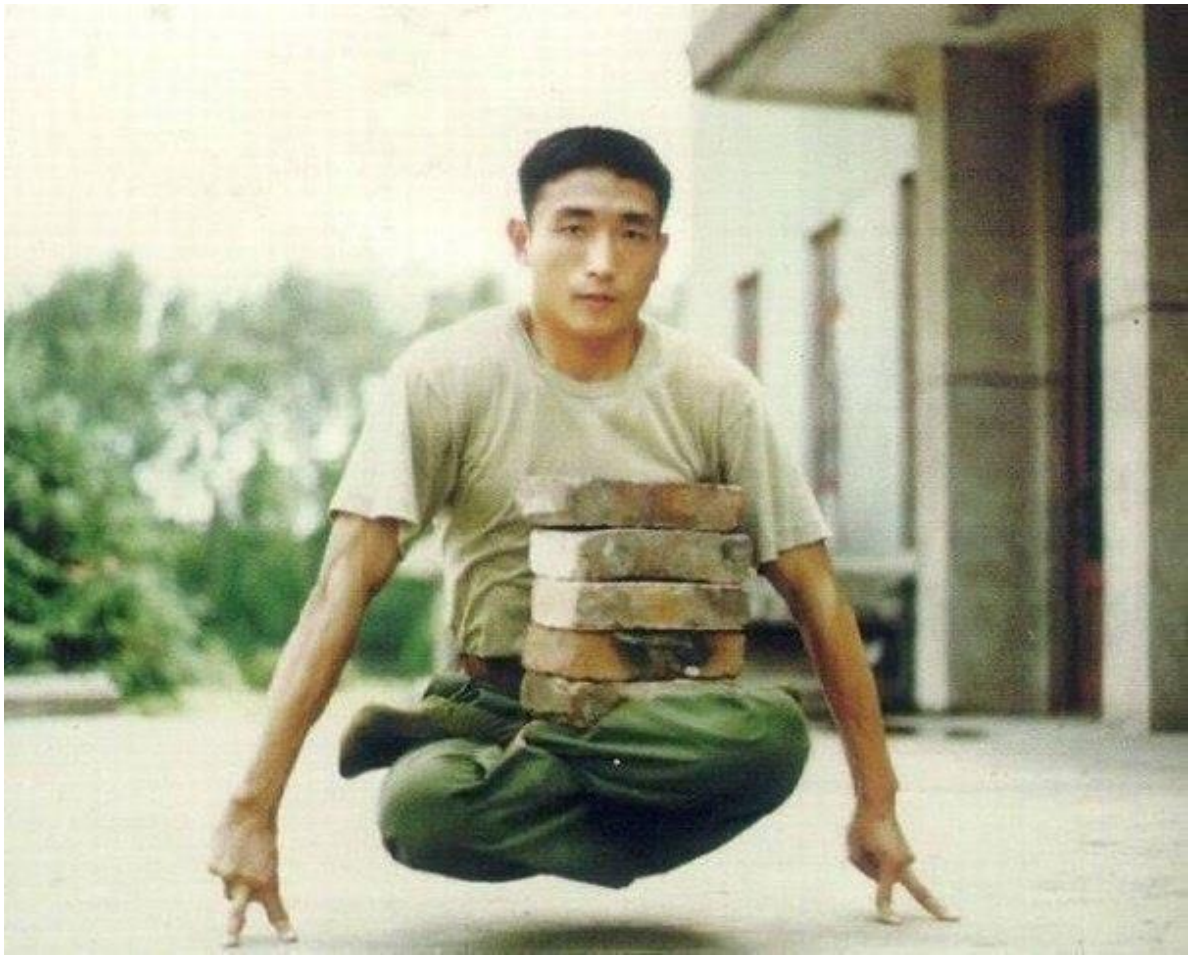
Good coding practice in real life

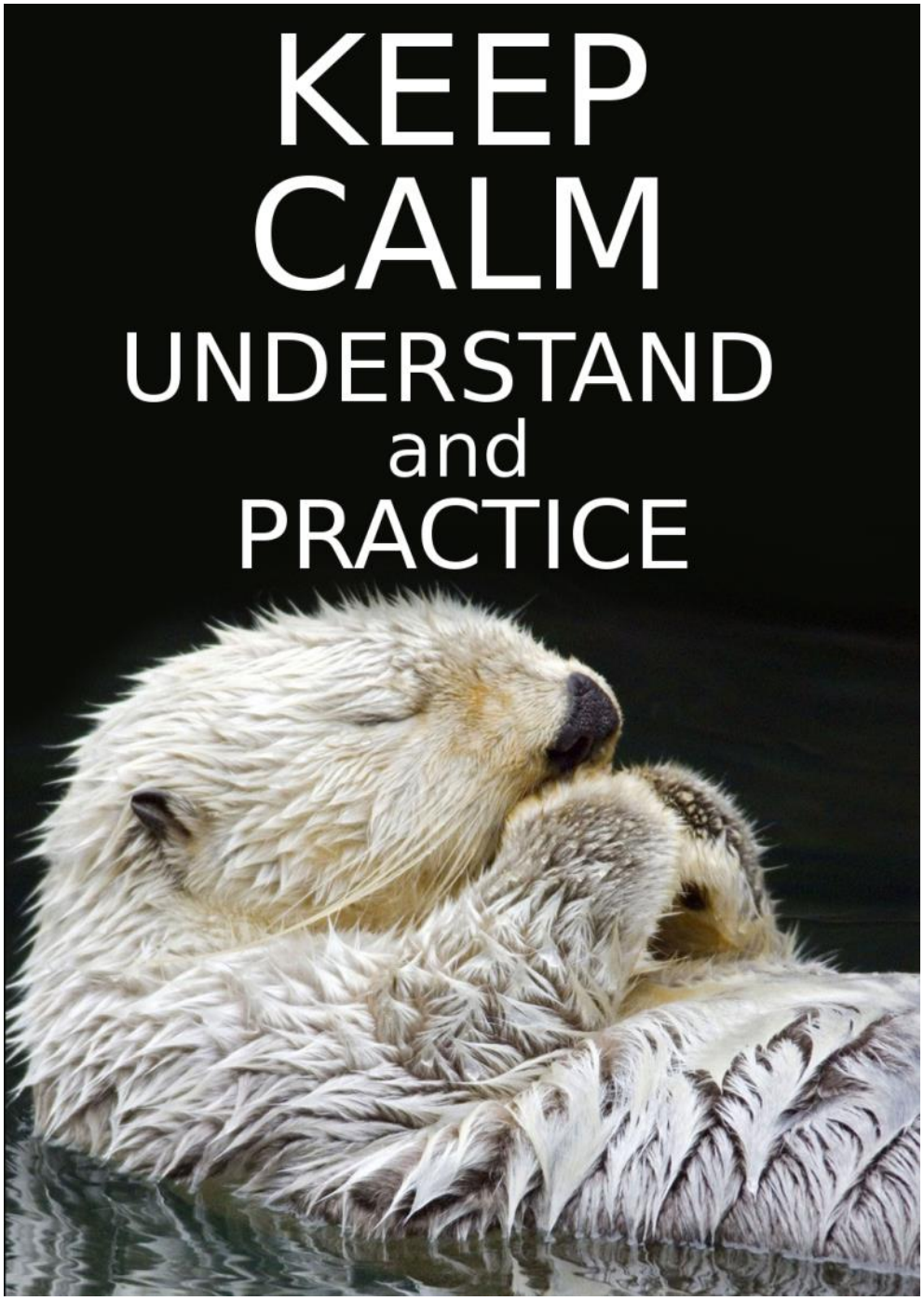
(with a focus on cohesion and coupling)

Vladimir Alekseichenko

Good practices make life easier

Good practices are not easy




KEEP
CALM
UNDERSTAND
and
PRACTICE

Agenda

- Coupling
  - Factors affecting coupling
  - Types of coupling
  - The types of module coupling
- Cohesion
  - The types of module cohesion
  - Determining module cohesion
- Relationship between coupling and cohesion

Cohesion and coupling

Three specific things with code structure:

- Keep things that have to change together as close together in the code as possible.
- Allow unrelated things in the code to change independently (also known as orthogonality).
- Minimize duplication in the code.

http://msdn.microsoft.com/en-us/magazine/cc947917.aspx

GRASP (object-oriented design)

General Responsibility Assignment Software Patterns (or Principles), abbreviated GRASP, consists of guidelines for assigning responsibility to classes and objects in object-oriented design.

http://en.wikipedia.org/wiki/GRASP_(object-oriented_design)

GRASP (object-oriented design)

- Controller
- Creator
- High Cohesion
- Indirection
- Information Expert
- Low Coupling
- Polymorphism
- Protected Variations
- Pure Fabrication

Highly coupled

http://www.designboom.com/art/thomas-kalak-bangkok-cable-ways/

Loosely coupled

http://wikimapia.org/24805148

Low cohesion

High cohesion

http://spb-otdam-darom.livejournal.com/8312881.html

Low cohesion

http://persbaglio.ilcannocchiale.it/?TAG=disordine

High cohesion

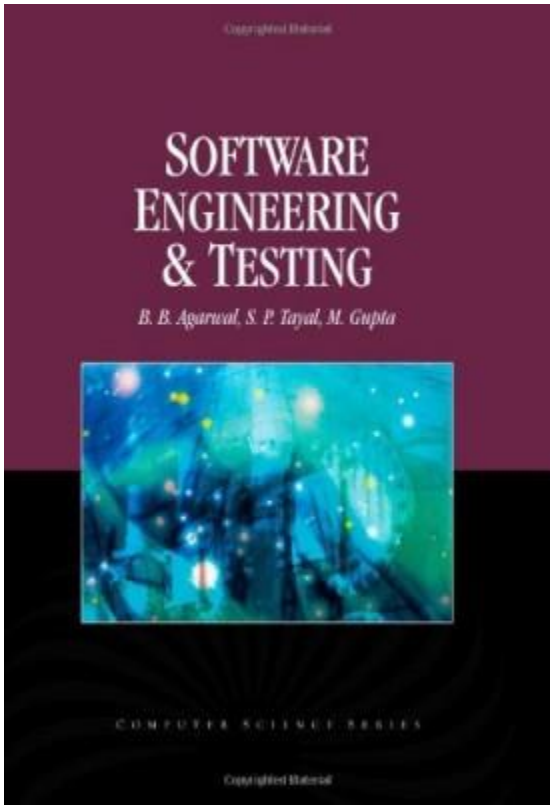
http://www.3deko.info/inter/ofor/415-uyutnyj-balkon-mesto-otdyxa-chast-ii.html

Low cohesion

High cohesion

http://renatar.livejournal.com/435236.html?thread=25823780

Software Engineering and Testing: An Introduction (Computer Science)



http://www.ebooks-share.net/software-engineering-and-testing-an-introduction-computer-science/

Coupling

- The coupling between two modules indicates the degree of interdependence between them.
- If two modules interchange a large amount of data, then they are highly interdependent.

The key question

How much of one module must be known in order to understand another module?

Highly coupled

When the modules are highly dependent on each other.

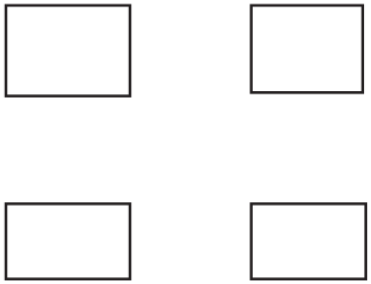
Loosely coupled

When the modules are dependent on each other but the interconnection among them is weak.

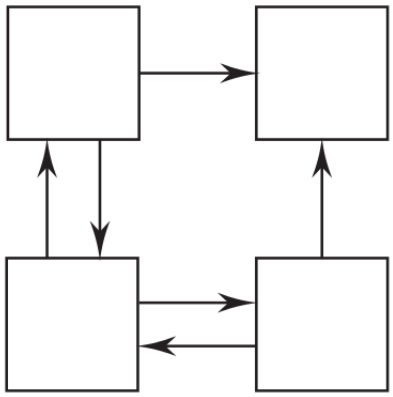
Uncoupled

When the different modules have no interconnection among them.

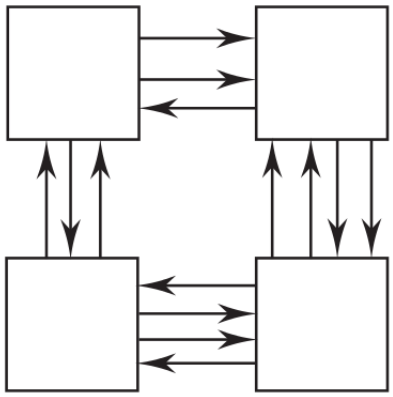
Coupling



UNCOUPLED MODULES:
NO DEPENDENCIES



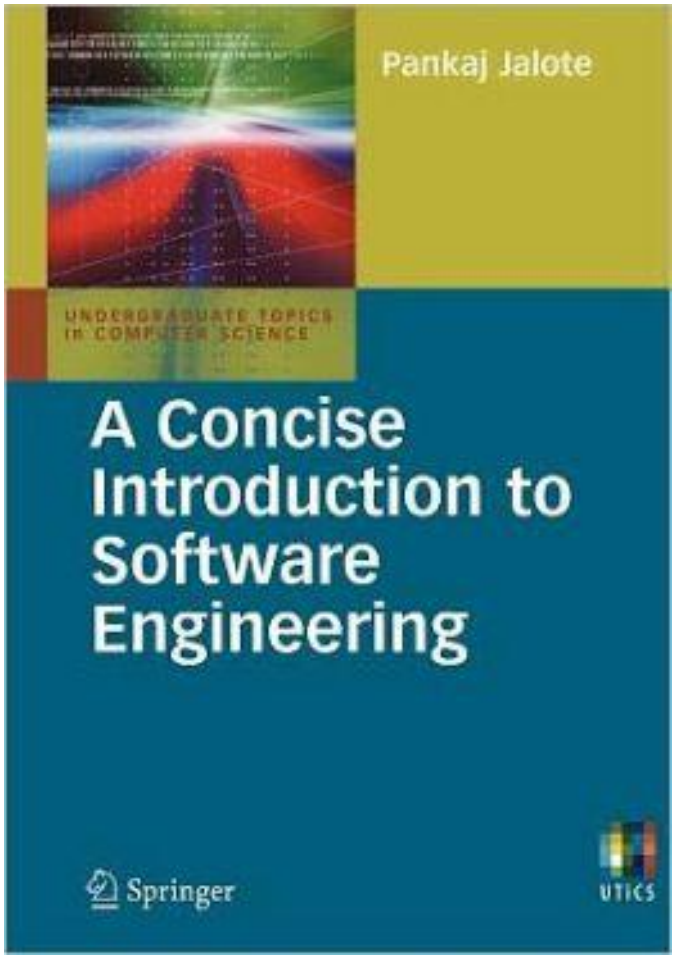
LOOSELY COUPLED:
SOME DEPENDENCIES



HIGHLY COUPLED:
MANY DEPENDENCIES

Software Engineering and Testing: An Introduction (Computer Science) by *B.B. Agarwal, M. Gupta, S.P. Tayal*

A Concise Introduction to Software Engineering



http://www.barnesandnoble.com/w/concise-introduction-to-software-engineering-pankaj-jalote/1101677570?ean=9781848822108

Coupling

Coupling between modules is the strength of interconnections between modules or a measure of interdependence among modules.

Coupling

In general, the more we must know about module A in order to understand module B,

the more closely connected A is to B.

Factors affecting coupling (1)

- Coupling would increase if a module is used by other modules via an indirect and obscure interface, like directly using the internals of a module or using shared variables.

Factors affecting coupling (2)

- Complexity of the interface is another factor affecting coupling.
- The more complex each interface is, the higher will be the degree of coupling.

Factors affecting coupling (3)

- The type of information flow along the interfaces is the third major factor affecting coupling.

Type of communication

There are two kinds of information that can flow along an interface:

- data
- control

Factors affecting coupling (4)

	Interface complexity	Type of connection	Type of communication
Low	Simple obvious	To module by name	Data
			Control
High	Complicated obscure	To internal elements	Hybrid

Types of coupling

In OO systems, three different types of coupling exist between modules:

- Interaction coupling
- Component coupling
- Inheritance coupling

Interaction coupling

It occurs due to methods of a class invoking methods of other classes.

Interaction coupling

- Coupling is lower if only data is passed, but is higher if control information is passed since the invoked method impacts the execution sequence in the calling method.
- Also, coupling is higher if the amount of data being passed is increased.

Component coupling

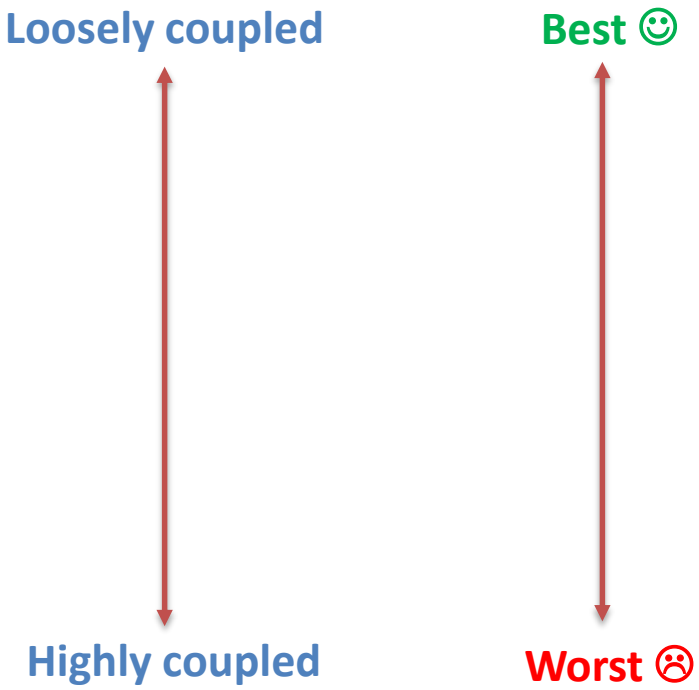
It refers to the interaction between two classes where a class has variables of the other class.

Inheritance coupling

It is due to the inheritance relationship between classes.

The types of module coupling

- Data coupling
- Stamp coupling
- Control coupling
- External coupling
- Common coupling
- Content coupling

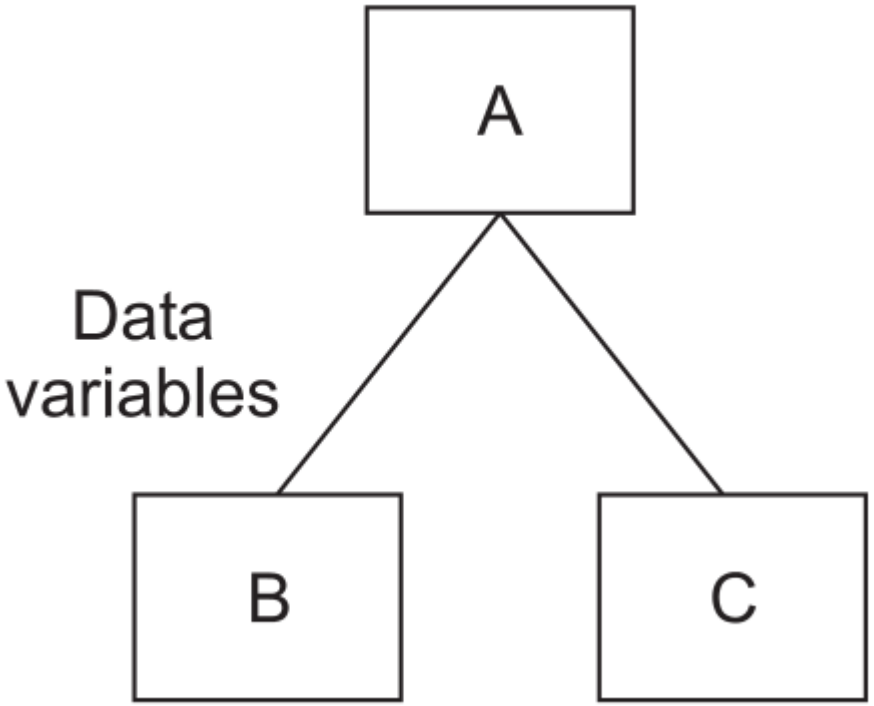


Software Engineering and Testing: An Introduction (Computer Science) by B.B. Agarwal, M. Gupta, S.P. Tayal

Data coupling

- Two modules are data coupled if they communicate using an elementary data item that is passed as a parameter between the two.
- When a non-global variable is passed to a module, modules are called data coupled.

Data coupling



Software Engineering and Testing: An Introduction (Computer Science) by B.B. Agarwal, M. Gupta, S.P. Tayal

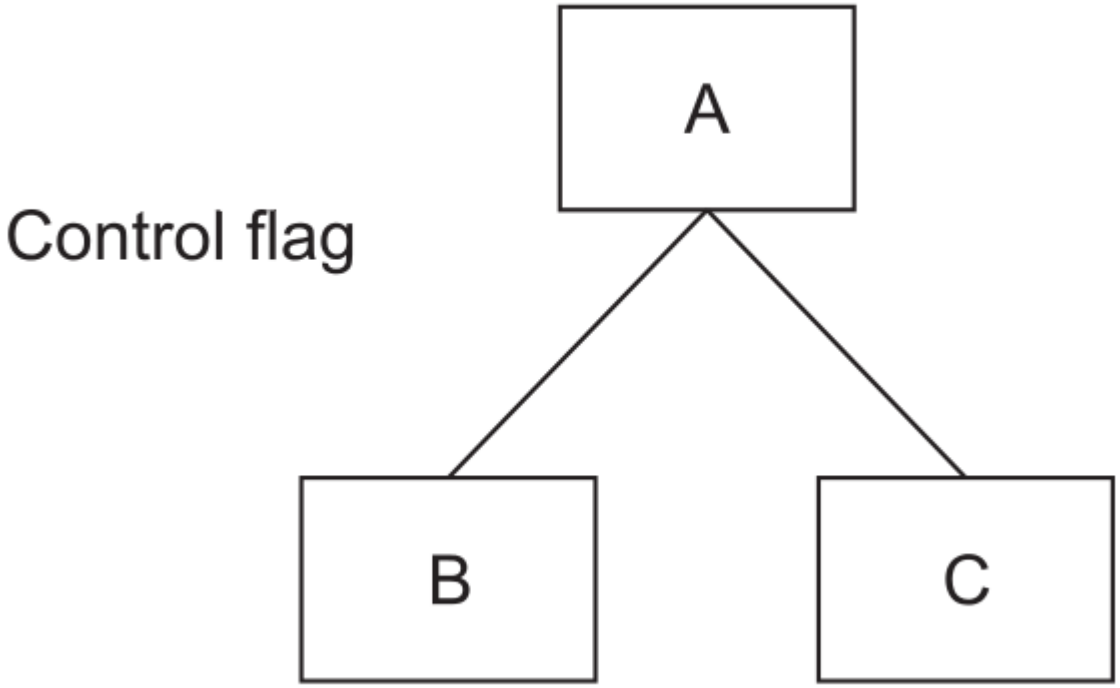
Stamp coupling

- Two modules are stamp coupled if they communicate using a composite data item, such as a record, structure, object, etc.
- When a module passes a non-global data structure or an entire structure to another module, they are said to be stamp coupled

Control coupling

- Control coupling exists between two modules if data from one module is used to direct the order of instruction execution in another.
- An example of control coupling is a flag set in one module that is tested in another module.

Control coupling



Software Engineering and Testing: An Introduction (Computer Science) by B.B. Agarwal, M. Gupta, S.P. Tayal

External coupling

- It occurs when modules are tied to an environment external to software.
- External coupling is essential but should be limited to a small number of modules with structures.

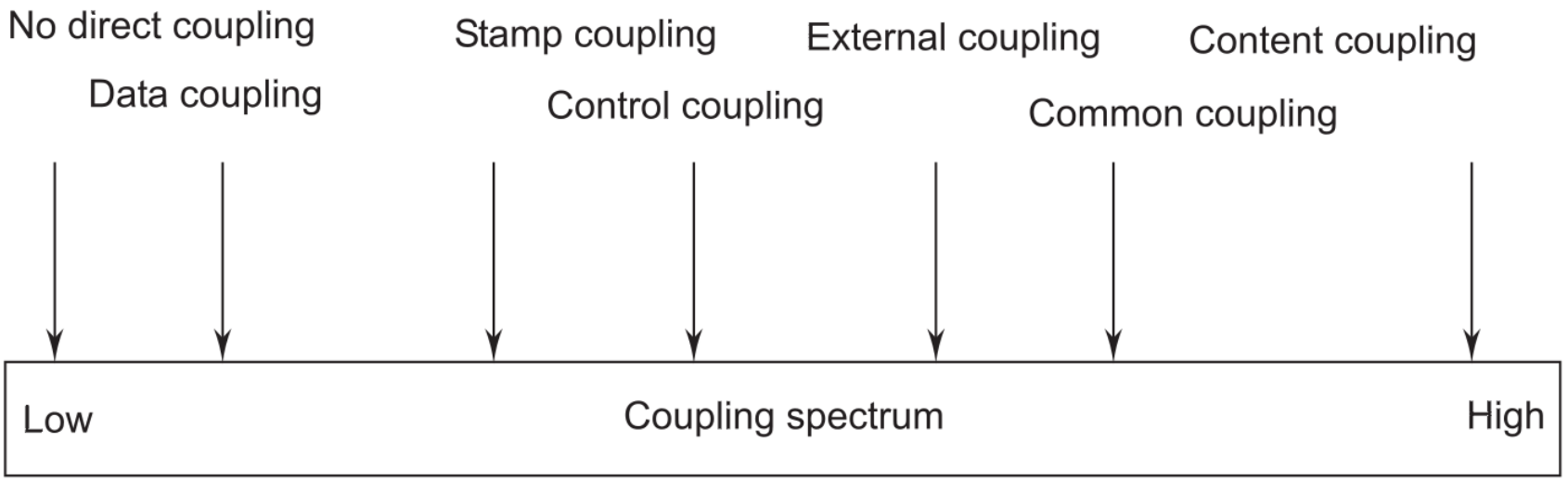
Common coupling

- Two modules are common coupled if they share some global data items (e.g., Global variables).

Content coupling

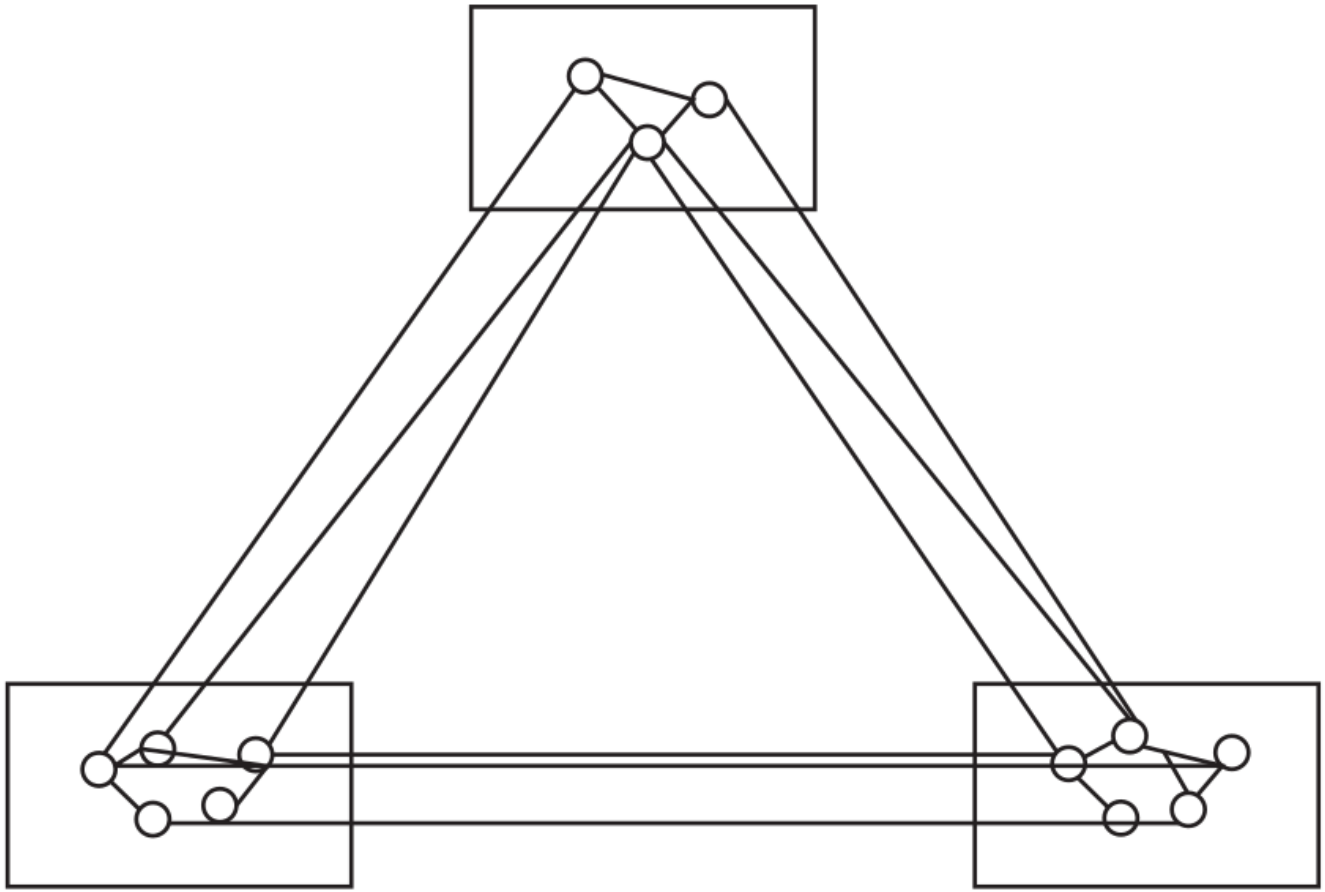
- Content coupling exists between two modules if their code is shared
- It is when one module directly refers to the inner workings of another module.
- Modules are highly interdependent on each other.

Coupling



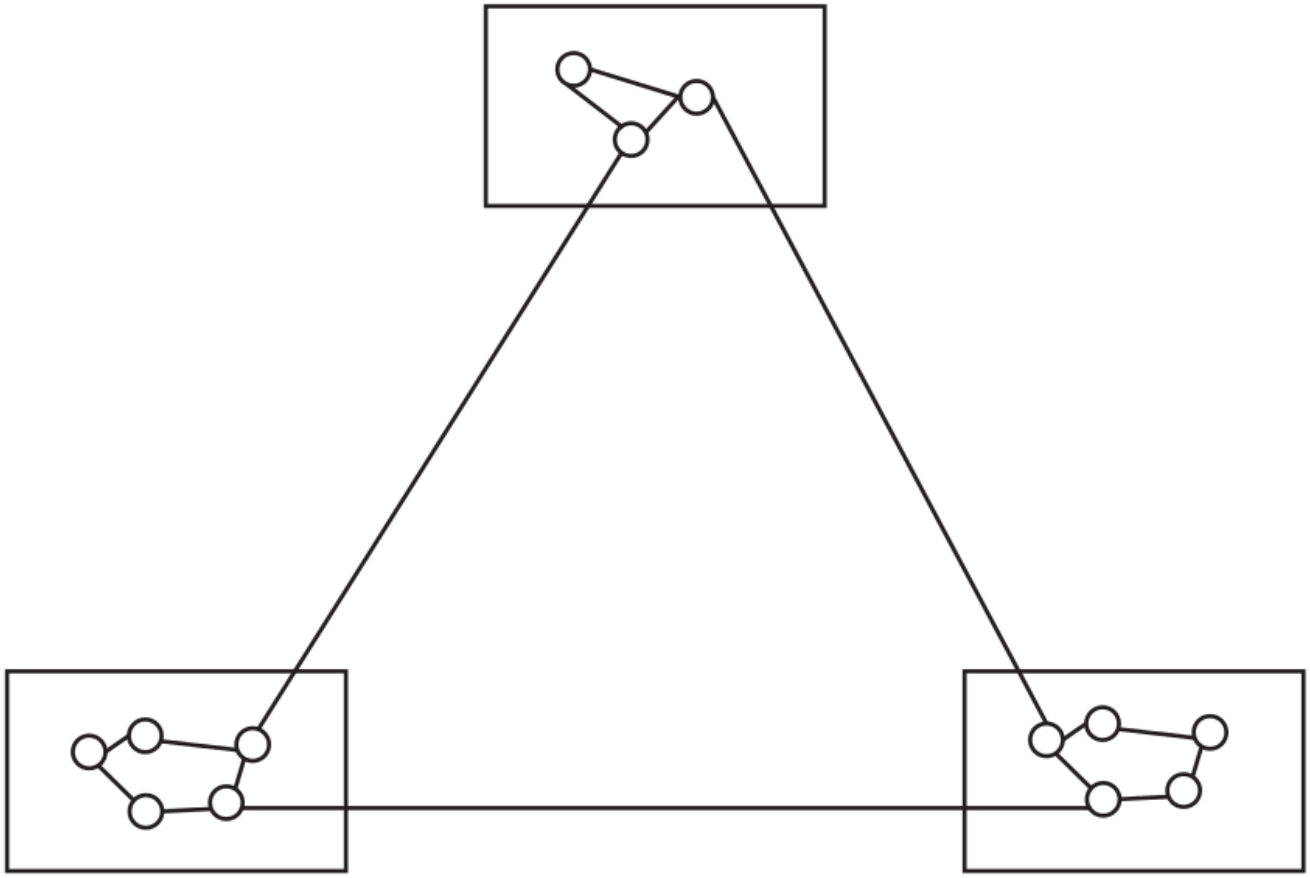
Software Engineering and Testing: An Introduction (Computer Science) by B.B. Agarwal, M. Gupta, S.P. Tayal

Highly coupled



Software Engineering and Testing: An Introduction (Computer Science) by B.B. Agarwal, M. Gupta, S.P. Tayal

Loosely coupled

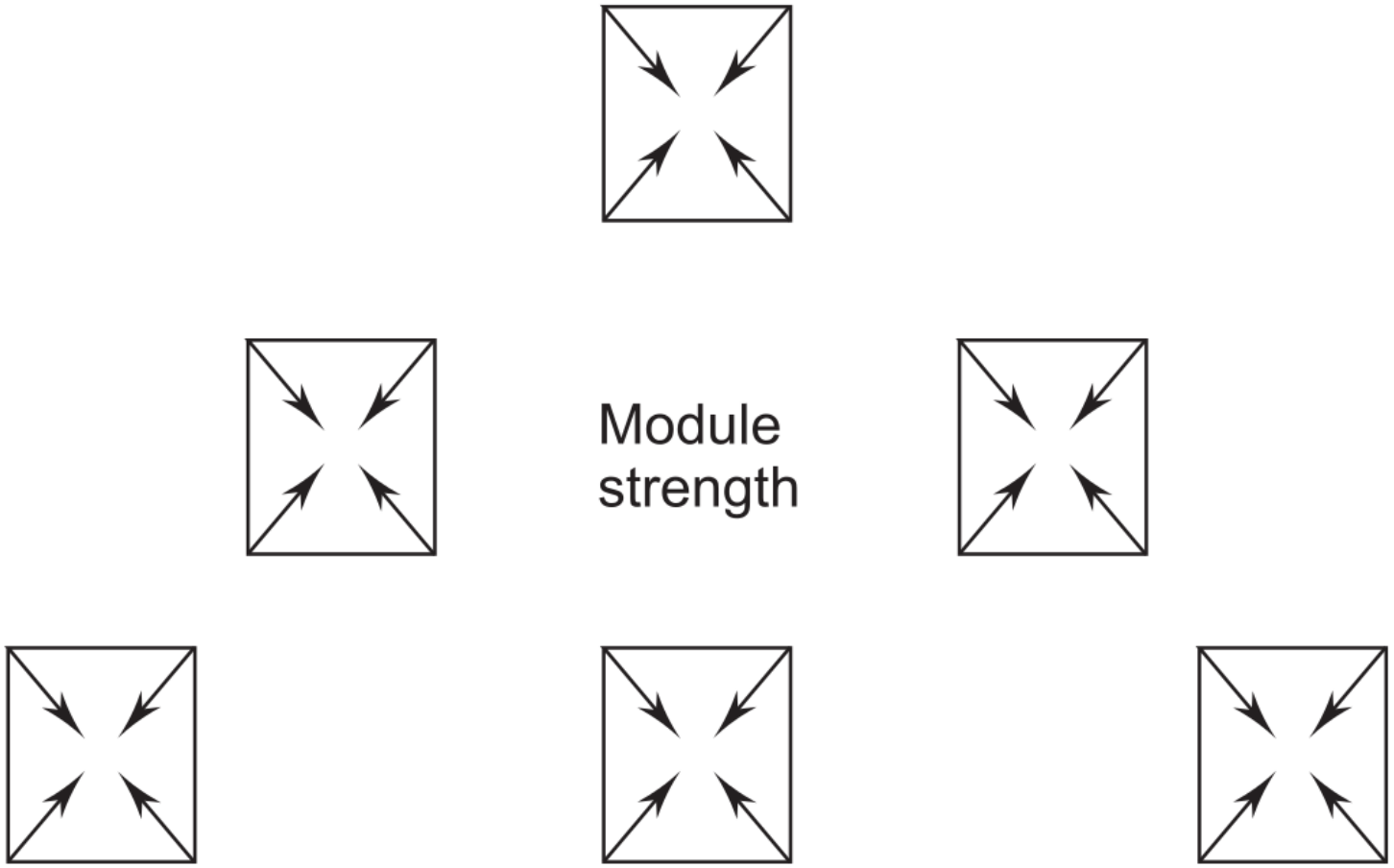


Software Engineering and Testing: An Introduction (Computer Science) by B.B. Agarwal, M. Gupta, S.P. Tayal

Cohesion

- Cohesion is a measure of the relative functional strength of a module.
- The cohesion of a component is a measure of the closeness of the relationships between its components.

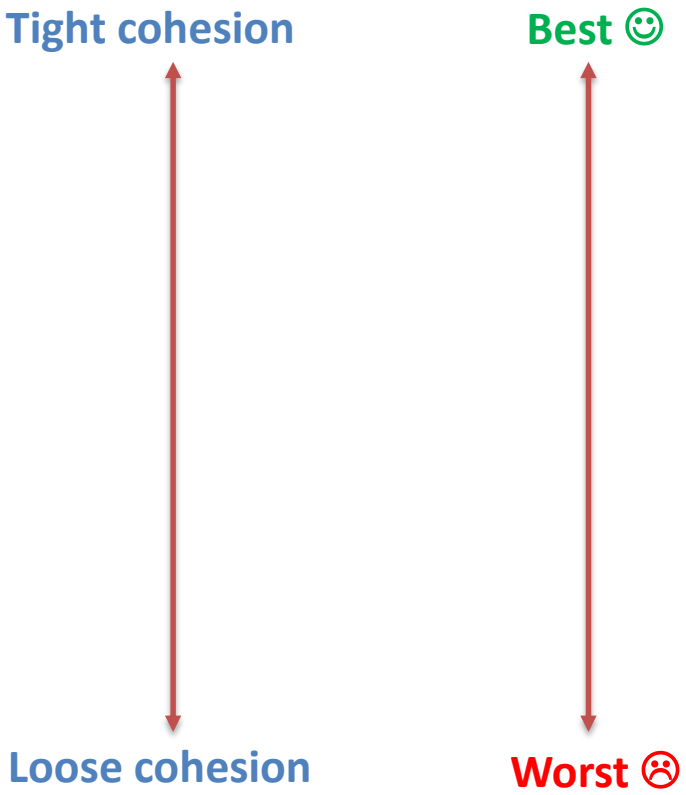
Cohesion-strength of Relation within Modules



Software Engineering and Testing: An Introduction (Computer Science) by B.B. Agarwal, M. Gupta, S.P. Tayal

The types of module cohesion

- Functional cohesion
- Sequential cohesion
- Communicational cohesion
- Procedural cohesion
- Temporal cohesion
- Logical cohesion
- Concidental cohesion

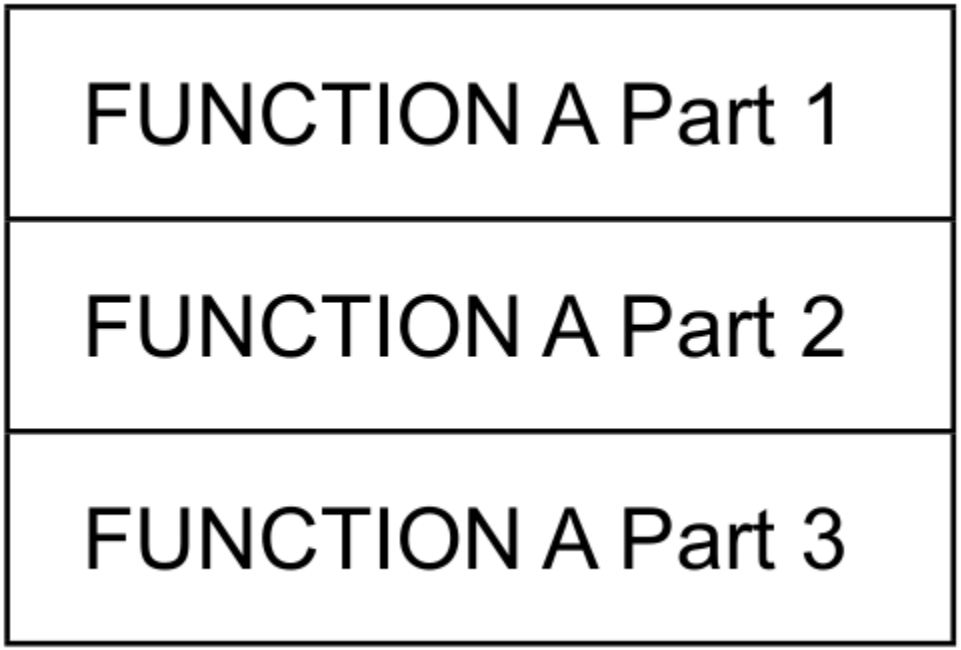


Functional cohesion

- Functional cohesion is said to exist if different elements of a module cooperate to achieve a single function (e.g., managing an employee's payroll)
- When a module displays functional cohesion, and if we are asked to describe what the module does, we can describe it using a single sentence.

Functional cohesion

Sequential with complete, related functions



Functional cohesion

Problems

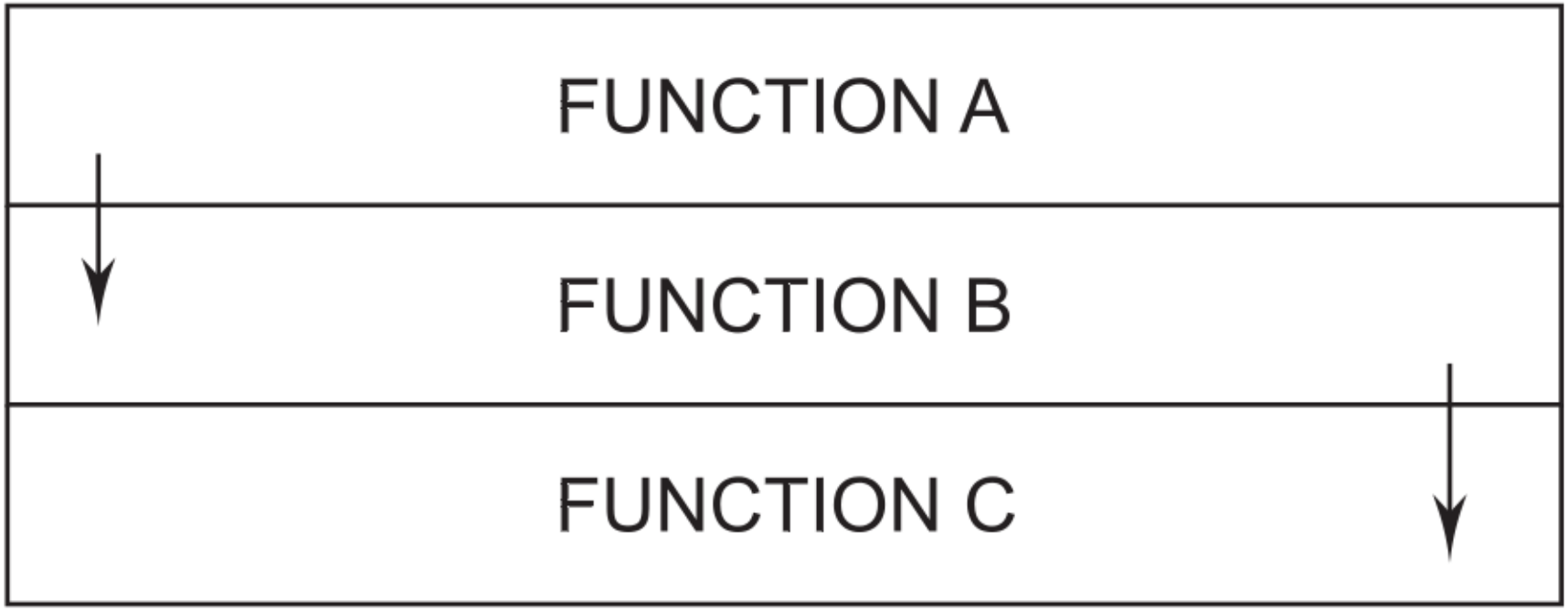
- cannot always be achieved ☺.

Sequential cohesion

- A module is said to possess sequential cohesion if the elements of a module form the parts of a sequence, where the output from one element of the sequence is input to the next.

Sequential cohesion

Output of one part is input to next



Software Engineering and Testing: An Introduction (Computer Science) by B.B. Agarwal, M. Gupta, S.P. Tayal

Sequential cohesion

Problems

- hidden interface(s)
- fixed combination and order of tasks
- bad reusability, maintainability

Communicational cohesion

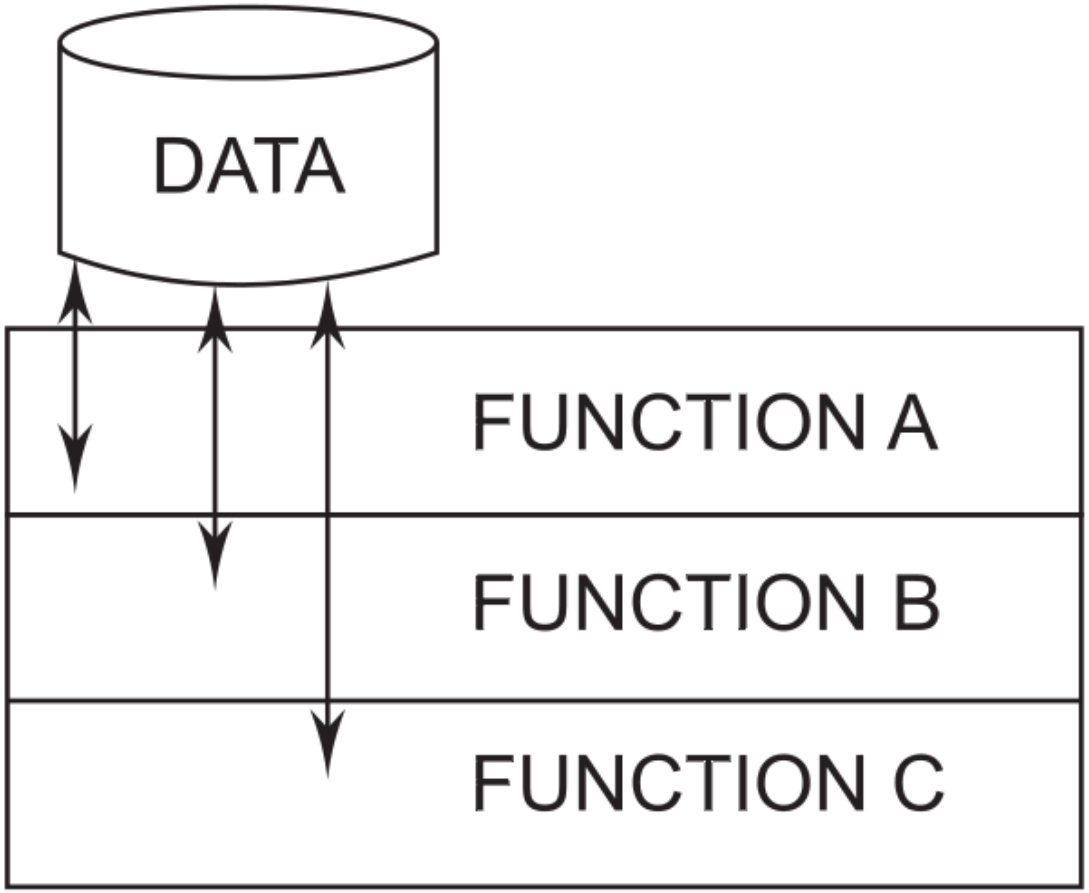
- A module is said to have communicational cohesion if all the functions of the module refer to or update the same data

Communicational cohesion

- A module with communicational cohesion has elements that are related by a reference to the same input or output data.

Communicational cohesion

Access same data



Software Engineering and Testing: An Introduction (Computer Science) by B.B. Agarwal, M. Gupta, S.P. Tayal

Communicational cohesion

Problems

- wide interface (with stamp data coupling)
- code sharing is tempting
- bad reusability, maintainability

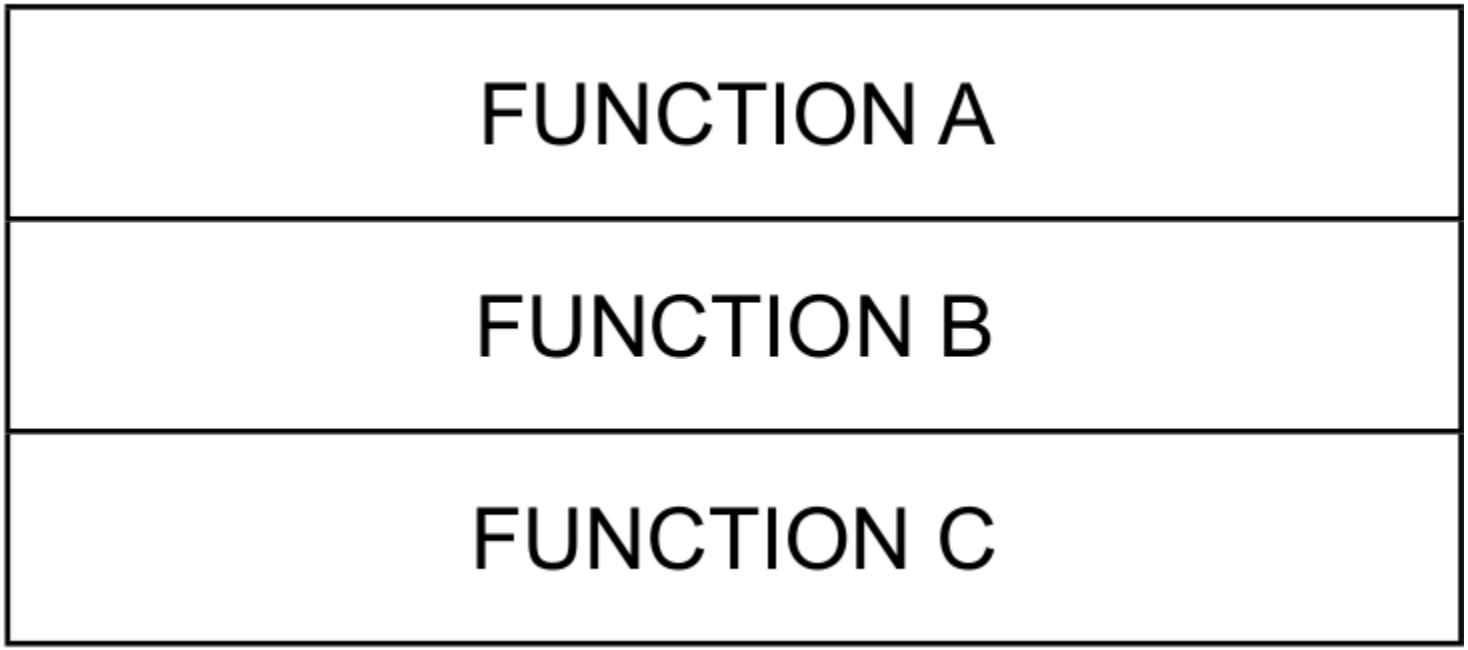
Procedural cohesion

- A module is said to possess procedural cohesion if the set of functions of the module are all part of a procedure (algorithm) in which a certain sequence of steps has to be carried out for achieving an objective

Procedural cohesion

- A procedurally cohesive module contains elements that belong to a common procedural unit.

Procedural cohesion related by order of function



Software Engineering and Testing: An Introduction (Computer Science) by B.B. Agarwal, M. Gupta, S.P. Tayal

Procedural cohesion

```
public class MakeCake
{
    public void AddIngredients()...
    public void Mix()...
    public void Bake()...
}
```

http://www.cs.sjsu.edu/~pearce/modules/lectures/ood/metrics/Cohesion.htm

Procedural cohesion

Problems

- fixed combination and order of tasks
- bad reusability, maintainability

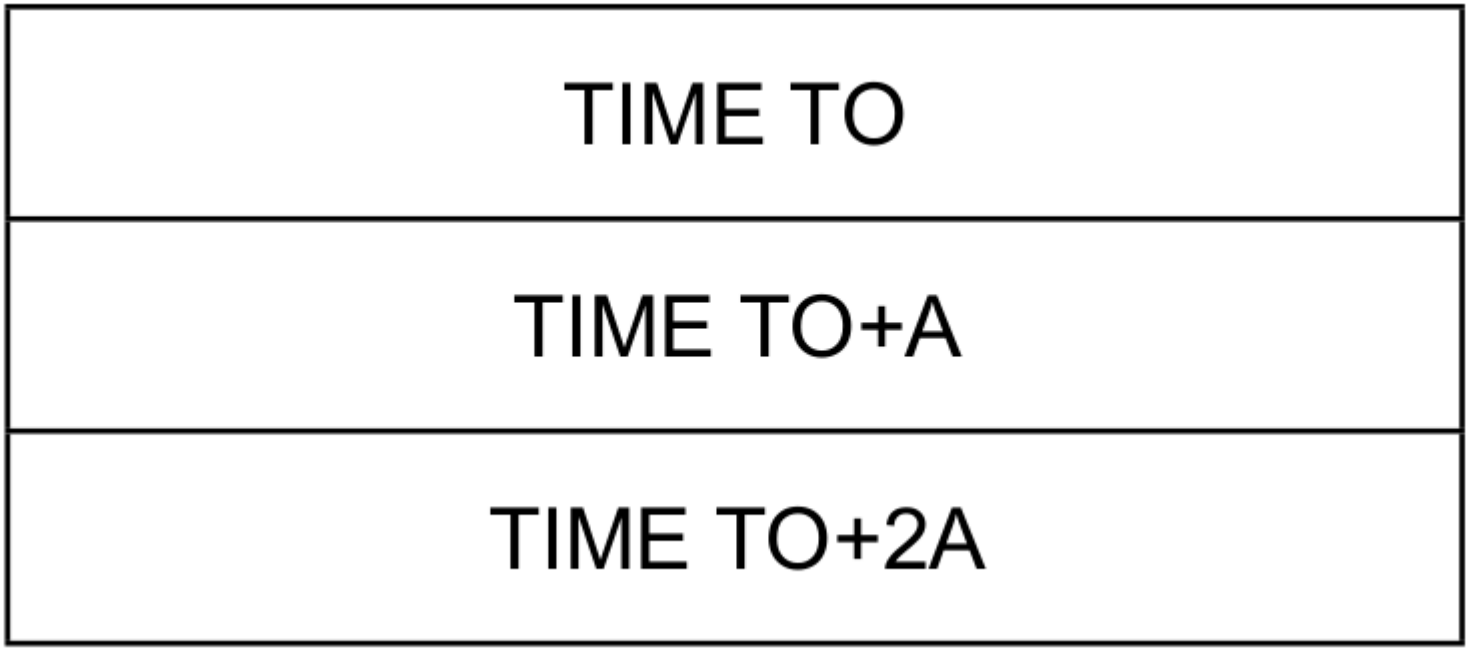
Temporal cohesion

- When a module contains functions that are related by the fact that all the functions must be executed in the same time span, the module is said to exhibit temporal cohesion.

Temporal cohesion

- Temporal cohesion is the same as logical cohesion, except that the elements are also related in time and are executed together.

Temporal cohesion related by time



Software Engineering and Testing: An Introduction (Computer Science) by B.B. Agarwal, M. Gupta, S.P. Tayal

Temporal cohesion

```
public class InitFuns {
    public void InitDisk()...
    public void InitPrinter()...
    public void InitMonitor()...
}
```

http://www.cs.sjsu.edu/~pearce/modules/lectures/ood/metrics/Cohesion.htm

Temporal cohesion

Problems

- fixed combination and order of tasks
- bad reusability, maintainability

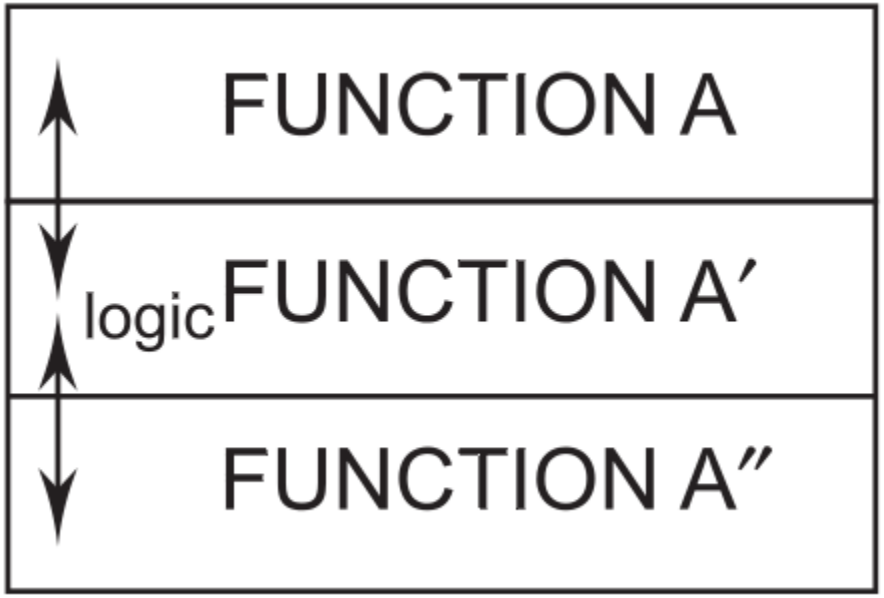
Logical cohesion

- A module is said to be logically cohesive if all elements of the module perform similar operations
- An example of logical cohesion is the case where a set of print functions generating different output reports are arranged into a single module.

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.
- A example of this kind of cohesion is a module that performs all the inputs or all the outputs.

Logical cohesion similar functions



Logical cohesion

```
public class AreaFuns
{
    public double CircleArea()...
    public double RectangleArea() ...
    public double TriangleArea()...
}
```

http://www.cs.sjsu.edu/~pearce/modules/lectures/ood/metrics/Cohesion.htm

Logical cohesion

Problems

- multiple functions
- bad reusability, maintainability
- violation of the principle of „local scope”

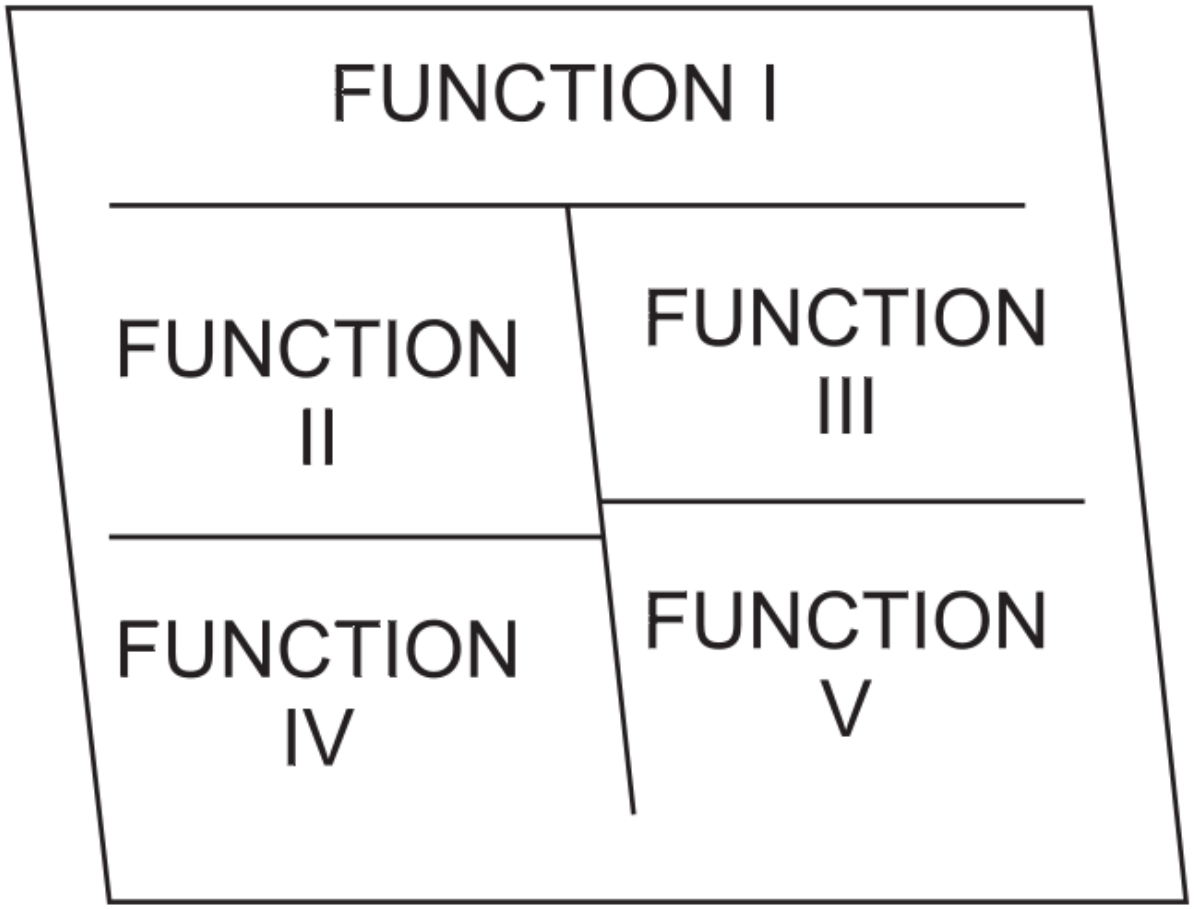
Coincidental cohesion

- A module is said to have coincidental cohesion if it performs a set of tasks that relate to each other very loosely.
- In this case, the module contains a random collection of functions.

Concidental cohesion

Coincidental cohesion occurs when there is no meaningful relationship among the elements of a module.

Coincidental cohesion parts unrelated



Software Engineering and Testing: An Introduction (Computer Science) by B.B. Agarwal, M. Gupta, S.P. Tayal

Concidental cohesion

```
public class MyFuns
{
    public void InitPrinter()...
    public double CalcInterest()...
    public DateTime GetDate()...
}
```

http://www.cs.sjsu.edu/~pearce/modules/lectures/ood/metrics/Cohesion.htm

Coincidental cohesion

Problems

- multiple functions
- no logical connection between functions

Coincidental, logical, temporal

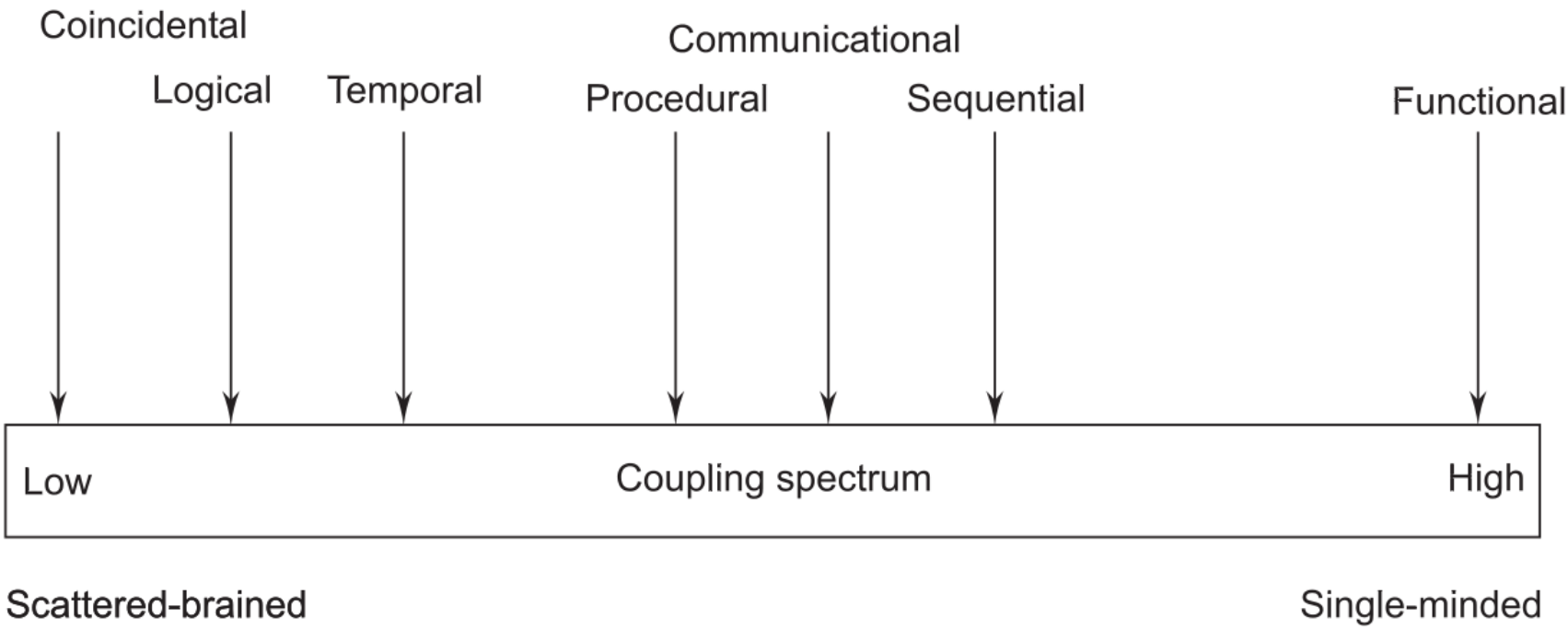
- One reason why coincidental, logical, and temporal cohesion are at the low end of our cohesion scale is because instances of such classes are unrelated to objects in the application domain.

InitFuns class

```
public class InitFuns {
    public void InitDisk()...
    public void InitPrinter()...
    public void InitMonitor()...
}
```

```
InitFuns x = new InitFuns(), y = new InitFuns();
```

Cohesion



Software Engineering and Testing: An Introduction (Computer Science) by B.B. Agarwal, M. Gupta, S.P. Tayal

Three aspects

Cohesion in OO systems has three aspects:

- Method cohesion
- Class cohesion
- Inheritance cohesion

Method cohesion

- It's the same as cohesion in functional modules.
- It focuses on why the different code elements of a method are together within the method.

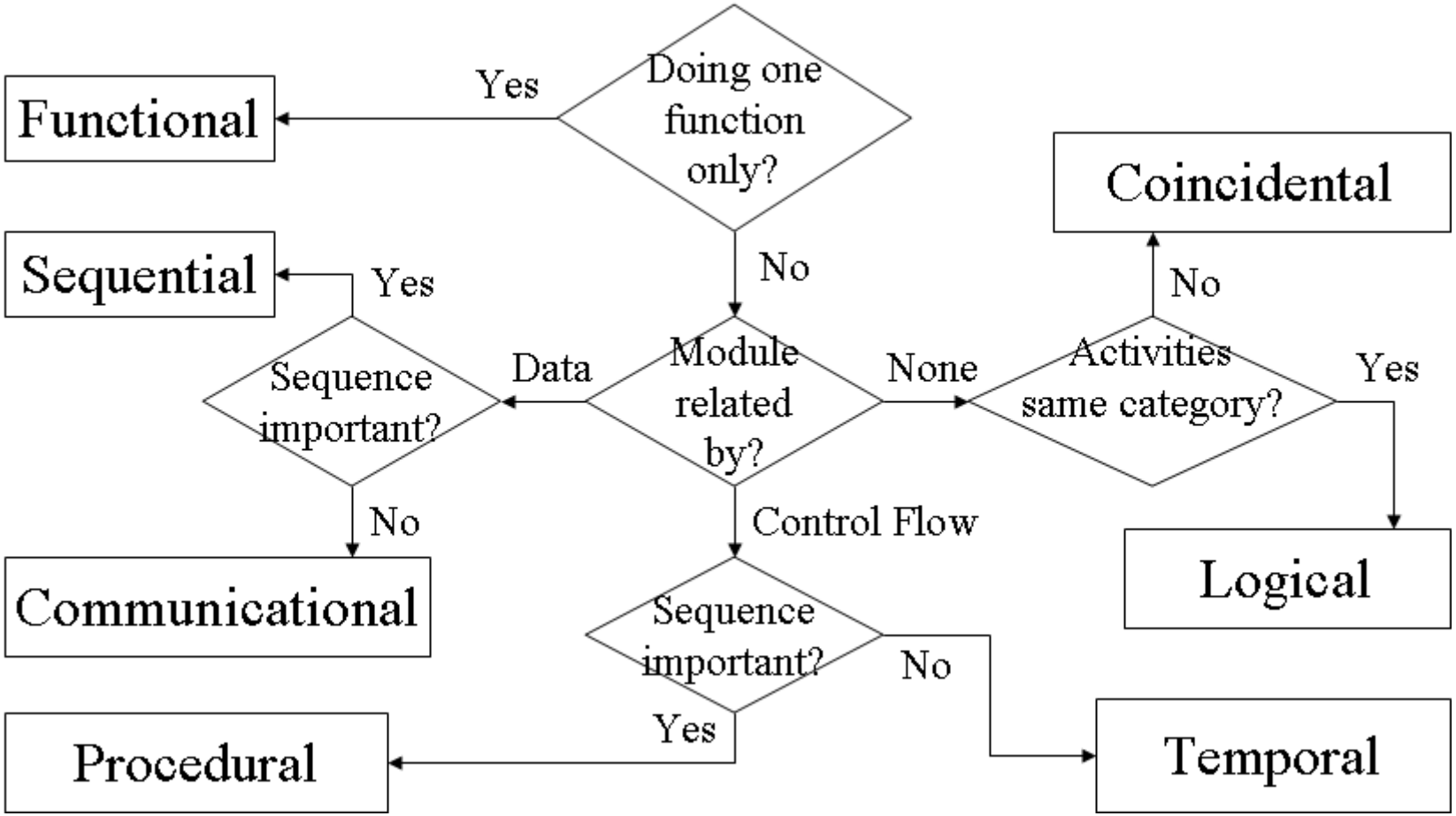
Class cohesion

- It focuses on why different attributes and methods are together in this class.
- The goal is to have a class that implements a single concept or abstraction with all elements contributing toward supporting this concept.

Inheritance cohesion

- Inheritance cohesion focuses on the reason why classes are together in a hierarchy.
- The two main reasons for inheritance are to model generalization-specialization relationship, and for code reuse.

Determining module cohesion



Why high cohesion?

- High cohesion usually means low coupling
- It insures that the functional breakdown of the program reflects the functional organization of the original problem
- High cohesion has been shown to be a good predictor of maintainability

Relationship between coupling and cohesion

- A software engineer must design the modules with the goal of high cohesion and low coupling.
- A good example of a system that has high cohesion and low coupling is the „plug and play” feature of a computer system.

Resources

Books and papers

- **Software Engineering and Testing: An Introduction (Computer Science)** by *B.B. Agarwal, M. Gupta, S.P. Tayal*
- **A Concise Introduction to Software Engineering** by *Pankaj Jalote*
- **Planning efficient software tests** by *M.S. Phadke*
- **Measuring Coupling and Cohesion In Object-Oriented Systems** by Martin Hitz, Behzad Montazeri
- **Measuring Cohesion and Coupling of Object-Oriented Systems** by Martin Hitz, Behzad Montazeri