# Coupling and Cohesion

When a software program is modularized, its tasks are divided into several modules based on some characteristics. As we know, modules are set of instructions put together in order to achieve some tasks. They are though, considered as single entity but may refer to each other to work together.

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.

## Cohesion

Cohesion is a measure that defines the **degree of intra-dependability within elements of a module**. The greater the cohesion, the better is the program design. Cohesion is a measure that defines the degree of intra-dependability within elements of a module. The greater the cohesion, the better is the program design.

There are seven types of cohesion, namely –

### Co-incidental cohesion

It is unplanned and random cohesion, which might be the result of breaking the program into smaller modules for the sake of modularization. Because it is unplanned, it may serve confusion to the programmers and is generally not-accepted.

### Logical cohesion

When logically categorized elements are put together into a module, it is called logical cohesion.

### Temporal Cohesion

When elements of module are organized such that they are processed at a similar point in time, it is called temporal cohesion.

### Procedural cohesion

When elements of module are grouped together, which are executed sequentially in order to perform a task, it is called procedural cohesion.

### Communicational cohesion

When elements of module are grouped together, which are executed sequentially and work on same data (information), it is called communicational cohesion.

### Sequential cohesion

When elements of module are grouped because the output of one element serves as input to another and so on, it is called sequential cohesion.

### Functional cohesion

It is considered to be the highest degree of cohesion, and it is highly expected. Elements of module in functional cohesion are grouped because they all contribute to a single well-defined function. It can also be reused.

## Coupling

Coupling is a measure that defines the level of **inter-dependability among modules of a program**. It tells at what level the modules interfere and interact with each other. The lower the coupling, the better the program.

There are five levels of coupling, namely -

### Content coupling

When a module can directly access or modify or refer to the content of another module, it is called content level coupling.

### Common coupling

When multiple modules have read and write access to some global data, it is called common or global coupling.

### Control coupling

Two modules are called control-coupled if one of them decides the function of the other module or changes its flow of execution.

### Stamp coupling

When multiple modules share common data structure and work on different part of it, it is called stamp coupling.

### Data coupling

Data coupling is when two modules interact with each other by means of passing data (as parameter). If a module passes data structure as parameter, then the receiving module should use all its components.

Ideally, no coupling is considered to be the best.

# Aggregation: Classes Within Classes

Aggregation is not directly related to inheritance, both aggregation and inheritance are class relationships that are more specialized than associations.

In Inheritance, if a class B is derived by inheritance from a class A, we can say that “B is a kind of A.” For this reason, **inheritance** is often called a **“is a kind of”** relationship. Example: starling is a kind of bird.

**Aggregation** is called a **“has a”** relationship. Aggregation is also called a “part-whole” relationship. Example: the book is part of the library.

In object-oriented programming, aggregation may occur when one object is an attribute of another. Here’s a case where an object of class A is an attribute of class B:

class A {

};

class B {

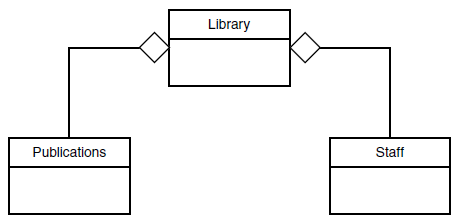
A objA; // define objA as an object of class A

};

In the UML, aggregation is considered a special kind of association.

Sometimes it’s hard to tell when an association is also an aggregation. It’s always safe to call a relationship an association, but if class A contains objects of class B, and is organizationally superior to class B, it’s a good candidate for aggregation.

Aggregation is shown in the same way as association in UML class diagrams, except that the “whole” end of the association line has an open diamond-shaped arrowhead.



UML class diagram showing aggregation.