Software Engineering

Coupling and Cohesion

Overview :

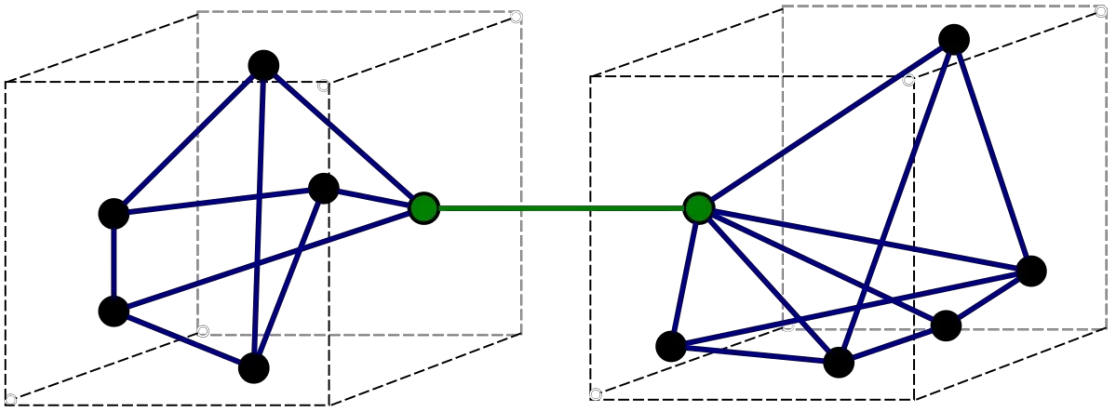
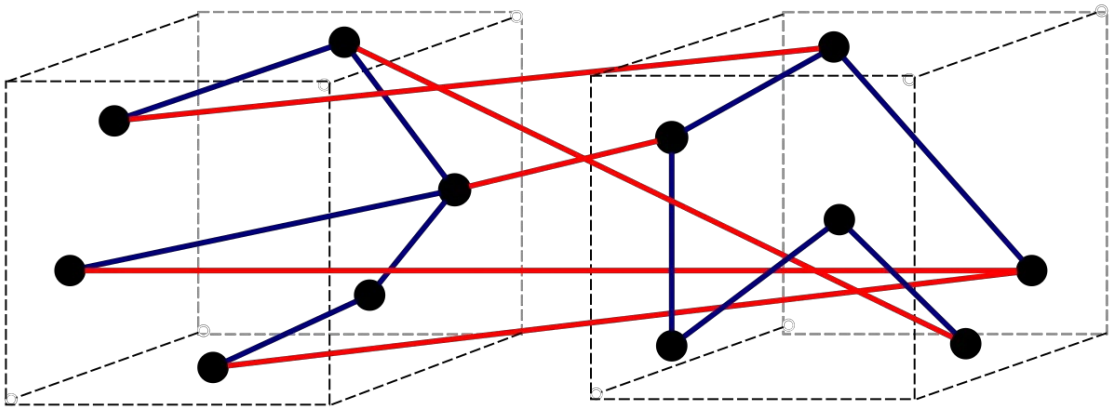
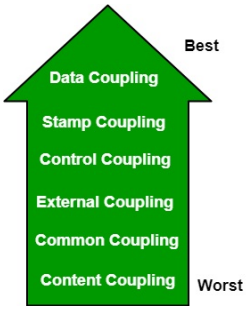
- In this shot, we'll be discussing the importance of coupling & cohesion in software engineering
- But before moving into this topic, we should know what modularization is so we can grasp the concept of coupling easily

Modularization

- Modularization is the process of breaking a software into multiple small modules, where each module works independently.
- The main advantage of modularization is that it makes it easy to understand the software, makes it reusable, and it can be tested easily.

Coupling in Software Engineering

- Coupling is the inter-dependency or degree of relationship between multiple modules/packages/components. Coupling is also called Inter-module Binding.
- Multiple modules/packages/components that are highly coupled are strongly dependent on each other. Multiple modules/packages/components that are loosely coupled are not or somehow dependent on each other.

Example Image	Image for Types of Coupling
 a) Good (loose coupling, high cohesion)  b) Bad (high coupling, low cohesion)	

Types of Coupling:

- **Data Coupling:**
  - If the dependency between the modules is based on the fact that they communicate by passing only data, then the modules are said to be data coupled.
  - In data coupling, the components are independent of each other and communicate through data.
  - Module communications don't contain tramp data. Example-customer billing system.

- **Stamp Coupling**
  - In stamp coupling, the complete data structure is passed from one module to another module. Therefore, it involves tramp data.
  - It may be necessary due to efficiency factors- this choice was made by the insightful designer, not a lazy programmer.

- **Control Coupling:**
  - If the modules communicate by passing control information, then they are said to be control coupled.
  - It can be bad if parameters indicate completely different behavior and good if parameters allow factoring and reuse of functionality.
  - Example- sort function that takes comparison function as an argument.

- **External Coupling:**
  - In external coupling, the modules depend on other modules, external to the software being developed or to a particular type of hardware.
  - Ex- protocol, external file, device format, etc.

- **Common Coupling:**
  - The modules have shared data such as global data structures. The changes in global data mean tracing back to all modules which access that data to evaluate the effect of the change.
  - So it has got disadvantages like difficulty in reusing modules, reduced ability to control data accesses, and reduced maintainability.

- **Content Coupling:**
  - In a content coupling, one module can modify the data of another module, or control flow is passed from one module to the other module.
  - This is the worst form of coupling and should be avoided.

Advantages of low coupling:

- **Improved maintainability**: Low coupling reduces the impact of changes in one module on other modules, making it easier to modify or replace individual components without affecting the entire system.
- **Enhanced modularity**: Low coupling allows modules to be developed and tested in isolation, improving the modularity and reusability of code.
- **Better scalability**: Low coupling facilitates the addition of new modules and the removal of existing ones, making it easier to scale the system as needed.

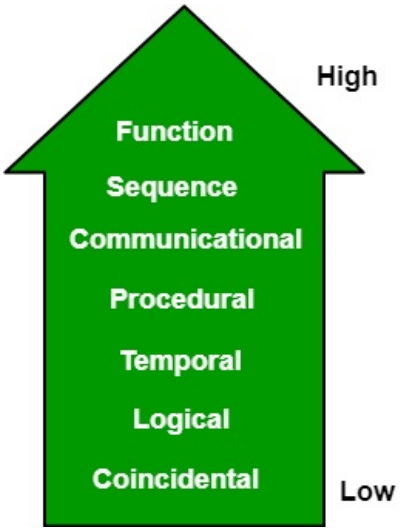
Disadvantages of high coupling:

- **Increased complexity**: High coupling increases the interdependence between modules, making the system more complex and difficult to understand.

- **Reduced flexibility**: High coupling makes it more difficult to modify or replace individual components without affecting the entire system.
- **Decreased modularity**: High coupling makes it more difficult to develop and test modules in isolation, reducing the modularity and reusability of code.

Cohesion in Software Engineering :

- Cohesion is a measure of the degree to which the elements of the module are functionally related.
- It is the degree to which all elements directed towards performing a single task are contained in the component.
- Basically, cohesion is the internal glue that keeps the module together.
- A good software design will have high cohesion.



Types of Cohesion :

- **Functional Cohesion:**
  - Every essential element for a single computation is contained in the component.
  - A functional cohesion performs the task and functions. It is an ideal situation.

- **Sequential Cohesion:**
  - An element outputs some data that becomes the input for other element, i.e., data flow between the parts.
  - It occurs naturally in functional programming languages.

- **Communicational Cohesion:**
  - Two elements operate on the same input data or contribute towards the same output data.
  - Example- update record in the database and send it to the printer.

- **Procedural Cohesion:**
  - Elements of procedural cohesion ensure the order of execution. Actions are still weakly connected and unlikely to be reusable.
  - Ex- calculate student GPA, print student record, calculate cumulative GPA, print cumulative GPA.

- **Temporal Cohesion:**
  - The elements are related by their timing involved. A module connected with temporal cohesion all the tasks must be executed in the same time span.
  - This cohesion contains the code for initializing all the parts of the system. Lots of different activities occur, all at unit time.

- **Logical Cohesion:**
  - The elements are logically related and not functionally. Ex- A component reads inputs from tape, disk, and network.
  - All the code for these functions is in the same component. Operations are related, but the functions are significantly different.

- **Coincidental Cohesion:**
  - The elements are not related(unrelated). The elements have no conceptual relationship other than location in source code. - It is accidental and the worst form of cohesion. Ex- print next line and reverse the characters of a string in a single component.

Advantages of high cohesion:

- **Improved readability and understandability**: High cohesion results in clear, focused modules with a single, well-defined purpose, making it easier for developers to understand the code and make changes.
- **Better error isolation**: High cohesion reduces the likelihood that a change in one part of a module will affect other parts, making it easier to
- **isolate and fix errors**.Improved reliability: High cohesion leads to modules that are less prone to errors and that function more consistently,
- leading to an overall improvement in the reliability of the system.

Disadvantages of low cohesion:

- **Increased code duplication**: Low cohesion can lead to the duplication of code, as elements that belong together are split into separate modules.
- **Reduced functionality**: Low cohesion can result in modules that lack a clear purpose and contain elements that don’t belong together, reducing their functionality and making them harder to maintain.
- **Difficulty in understanding the module**: Low cohesion can make it harder for developers to understand the purpose and behavior of a module, leading to errors and a lack of clarity.

Let's Distinguish Coupling & Cohesion

Coupling	Cohesion
1. Coupling is the concept of inter module .	1. Cohesion is the concept of intra module
2. Coupling represents the relationships between modules.	2.Cohesion represents the relationship within module
3. Increasing in coupling is avoided for software	3. Increasing in cohesion is good for software
4. Coupling represents the independence among modules	4. Cohesion represents the function strength of modules
5. Where a loosely coupling gives the best software	5. Highly cohesive gives the best result

---