Software Engineering

هندسة البرمجيات

جامعة بغداد

كلية التربيه للعلوم الصرفه/ابن الهيثم

قسم علوم الحاسبات

المرحلة الثالثة

م.د.علي يحيى غني

نظري
محاضرة الاسبوع التاسع عشر

Software Design

Topics covered

- ✧ What is Software Design?
- ✧ Characteristics of a good software design.
- ✧ What is Modularity or Modularization?
- ✧ Software Design and its activities (Architectural design and Detailed design).
- ✧ Coupling and Cohesion.

Software Design

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

- For assessing user requirements, an SRS (Software Requirement Specification) document is created whereas for coding and implementation, there is a need of more specific and detailed requirements in software terms. The output of this process can directly be used into implementation in programming languages.

Characteristic of a good software design

Most researchers and software engineers agree on a few desirable characteristics that every good software design for general application must process. The characteristics are listed bellow:

- ✧ Correctness: Implement all functionalities identified in the SRS document.

- ✧ Understandability: A good design is easily understandable.

Characteristic of a good software design

✧ Efficiency: It should be efficient.

✧ Maintainability: It should be easily amenable to change.

Possibly the most important goodness criterion is design correctness. A design has to be correct to be acceptable. Given that a design solution is correct, Understandability of design is possibly the most important issue to be considered while judging the goodness of a design.

Characteristic of a good software design

- A design that is easy to understand is also easy to develop, maintain and changes. Thus, unless a design is easily understandable, it would require tremendous effort to implement and maintain it.

Characteristic of a good software design

According to Frank Tsui and Orlando Karam (2007), a good software design is:

- ✧ Easy to understand.
- ✧ Easy to change.
- ✧ Easy to reuse.
- ✧ Easy to test.
- ✧ Easy to integrate.
- ✧ Easy to code.

What is Modularity or 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. These modules may work as basic constructs for the entire software. Designers tend to design modules such that they can be executed and/or compiled separately and independently.

- Modular design follows the rule of 'divide and conquer' problem-solving strategy, this is because there are many other benefits attached with the modular design of a software.

What is Modularity or Modularization?

Advantage 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
- ✧ Desired from security aspect

Software Design and its activities

- 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.

- Detailed Design: Detailed design deals with the implementation part of what is seen as a system and its sub-systems in the previous design. It is more detailed towards modules and their implementations. It defines logical structure of each module and their interfaces to communicate with other modules.

Coupling and Cohesion

- Coupling often refers to how the modules belong together. Modules should be as independent as possible from other modules. (Low Coupling).
- Cohesion often refers to how the functions of a module belong together. Related code should be close to each other to make it highly cohesive (High Cohesion).

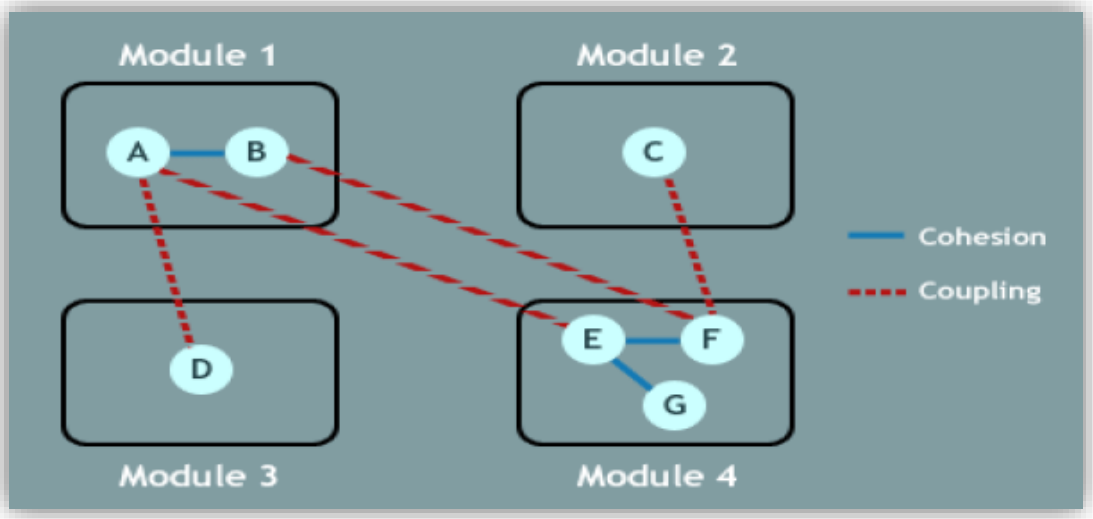

Figure 29: Coupling and Cohesion

"Good Software Design has Low Coupling and High Cohesion"

Coupling and Cohesion

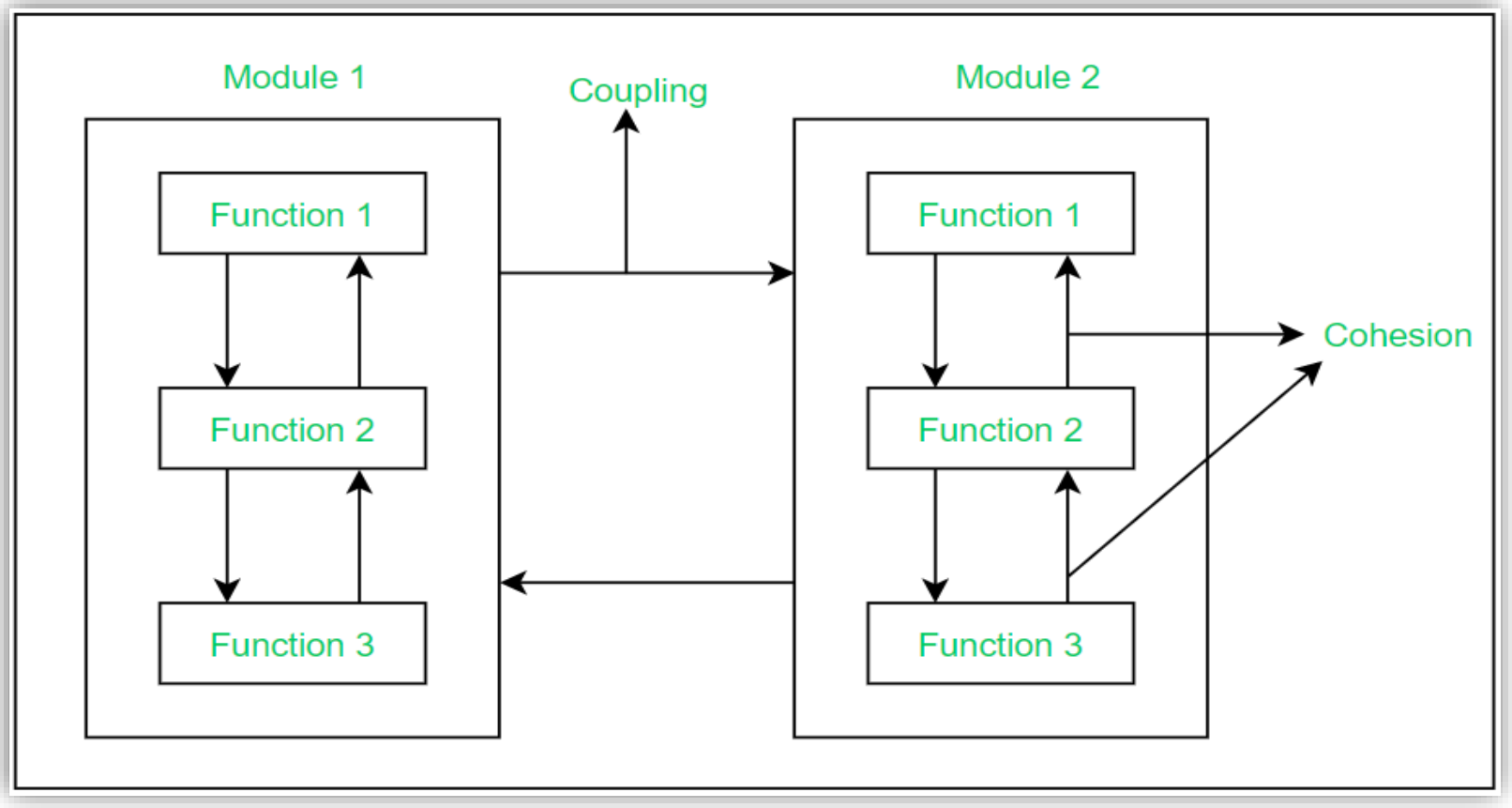

Figure 30: Coupling and Cohesion

Cohesion

- The person class has low cohesion, simply because Person's responsibilities is relevant to save information about people. It do not relate to functionalities about read/write to file. So, to reduce low cohesion, we should separate the implementation about read/write file into other class such as File, …

```
public class Person {
    private int    age;
    private String name;

    // getter, setter properties.

    // method
    public void readInfor();

    public void writeInfor();
}
```

Figure 31: Low Cohesion

Cohesion

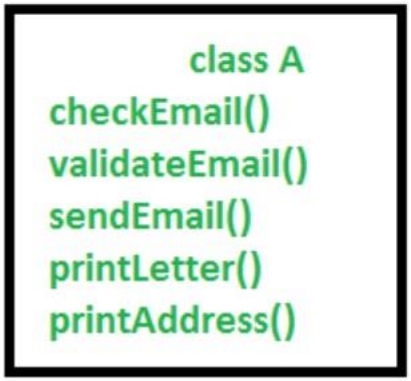

Fig: Low cohesion

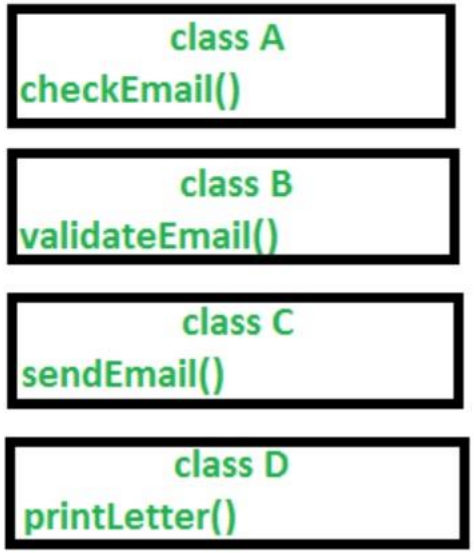

Fig: High cohesion

Figure 32: Cohesion

Coupling

- Low coupling can be achieved by having less classes linking to one another. The best way to reduce coupling is by providing an API (interface). OOP languages like C#, Java, C++ offer quite a few mechanisms to offer low coupling like public classes, public methods, interfaces and other interaction points between different classes and modules.

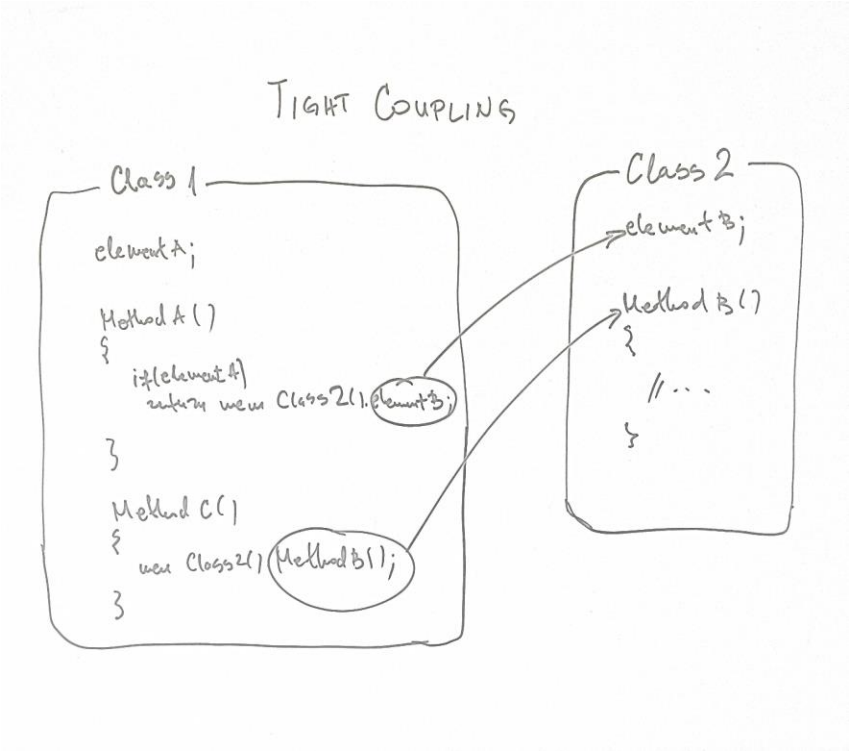

Figure 33:High or Tight Coupling

Thanks