Software Engineering

Zhang Shuang

zhangs@swc.neu.edu.cn

Chapter 4 Introduction to Object

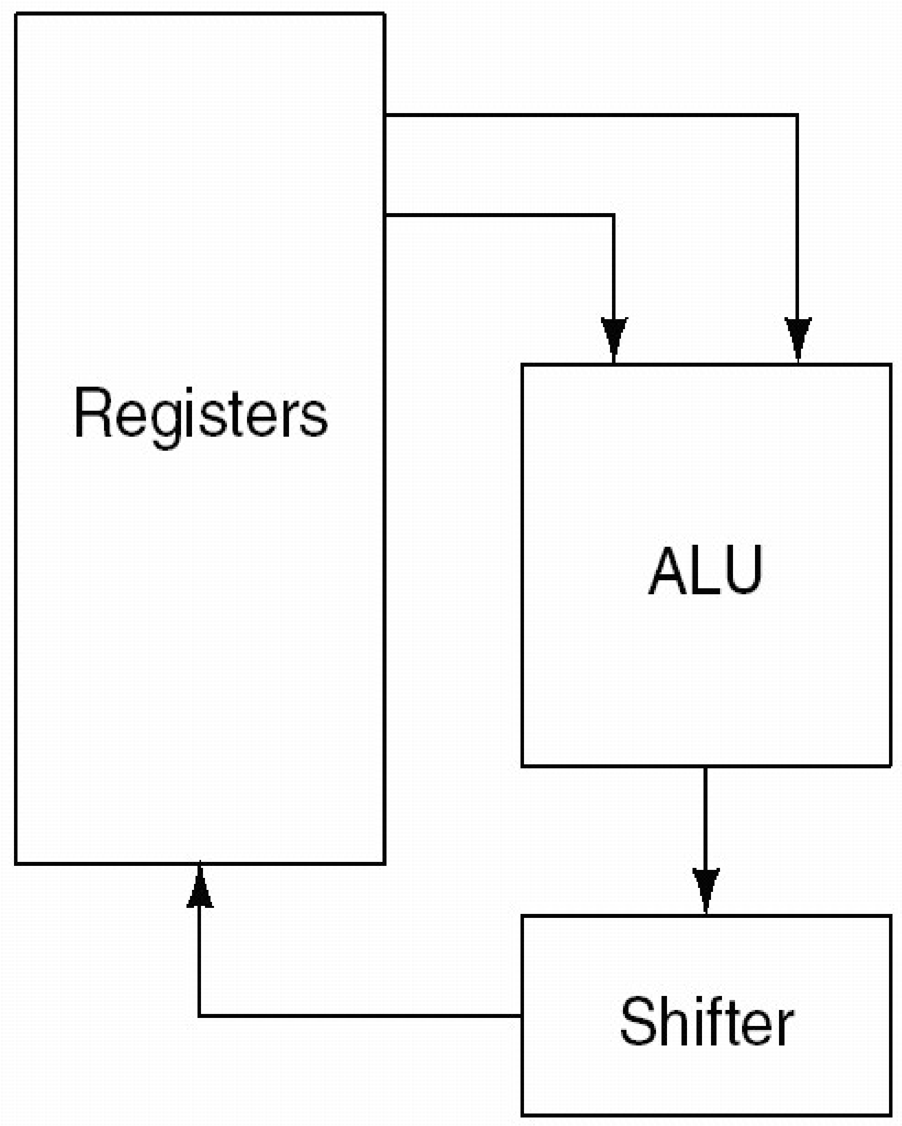
- What is a module?
- Cohesion
- Coupling
- Data Encapsulation
- Information Hiding
- Objects & OO Paradigm
- UML

4.1 What is a Module?

- **What is a module?**
  - **A lexically contiguous (词法相邻) sequence of program statements, bounded by boundary elements (边界元素, e.g. {…} in Java or C++), with an aggregate identifier (聚合标识符, e.g. *class* in Java or *function* in C or C++).**
  - **A *class* is a module; a *function* is also a module.**

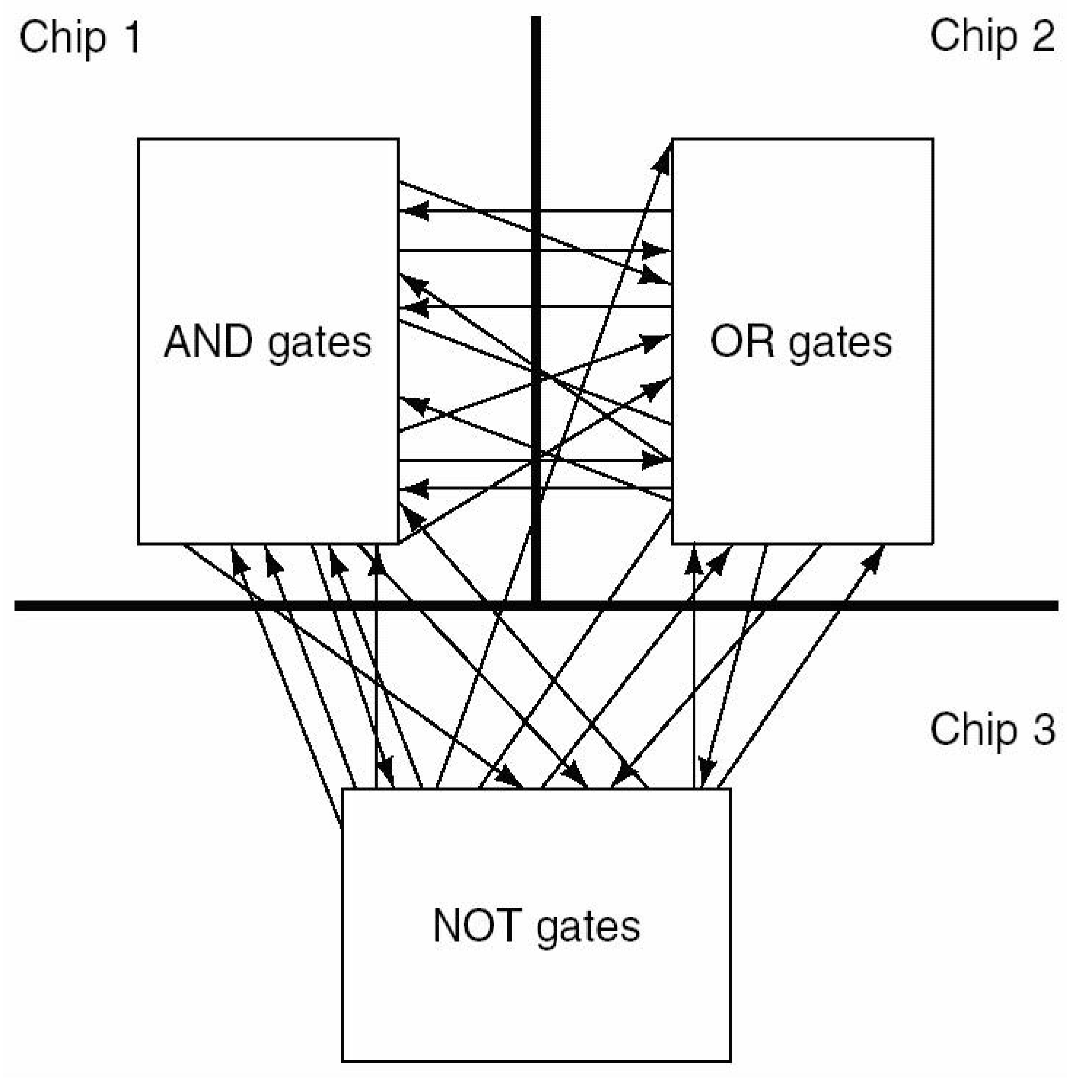
Design of Computer

- A highly incompetent computer architect decides to build an ALU, shifter and 16 registers with AND, OR, and NOT gates, rather than NAND or NOR gates.



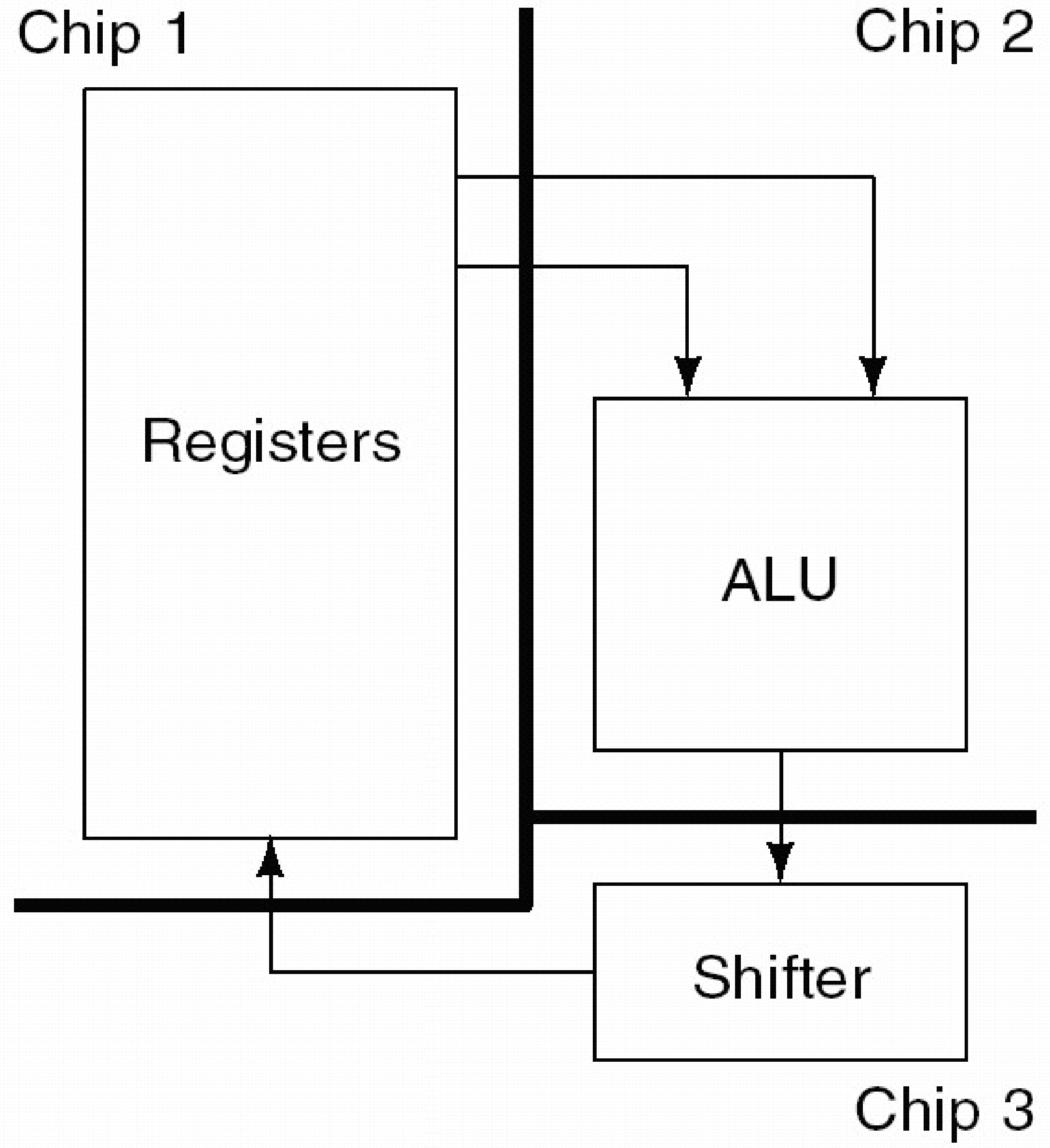
Design of Computer (contd)

- Redesign with one gate type per chip
- Resulting "masterpiece"



Design of Computer (contd)

- Architect designs 3 silicon chips



Computer Design (contd)

- The two designs are functionally equivalent
  - Second design is
    - Hard to understand
    - Hard to locate faults
    - Difficult to extend or enhance
    - Cannot be reused in another product
- Modules must be like the first design
  - Maximal relationships within modules, minimal relationships between modules

Composite / Structured Design

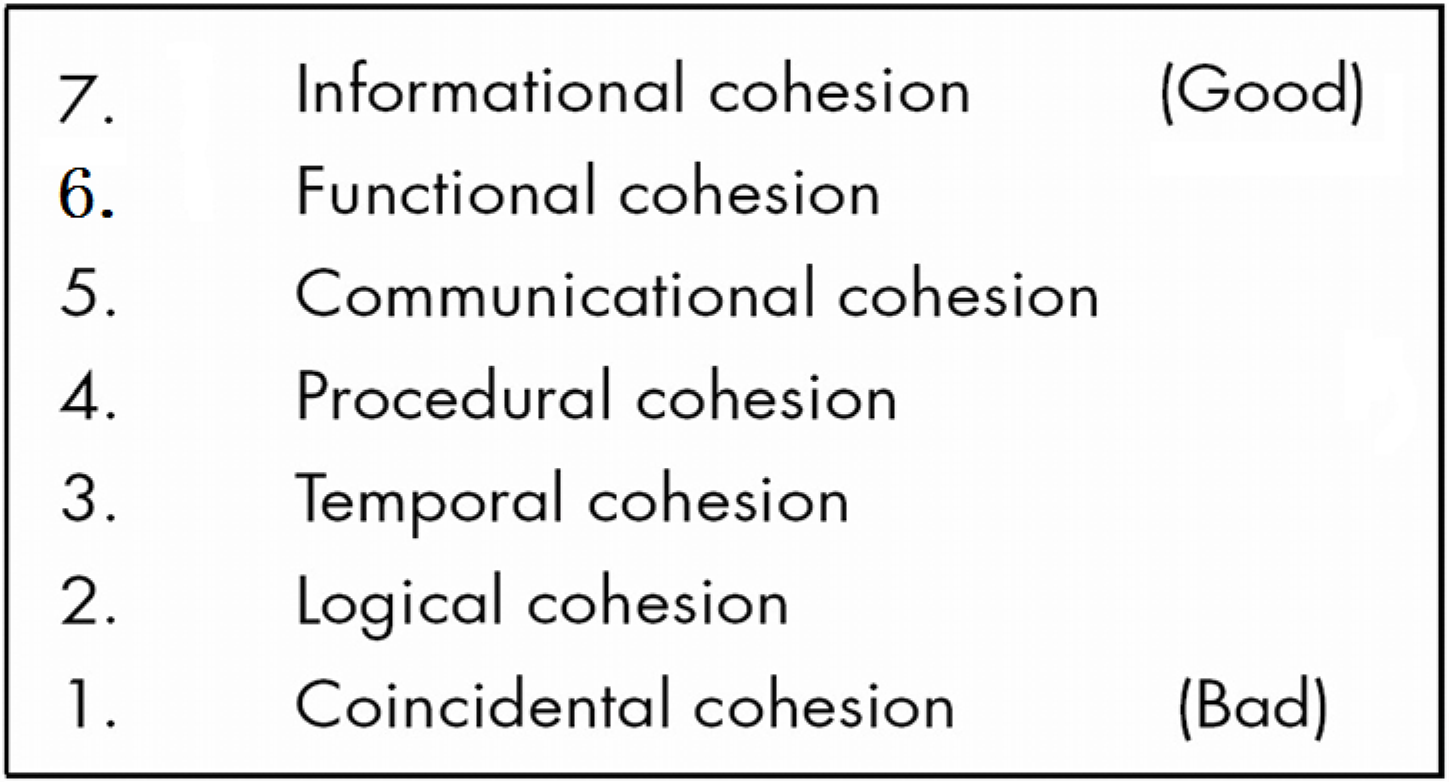
- Method for breaking up a product into modules for
  - maximal interaction within module, and
  - minimal interaction between modules
- Module cohesion
  - Degree of interaction within a module
- Module coupling
  - Degree of interaction between modules

Function, Logic, and Context of module

- In C/SD, the name of a module is its *function*
- Example
  - Module computes square root of double precision integers using Newton’s algorithm. Module is named *computeSquareRoot*

4.2 Cohesion

- Degree of interaction within a module
- Seven categories or levels of cohesion (non-linear scale)

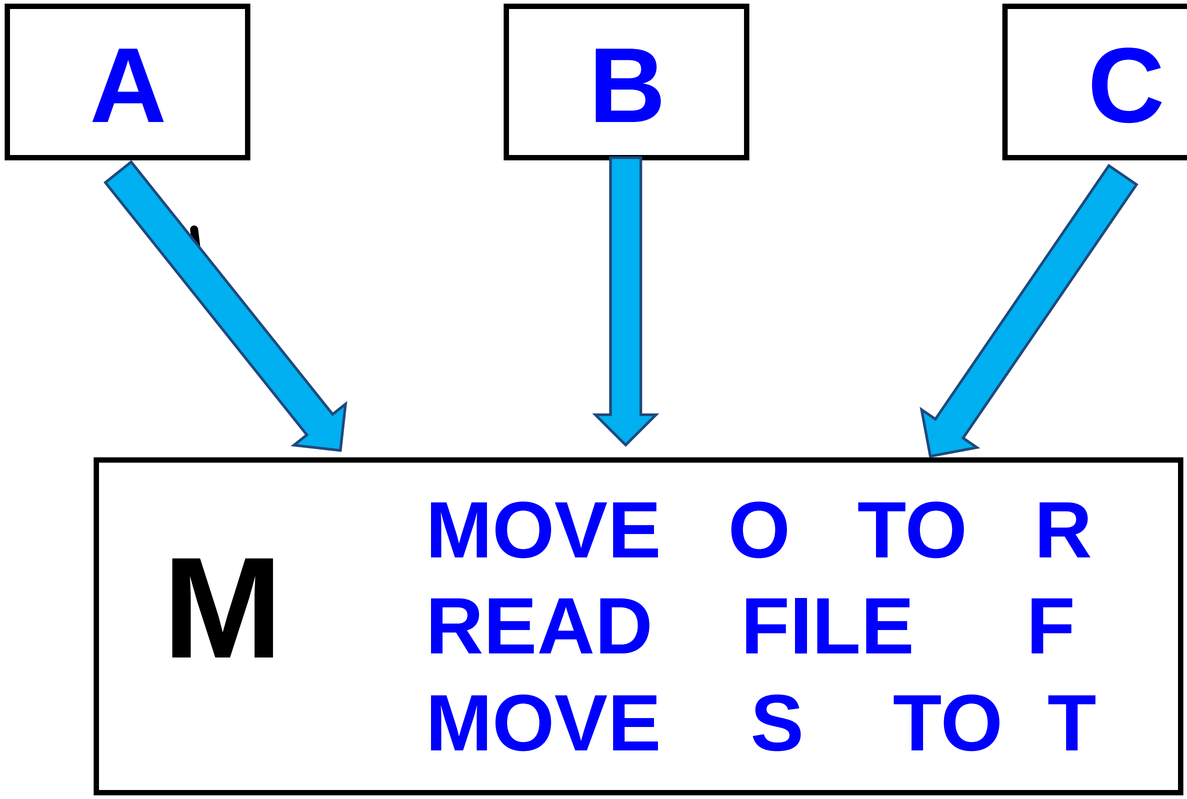


7.	Informational cohesion	(Good)
6.	Functional cohesion	
5.	Communicational cohesion	
4.	Procedural cohesion	
3.	Temporal cohesion	
2.	Logical cohesion	
1.	Coincidental cohesion	(Bad)

1. Coincidental Cohesion

- A module has coincidental cohesion if it performs multiple, completely unrelated actions
- Arise from rules like——"Every module will consist of between 35 and 50 statements"
- Example

1. Coincidental Cohesion

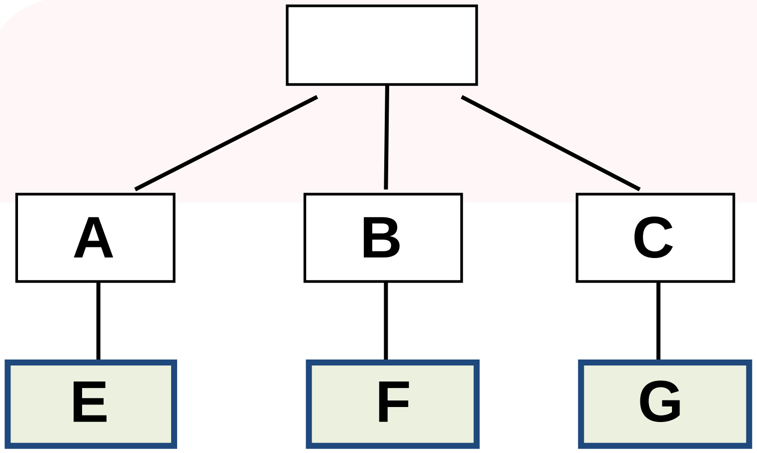


Why Is Coincidental Cohesion So Bad?

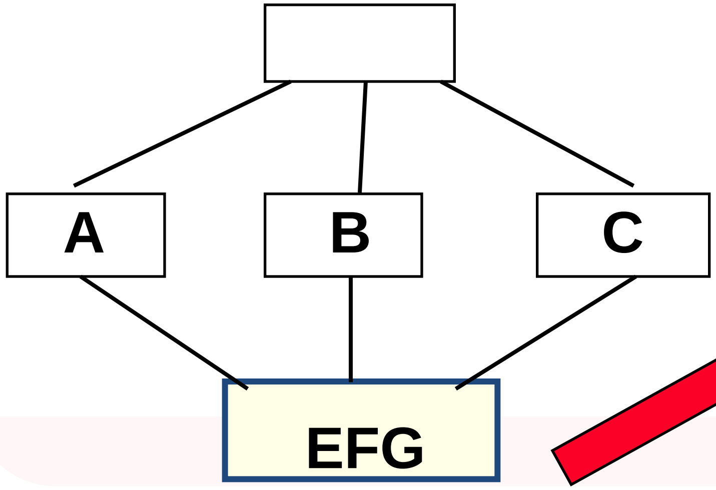
- Degrades maintainability
- Modules are not reusable
- This is easy to fix
  - Break into separate modules each performing one task

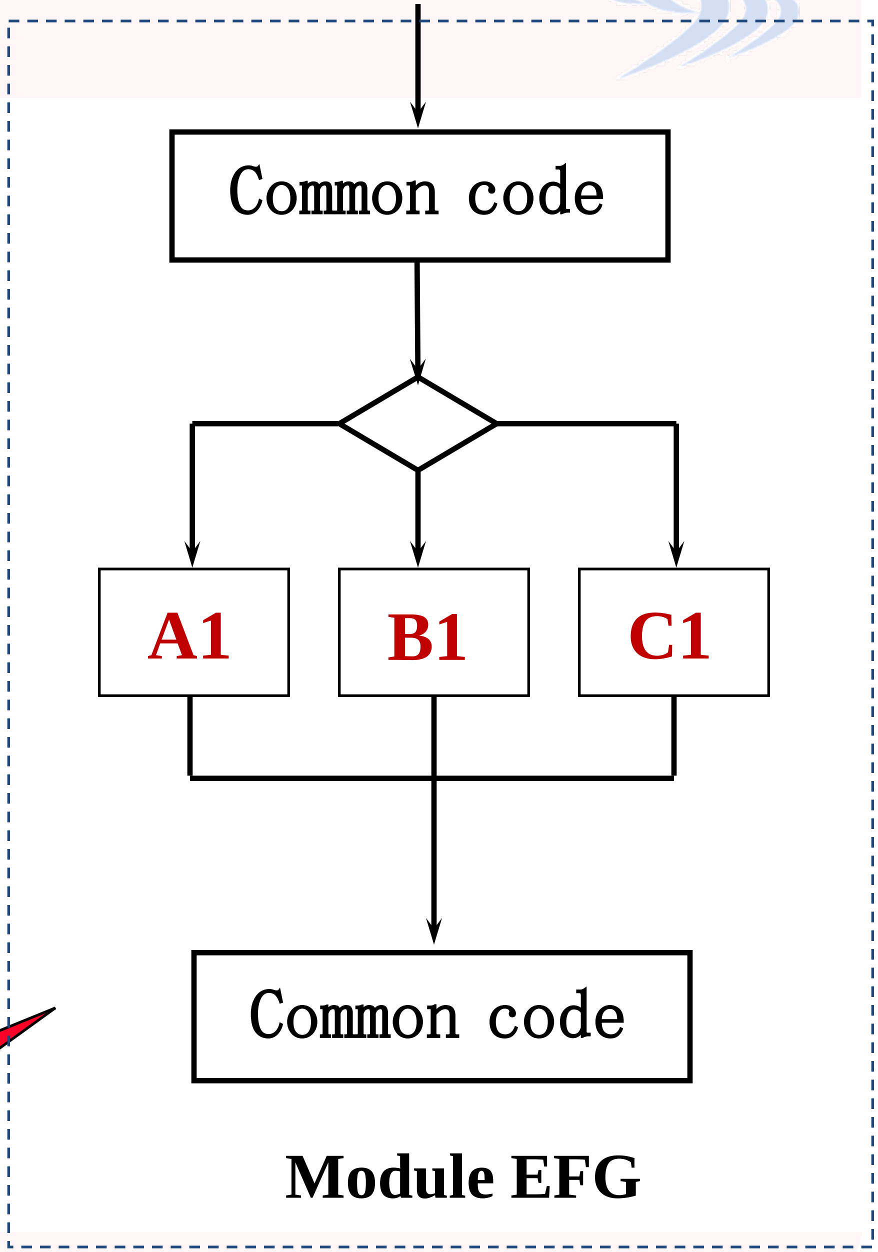
2. Logical Cohesion

- A module has logical cohesion when it performs a series of related actions, one of which is selected by the calling module
- Example



E,F and G are logically similar, they are composed into a new module EFG





Why Is Logical Cohesion So Bad?

- The interface is difficult to understand
- Difficult to modify
- Code for more than one action may be intertwined
- Increase coupling
- Low efficiency

3. Temporal Cohesion

- **A module has temporal cohesion when it performs a series of actions related in time**
- **The actions in the module must execute in the same time**
- **Example:**
  - ***Initialization*** **module**
  - ***ErrorHandling*** **module**
  - ***SystemTermination*** **module**

Why Is Temporal Cohesion So Bad?

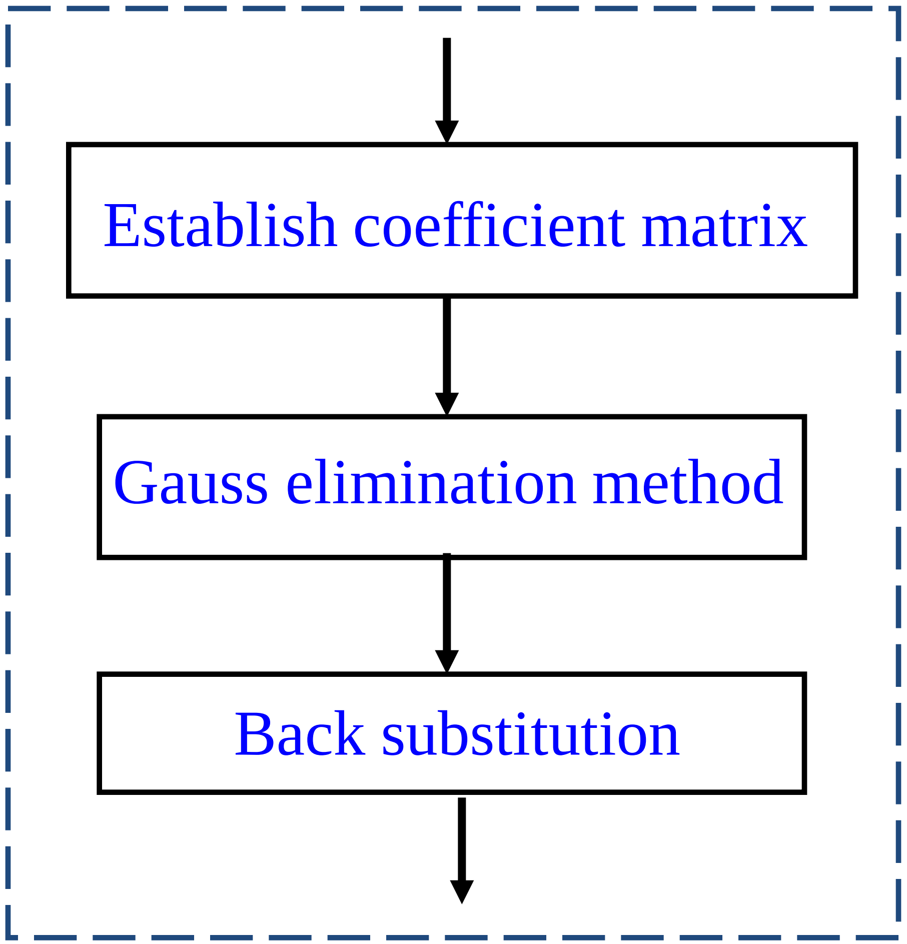
- Actions of this module are weakly related to one another, but strongly related to actions in other modules.
  - Consider sales district table

- Not reusable

4. Procedural Cohesion

- A module has procedural cohesion if it performs a series of actions related by the procedure to be followed by the product
- Example

Procedural Cohesion



Gauss elimination algorithm

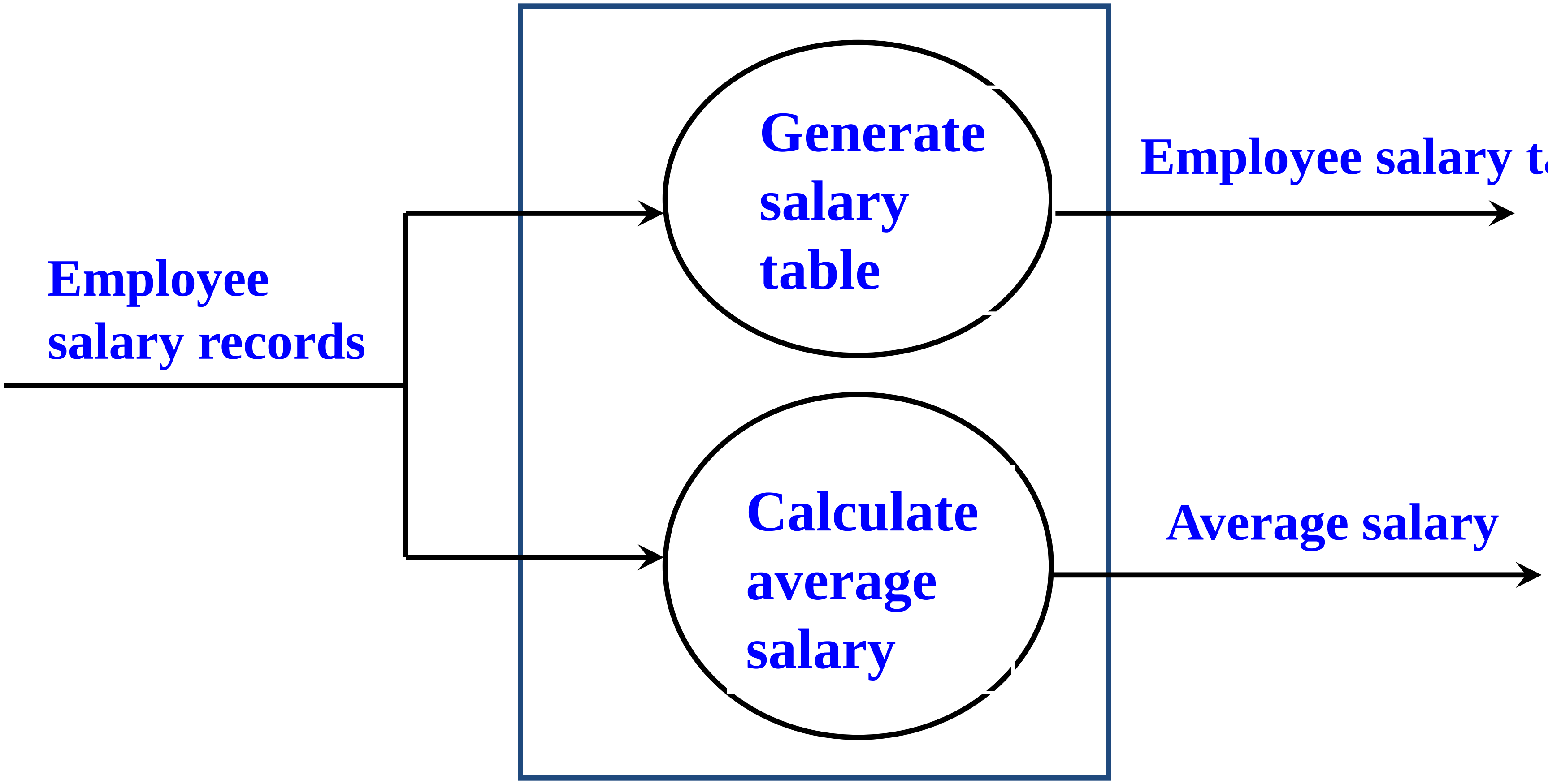
Why Is Procedural Cohesion So Bad?

- Actions are still weakly connected, so module is not reusable

5. Communicational Cohesion

- A module has communicational cohesion if it performs a series of actions related by the procedure to be followed by the product, but in addition all the actions operate on the same input or output data
- Example

Example of communicational cohesion



Module of generating salary table and calculating average salary

Why Is Communicational Cohesion So Bad?

- **Still lack of reusability**

6. Functional Cohesion

- Module with functional cohesion performs exactly one action

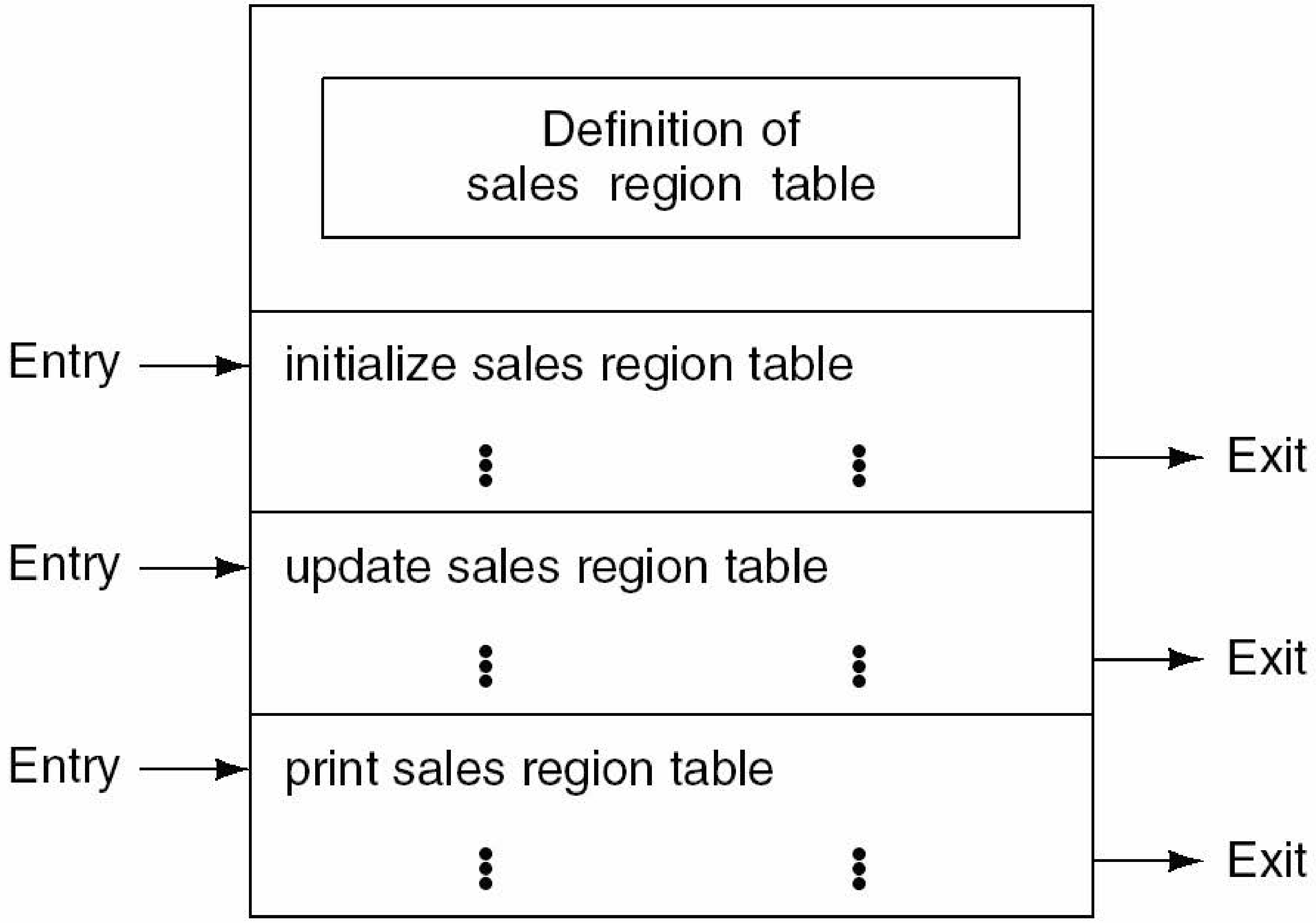
Why is functional cohesion so good?

- More reusable
- Corrective maintenance easier
    - Fault isolation
    - Fewer regression faults
- Easier to extend product

7. Informational Cohesion

- A module has informational cohesion if it performs a number of actions, each with its own entry point, with independent code for each action, all performed on the same data structure

Why Is Informational Cohesion So Good?


Definition of sales region table
Entry
initialize sales region table
Exit
Entry
update sales region table
Exit
Entry
print sales region table
Exit

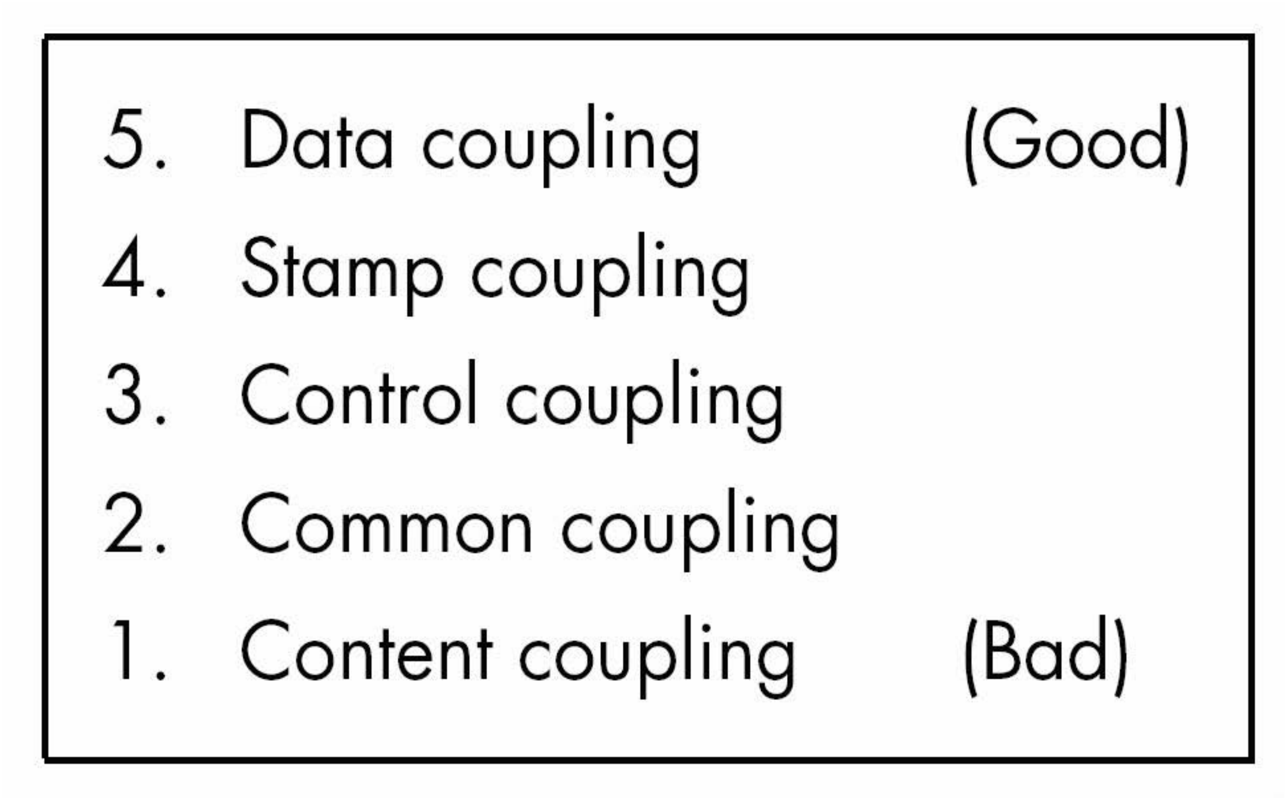
- **Essentially, this is an abstract data type (see later)**

4.3 Coupling

- **Coupling ---- Degree of interaction between two modules.**

4.3 Coupling

- **Five categories or levels of coupling (non-linear scale):**



5.	Data coupling	(Good)
4.	Stamp coupling	
3.	Control coupling	
2.	Common coupling	
1.	Content coupling	(Bad)

1. Content Coupling

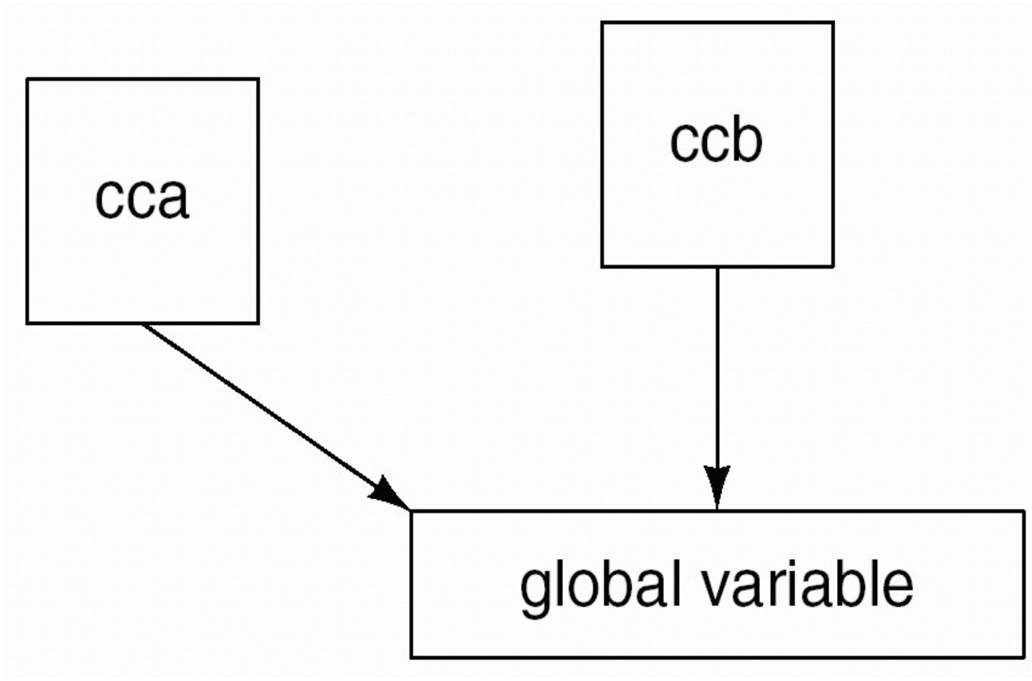
- Two modules are content coupled if one directly references contents of the other.

1. Content Coupling

```
public class Product {
        public float unitPrice;
        private float unitPrice
        …
        setUnitPrice(float pUnitPrice){unitPrice=pUnitPrice;}
}
public class Order  {
        private Product myProduct=new Product();
        public void setItem() {
                myProduct.unitPrice = -100;
                myProduct.setUnitPrice(100);
        }
}
```

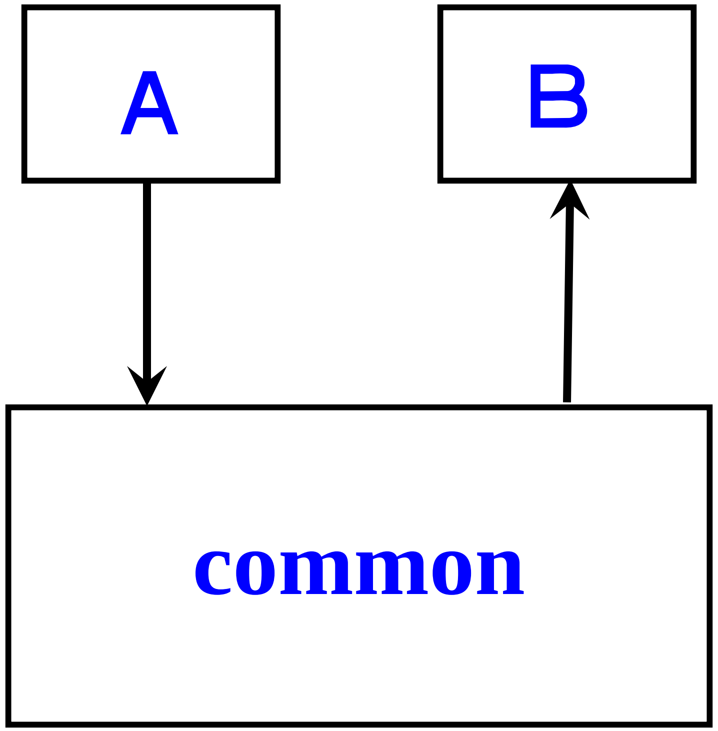
2. Common Coupling

- Two modules are common coupled if they have write access to global data

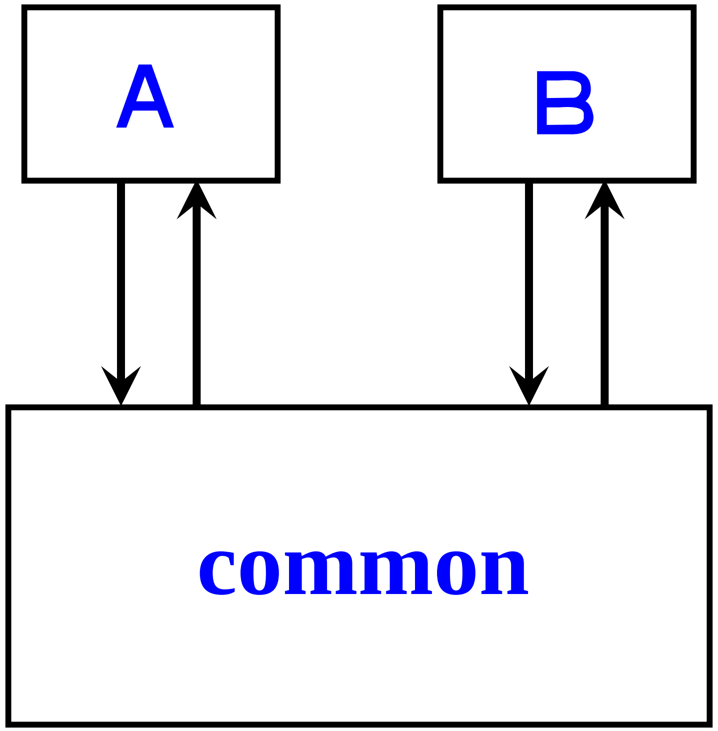


- Example 1
  - Modules *cca* and *ccb* can access and change value of global variable

Common Coupling Examples

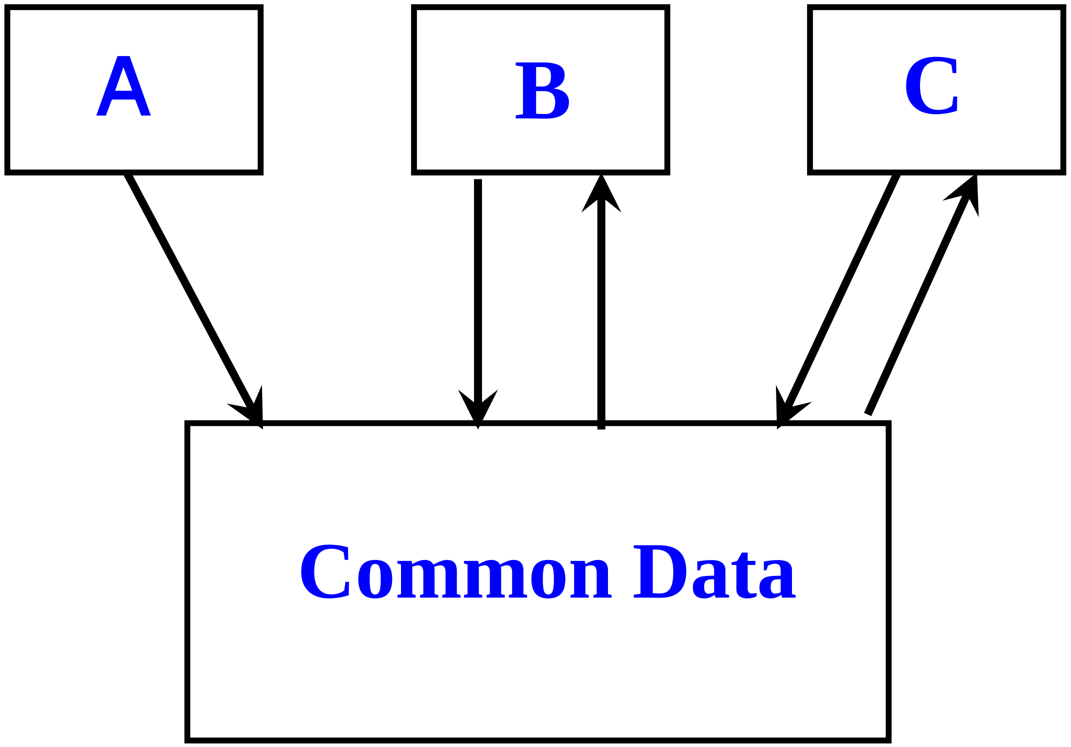


Loose coupling

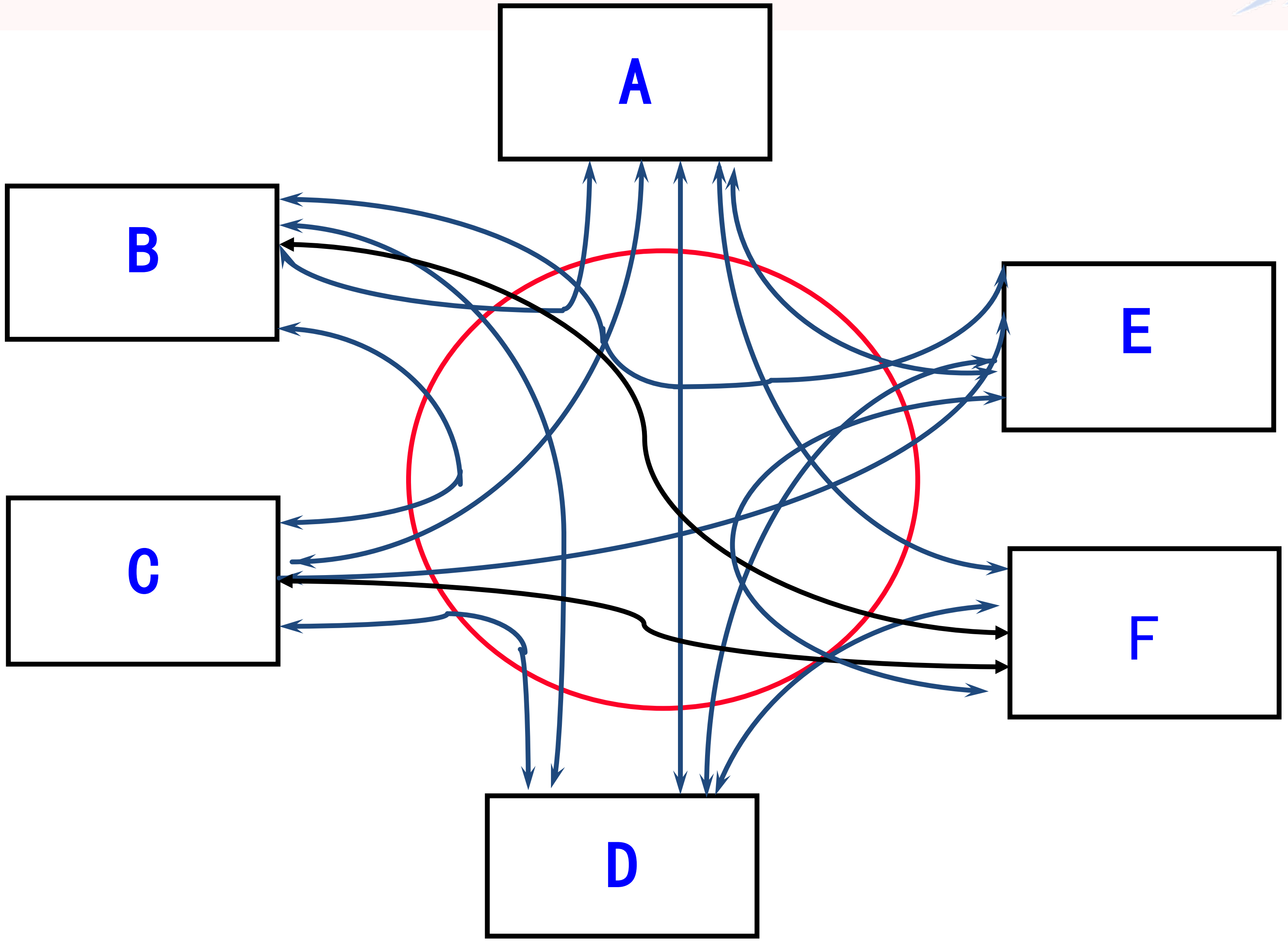


Close coupling

Common Coupling Examples

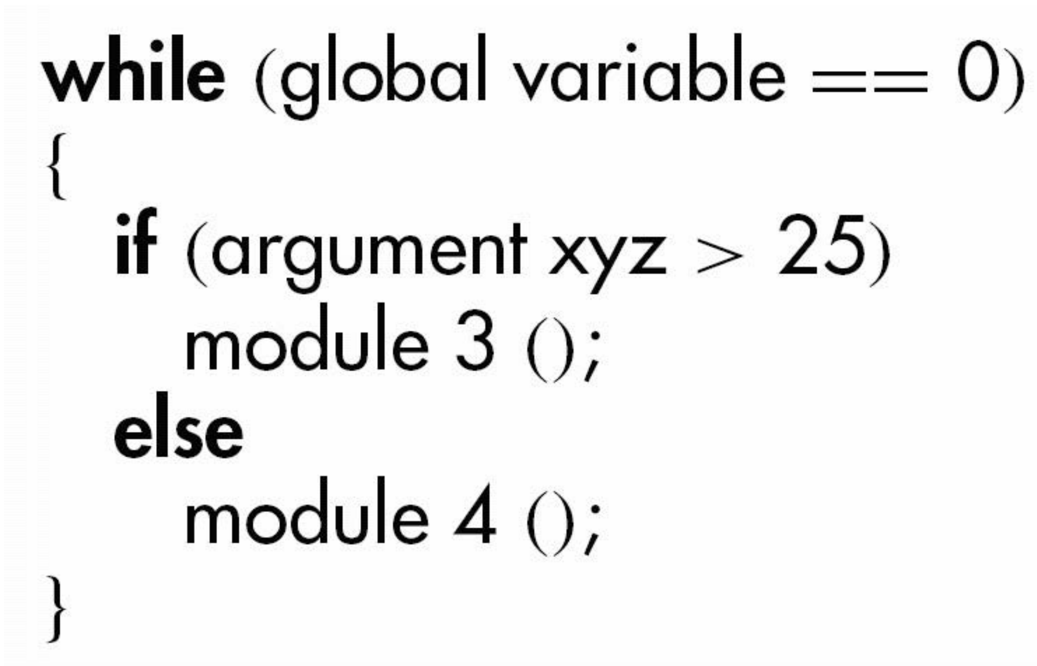


Common Coupling Examples



Why Is Common Coupling So Bad?

- Contradicts the spirit of structured programming
  - The resulting code is virtually unreadable



```
while (global variable == 0)
{
    if (argument xyz > 25)
        module 3 ();
    else
        module 4 ();
}
```

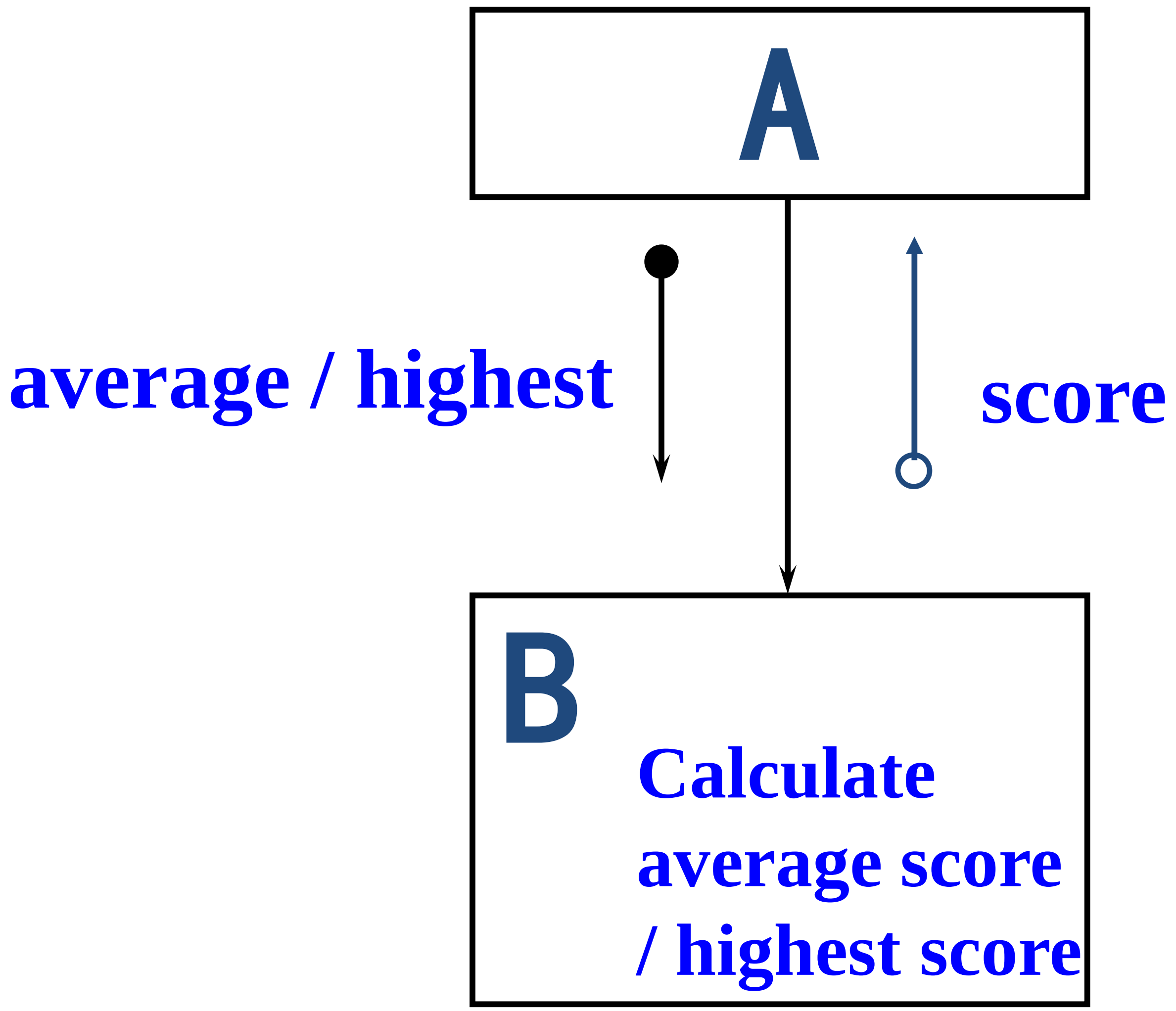
Why Is Common Coupling So Bad?

- Modules can have side-effects
  - This affects their readability
- Entire module must be read to find out what it does
- Difficult to reuse
- Module exposed to more data than necessary

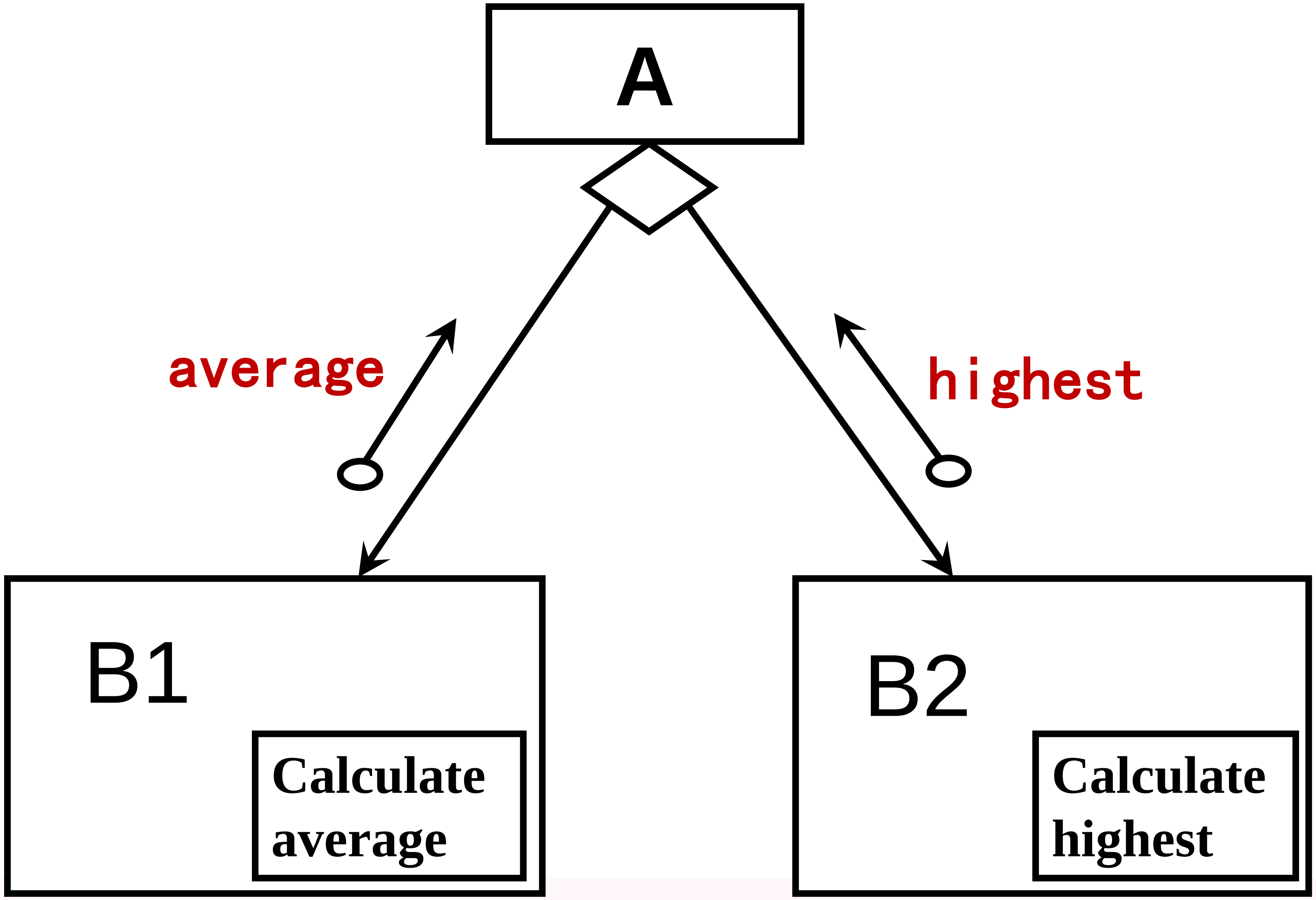
3. Control Coupling

- Two modules are control coupled if one passes an element of control to the other
- Example 1
  - Operation code passed to module with logical cohesion
- Example 2
  - Control-switch passed as argument

Control coupling example



Control coupling example



Why Is Control Coupling So Bad?

- **Modules are not independent; module *b* (the called module) must know internal structure and logic of module *a*.**
  - **Affects reusability**
- **Associated with modules of logical cohesion**

4. Stamp Coupling

- **Some languages allow only simple variables as parameters**
  - ***part number***
  - ***satellite altitude***
  - ***temprature***
- **Many languages also support passing of data structures**
  - ***part record***
  - ***satellite coordinates***
  - ***segment table***

4. Stamp Coupling (contd)

- Two modules are stamp coupled if a data structure is passed as a parameter, but the called module operates on some but not all of the individual components of the data structure.

Why Is Stamp Coupling So Bad?

```
public class Order    {
    public float calcTotalMoney(User user)    {
        int userLevel = user.getLevel();
        int userConsumeScore= user.getConsumeScore();
        …//the following will compute the total cost of order
        …
    }
    …
}
```

Why Is Stamp Coupling So Bad?

```
public class Order   {
    public float calcTotalMoney
                        (int userLevel, int userConsumeScore)   {
        …//the following will compute the total cost of order
        …
    }
    …
}
```

Why Is Stamp Coupling So Bad?

- It is not clear, without reading the entire module, which fields of a record are accessed or changed
- Difficult to understand
- Unlikely to be reusable

Why Is Stamp Coupling So Bad?

- **More data than necessary is passed**
  - **Uncontrolled data access can lead to computer crime**
- **There is nothing wrong with passing a data structure as a parameter, provided all the components of the data structure are accessed and/or changed**

  invert matrix (original matrix, inverted matrix);

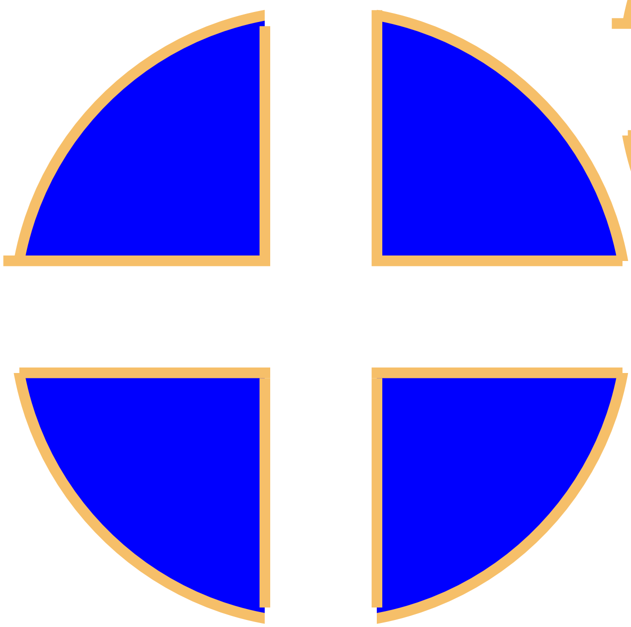
  print inventory record (warehouse record);

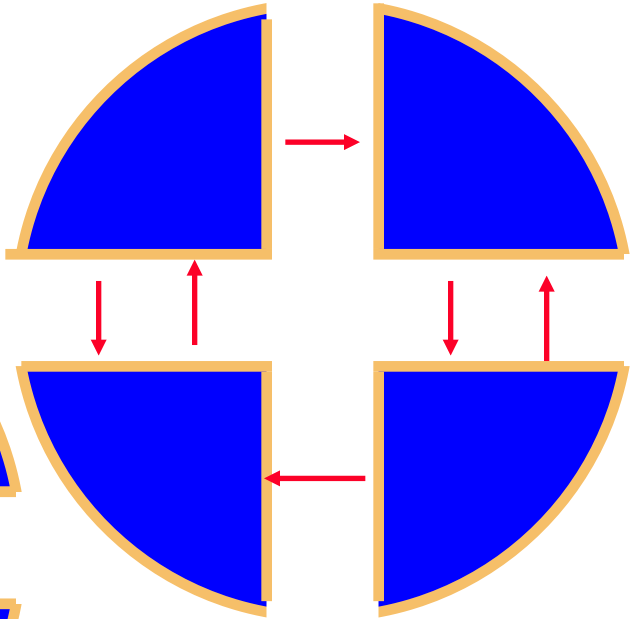
5. Data Coupling

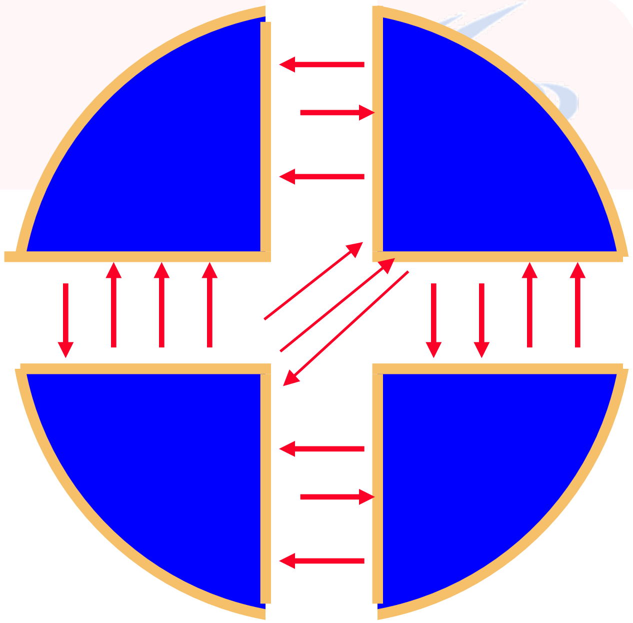
- **Two modules are data coupled if all parameters are homogeneous data items, simple parameters, or data structures all of whose elements are used by called module.**
- **Examples**
  - ***display time of arrival (flight number);***
  - ***compute product (first number, second number);***
  - ***get job with highest priority (job queue);***

Why Is Data Coupling So Good?

- The difficulties of content, common, control, and stamp coupling are not present
- Maintenance is easier


No coupling


Loosely coupling


Tightly coupling

Good design has

high cohesion

&

low coupling

Thank You !