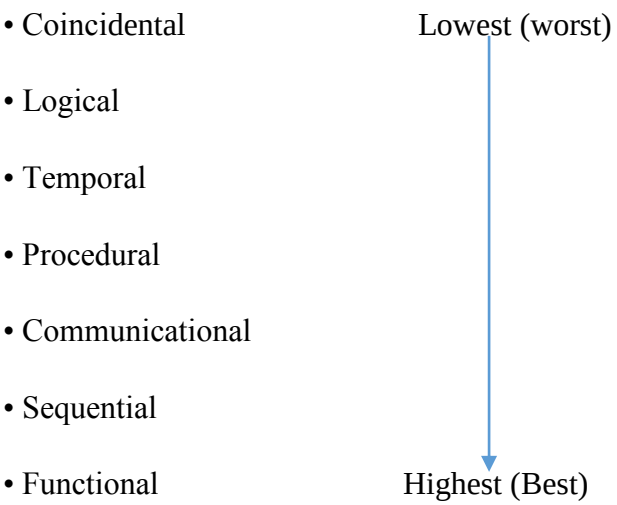
6.2.2 Cohesion

Coupling is reduced when elements in different modules have little or no bonds between them. **With cohesion, we are interested in determining how closely the elements of a module are related to each other.** Cohesion of a module gives the designer an idea about whether the different elements of a module belong together in the same module. Usually, the greater the cohesion of each module in the system, the lower the coupling between modules is. There are several levels of cohesion:



These levels do not form a linear scale.

Coincidental cohesion occurs when there is no meaningful relationship among the elements of a module. It occurs when an existing program is "modularized" by chopping it into pieces and making different pieces modules. For Example, if one of the modules code needs to be modified and this modification includes the common code, it is likely that other modules using the code do not want the code modified. Consequently, the modification of this "common module" may cause other modules to behave incorrectly. The modules using these modules are therefore not modifiable separately and have strong interconnection between them. We can say that, it is poor practice to chop a module into smaller modules to reduce the module size.

A module has **logical cohesion** if there is some logical relationship between the elements of a module, and the elements perform functions that fall in the same logical class. For example, a module that performs all the inputs or all the outputs. In such a situation, if we want to input or output a particular record, we have to somehow convey this to the module. Often, this will be done by passing some kind of special status flag, which will be used to determine what statements to execute in the module.

Temporal cohesion is the same as logical cohesion, except that the elements are also related in time and are executed together. Modules that perform activities like "initialization," "clean-up," and "termination" are usually temporally bound. This avoids the problem of passing the flag, and the code is usually simpler.

A procedurally cohesive module contains elements that belong to a common procedural unit. For example, a loop or a sequence of decision statements in a module may be combined to form a separate module. A module with only procedural cohesion may contain only part of a complete function or parts of several functions.

A module with **communicational cohesion** has elements that are related by a reference to the same input or output data. An example of this could be a module to "print and punch record." Communicational cohesive modules may perform more than one function. However, communicational cohesion is sufficiently high as to be generally acceptable if alternative structures with higher cohesion cannot be easily identified.

We get **sequential cohesion** when the elements are together in a module because the output of one forms the input to another. If we have a sequence of elements in which the output of one forms the input to another, sequential cohesion does not provide any guidelines on how to combine them into modules. Different possibilities exist: combine all in one module, put the first half in one and the second half in another, the first third in one and the rest in the other, and so forth.

Functional cohesion is the strongest cohesion. In a functionally bound module, all the elements of the module are related to performing a single function. By function, we do not mean simply mathematical functions; modules accomplishing a single goal are also included. Functions like "compute square root" and "sort the array" are clear examples of functionally cohesive modules.

A useful technique for determining if a module has functional cohesion is to write a sentence that describes the function or purpose of the module. The following tests can then be made:-

1. If the sentence must be a compound sentence, if it contains a comma, or it has more than one verb, the module is probably performing more than one function, and it probably has sequential or communicational cohesion.

2. If the sentence contains words relating to time, like "first," "next," "when," and "after" the module probably has sequential or temporal cohesion.

3. If the predicate of the sentence does not contain a single specific object following the verb (such as "edit all data") the module probably has logical cohesion.

4. Words like "initialize" and "cleanup" imply temporal cohesion.