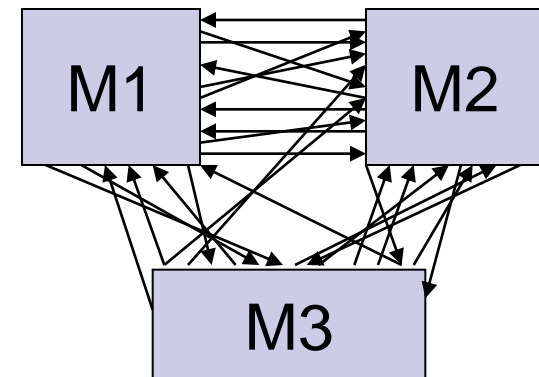# Cohesion and Coupling

Outline

- Cohesion
- Coupling

Characteristics of Good Design

- Component independence
  - High cohesion
  - Low coupling
- Exception identification and handling
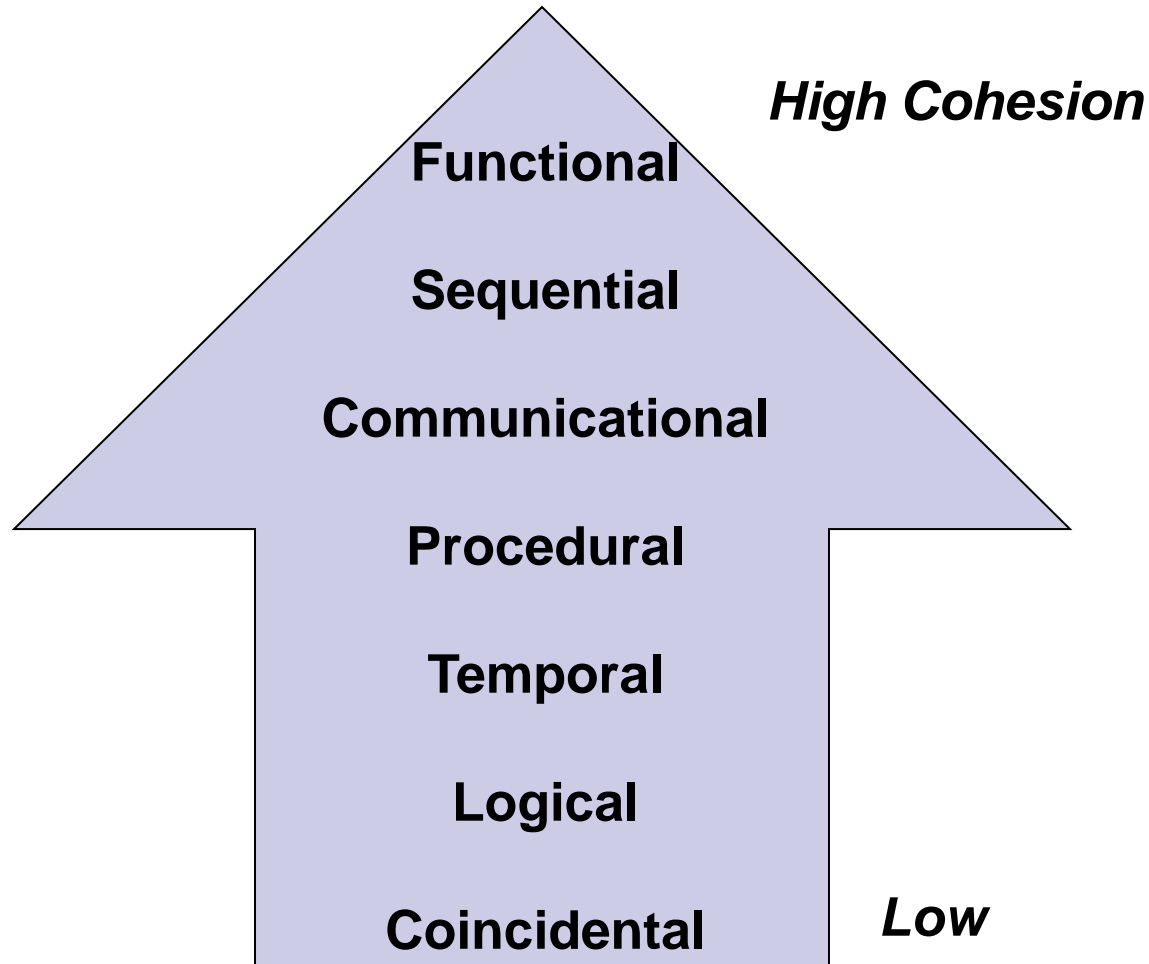- Fault prevention and fault tolerance
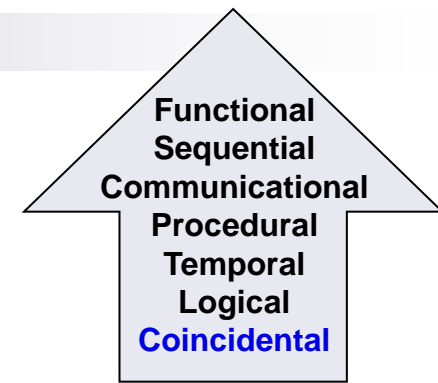- Design for change

M1   M2

M3

Cohesion

- Definition
    - The degree to which all elements of a component are directed towards a single task.
    - The degree to which all elements directed towards a task are contained in a single component.
    - The degree to which all responsibilities of a single class are related.
- Internal glue with which component is constructed
- All elements of component are directed toward and essential for performing the same task.

Type of Cohesion

Functional

High Cohesion

Sequential

Communicational

Procedural

Temporal

Logical

Coincidental

Low

Coincidental Cohesion

- Def: Parts of the component are unrelated (unrelated functions, processes, or data)

- Parts of the component are only related by their location in source code.

- Elements needed to achieve some functionality are scattered throughout the system.

- Accidental

- Worst form

Example

1. Print next line
2. Reverse string of characters in second argument
3. Add 7 to 5th argument
4. Convert 4th argument to float

Logical Cohesion
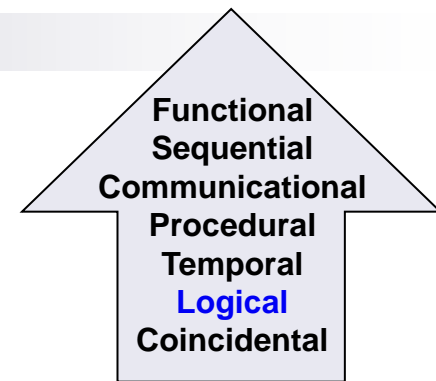
- Def: Elements of component are related logically and not functionally.

- Several logically related elements are in the same component and one of the elements is selected by the client component.
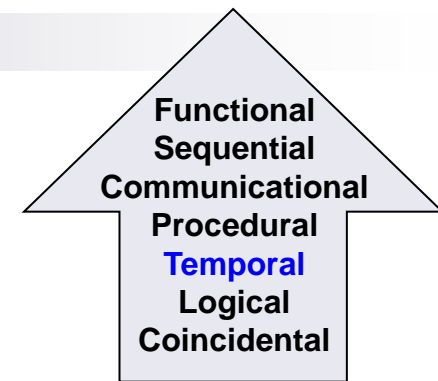
8

Example

- A component reads inputs from tape, disk, and network.

- All the code for these functions are in the same component.

- Operations are related, but the functions are significantly different.

Improvement?

Temporal Cohesion

- Def: Elements are related by timing involved
- Elements are grouped by when they are processed.
- Example: An exception handler that
  - ☐ Closes all open files
  - ☐ Creates an error log
  - ☐ Notifies user
  - ☐ Lots of different activities occur, all at same time
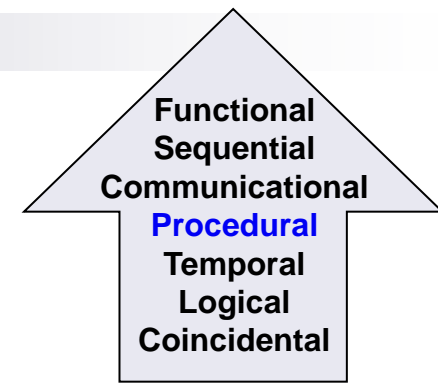
Example

- A system initialization routine: this routine contains all of the code for initializing all of the parts of the system. Lots of different activities occur, all at init time.
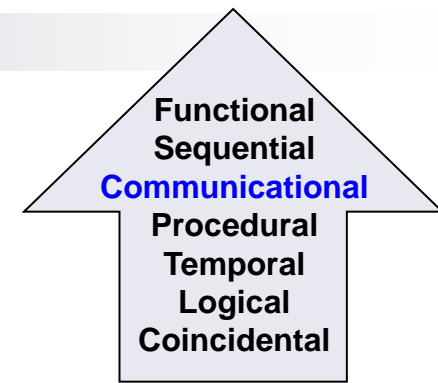
Improvement?

Procedural Cohesion

- **Def: Elements of a component are related only to ensure a particular order of execution.**

- Actions are still weakly connected and unlikely to be reusable.

- Example:

  - ...
  - Write output record
  - Read new input record
  - Pad input with spaces
  - Return new record
 - ...

Communicational Cohesion

- Def: Functions performed on the same data or to produce the same data.

- Examples:
  - ☐ Update record in data base and send it to the printer
    - Update a record on a database
    - Print the record
  - ☐ Fetch unrelated data at the same time.
    - To minimize disk access

Sequential Cohesion
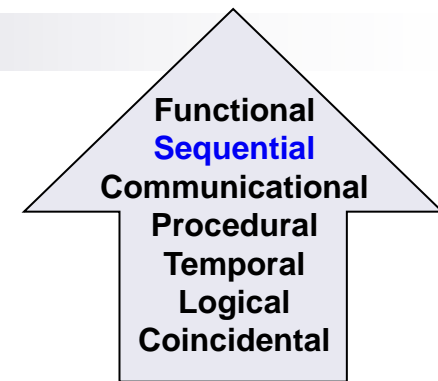
- Def: The output of one part is the input to another.

- *Data flows* between parts (different from procedural cohesion)

- Occurs naturally in functional programming languages

- Good situation
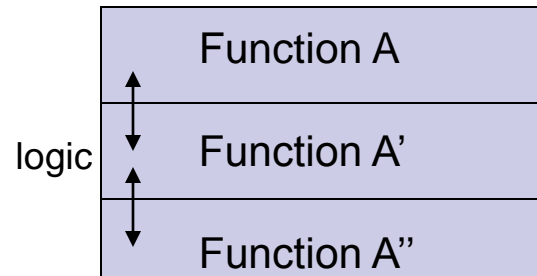
Functional Cohesion
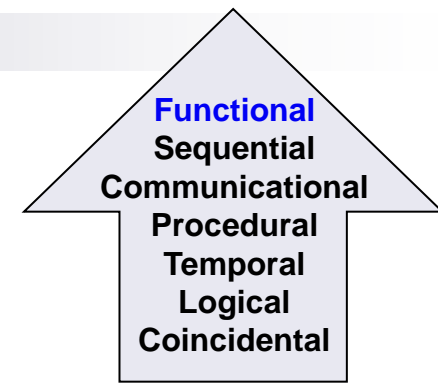
- Def: Every essential element to a single computation is contained in the component.

- Every element in the component is essential to the computation.

- Ideal situation

- What is a functionally cohesive component?
  - One that not only performs the task for which it was designed but
  - it performs only that function and nothing else.

Examples of Cohesion

Function A	
Function B	Function C
Function D	Function E

Coincidental
Parts unrelated

logic

Function A
Function A'
Function A''

Logical
Similar functions

Time t_0
Time $t_0 + X$
Time $t_0 + 2X$

Temporal
Related by time

Function A
Function B
Function C

Procedural
Related by order of functions

Examples of Cohesion (Cont.)

Function A
Function B
Function C
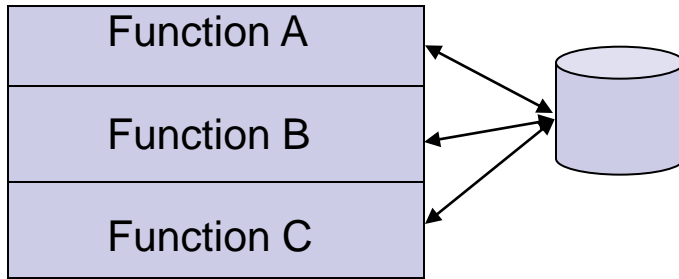
Communicational
Access same data

Function A
Function B
Function C
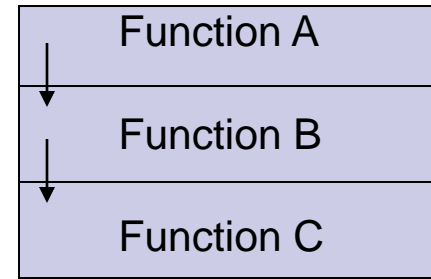
Sequential
Output of one is input to another

Function A part 1
Function A part 2
Function A part 3
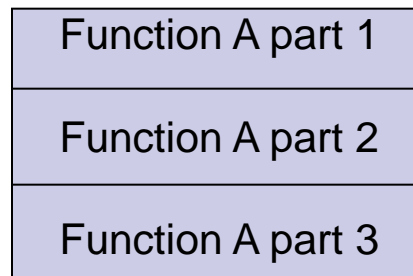
Functional
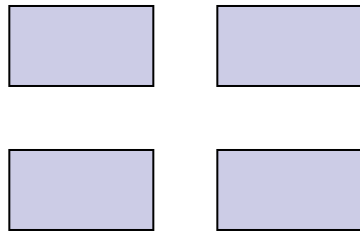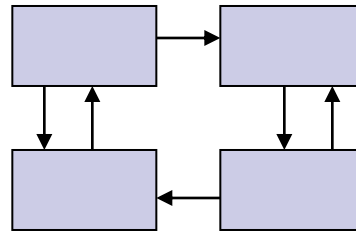Sequential with complete, related functions

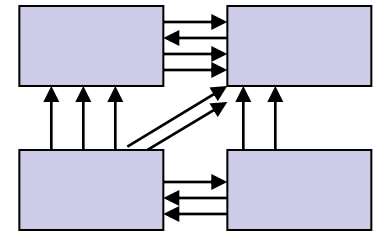# Outline

- ✓ Cohesion
- ■ Coupling

Coupling

- **The degree of dependence such as the amount of interactions among components**

No dependencies

Loosely coupled
some dependencies

Highly coupled
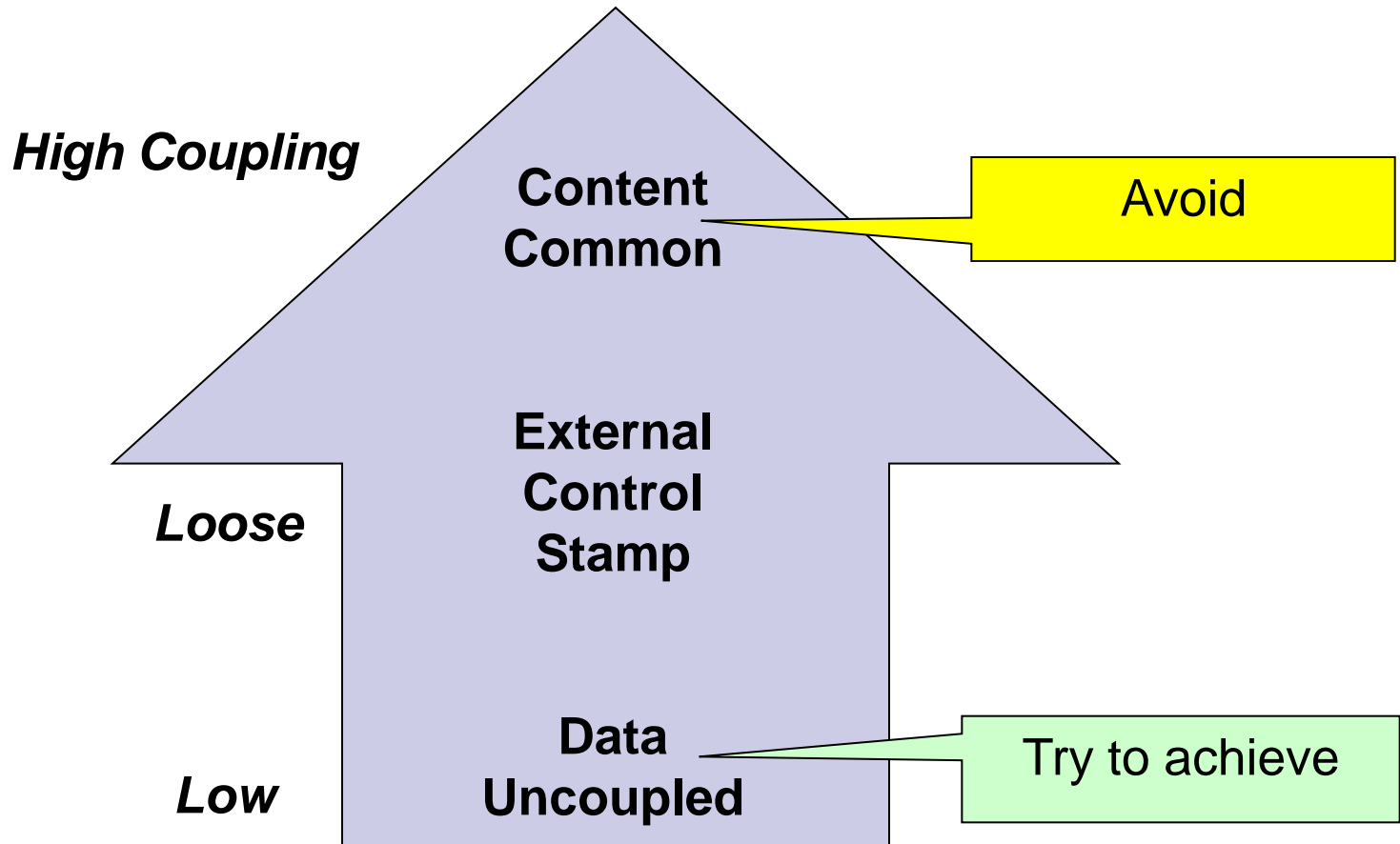many dependencies

Coupling

- The degree of dependence such as the amount of interactions among components

- How can you tell if two components are coupled?

- (In pairs, 2 minutes)

Indications of Coupling

- ?

Type of Coupling

High Coupling

**Content
Common**

Avoid

Loose

**External
Control
Stamp**

Low

**Data
Uncoupled**

Try to achieve

Content Coupling

Content
Common
External
Control
Stamp
Data
Uncoupled

- Def: One component modifies another.

- Example:
  - Component directly modifies another's data
  - Component modifies another's code, e.g., jumps (goto) into the middle of a routine

- Question
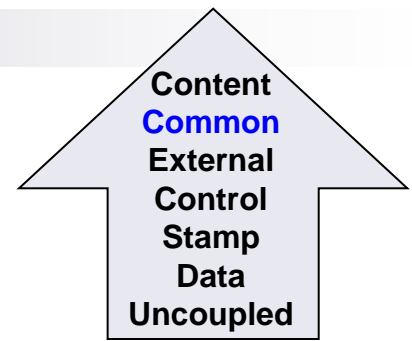 - Language features allowing this?

23

Example

Part of a program handles lookup for customer.

When customer not found, component adds customer by directly modifying the contents of the data structure containing customer data.

Improvement?

Common Coupling

- **Def: More than one component share data such as global data structures**

- Usually a poor design choice because
  - Lack of clear responsibility for the data
  - Reduces readability
  - Difficult to determine all the components that affect a data element (reduces maintainability)
  - Difficult to reuse components
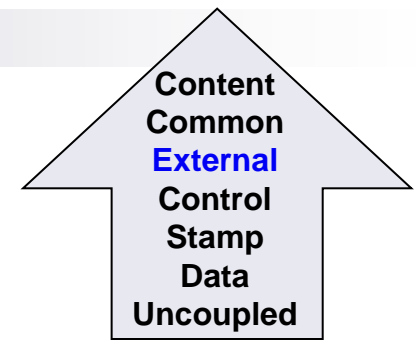 - Reduces ability to control data accesses

Example

Process control component maintains current data about state of operation. Gets data from multiple sources. Supplies data to multiple sinks. Each source process writes directly to global data store. Each sink process reads directly from global data store.
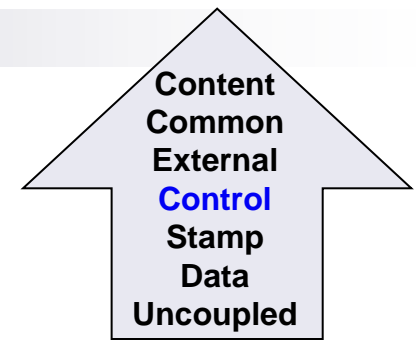
Improvement?

External Coupling

- Def: Two components share something externally imposed, e.g.,
  - ☐ External file
  - ☐ Device interface
  - ☐ Protocol
  - ☐ Data format
- Improvement?

Control Coupling

- Def: Component passes control parameters to coupled components.

- May be either good or bad, depending on situation.
  - Bad if parameters indicate completely different behavior
  - Good if parameters allow factoring and reuse of functionality

- Good example: sort that takes a comparison function as an argument.
  - The sort function is clearly defined: return a list in sorted order, where sorted is determined by a parameter.

Stamp Coupling
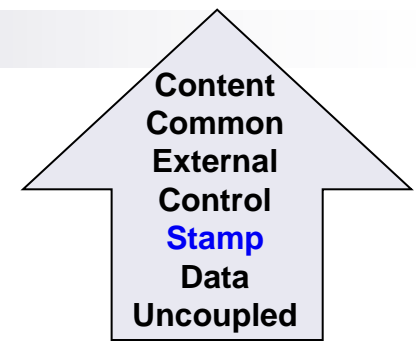
- Def: Component passes a data structure to another component that does not have access to the entire structure.

- Requires second component to know how to manipulate the data structure (e.g., needs to know about implementation).

- The second has access to more information that it needs.

- May be necessary due to efficiency factors: this is a choice made by insightful designer, not lazy programmer.
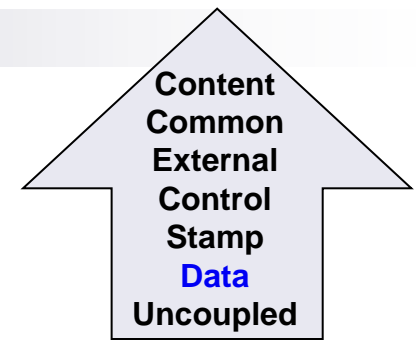
Example

Customer Billing System

The print routine of the customer billing accepts customer data structure as an argument, parses it, and prints the name, address, and billing information.
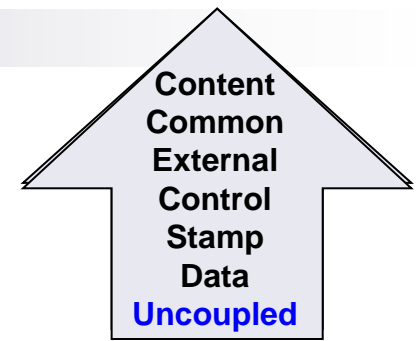
Improvement?

Data Coupling

- Def: Component passes data (not data structures) to another component.

- Every argument is simple argument or data structure in which all elements are used

- Good, if it can be achieved.

- Example: Customer billing system
  - The print routine takes the customer name, address, and billing information as arguments.

Uncoupled

- Completely uncoupled components are not systems.

- Systems are made of interacting components.

Consequences of Coupling

- Why does coupling matter? What are the costs and benefits of coupling?

- (pairs, 3 minutes)

Consequences of Coupling

- **High coupling**
  - ☐ Components are difficult to understand in isolation
  - ☐ Changes in component ripple to others
  - ☐ Components are difficult to reuse
    - Need to include all coupled components
    - Difficult to understand
- **Low coupling**
  - ☐ May incur performance cost
  - ☐ Generally faster to build systems with low coupling

In Class

Groups of 2 or 3:

- P1: What is the effect of cohesion on maintenance?

- P2: What is the effect of coupling on maintenance?