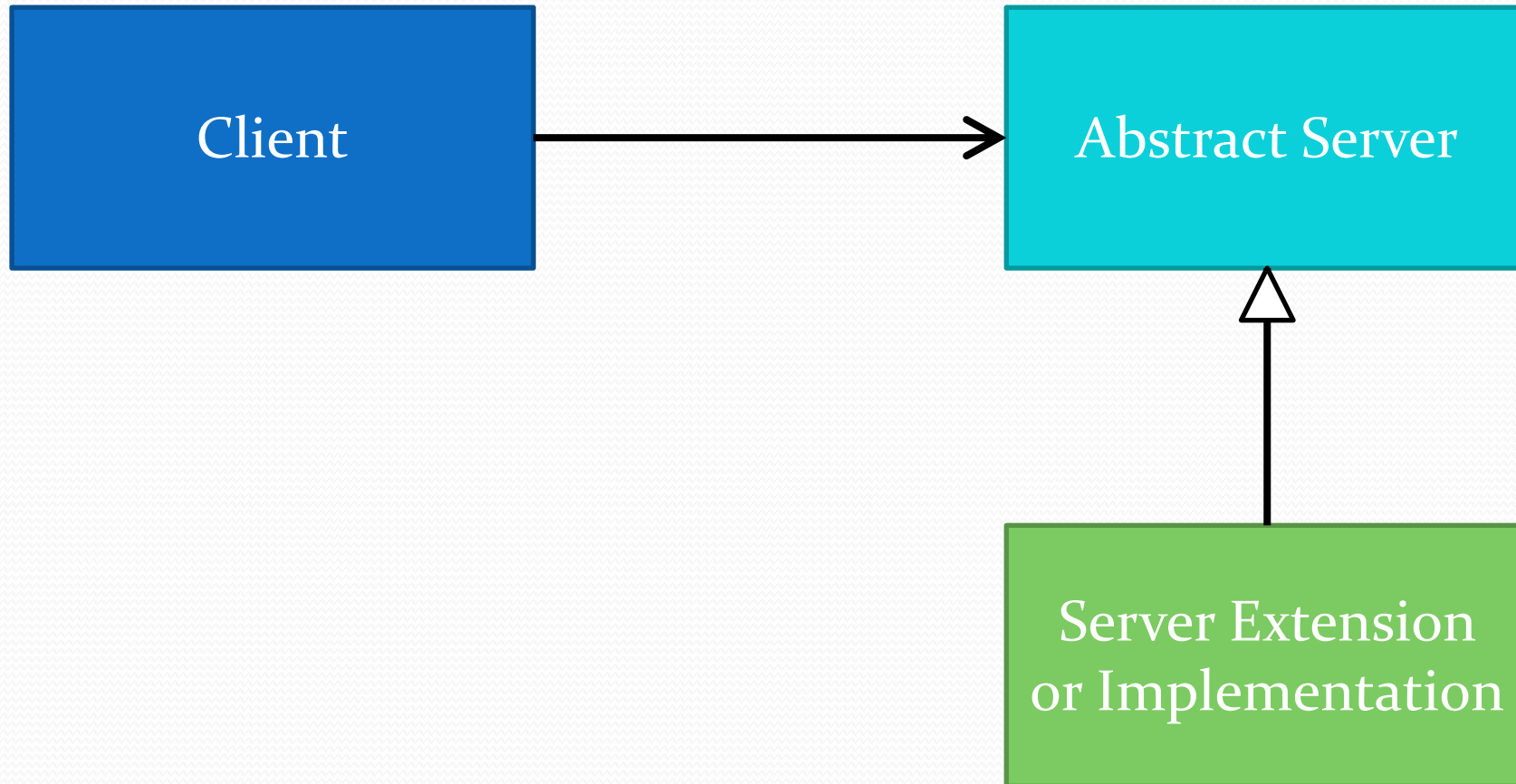# Design Defects and Restructuring

Lecture 4

Sat, Oct 02, 2021

Common Design Practice

Data Model

- Collection of conceptual tools for describing data, data relationships, data semantics, and consistency constraints
- A data model provides a way to describe the design of a database at the physical, logical, and view levels
- Types of Data Models
  - Relational Model
  - Entity-Relationship Model
  - Object-Based Data Model
  - Semi-Structured Data Model

Class Exercise

- Library Management
  - Borrowing a book
- University Management
  - Registering in a course

Definition of Coupling

- Coupling is the degree of interaction between two modules
- There are six kinds of module coupling
  - Content Coupling
  - Common Coupling
  - External Coupling
  - Control Coupling
  - Stamp Coupling
  - Data Coupling

Content Coupling

- Two or more modules exhibit **content coupling** if one refers to the "inside" – the "internal" or "private" part – of the other in some way

Common Coupling

- Two or more modules exhibit **common coupling** if they refer to the same global data area – that is, to something that corresponds to a data store on a DFD or a "register" that must be shared by several processes

External Coupling

- Two or more modules exhibit **external coupling** if they share direct access to the same I/O device or are "tied to the same part of the environment external to software" in some other way

Control Coupling

- Two modules exhibit **control coupling** if one ("module A") passes to the other ("module B") a piece of information that is intended to control the internal logic of the other
  - This will often be a value used in a test for a case statement, if-then statement, or while loop, in module B's source code

Stamp Coupling

- Two modules ("A" and "B") exhibit **stamp coupling** if one passes directly to the other "composite" piece of data – that is, a piece of data with meaningful internal structure – such as a record (or structure), array, or (pointer to) a list or tree

Data Coupling

- Two modules exhibit **data coupling** if one calls the other directly and they communicate using "parameters" – a simple list of inputs and outputs
  - The modules exhibit stamp coupling if "composite" data types are used for parameters as well
- Ideally, this **data coupling** is the usual type of interaction between modules that need to communicate at all

Definition of Cohesion

- Cohesion refers to the strength of a method as it relates to the routines within it
- There are following categories of cohesion
  - Functional
  - Sequential
  - Communicational
  - Temporal
  - Procedural
  - Logical
  - Coincidental

Functional Cohesion

- A method has strong functional cohesion when it does just one thing
- Examples
  - Compute cosine of angle
  - Read transaction record
  - Determine customer mortgage repayment
  - Calculate net employee salary
  - Assign seat to airline customer

Sequential Cohesion

- A method has sequential cohesion when it depends on another method being called first AND it shares data with the first method
- A sequentially cohesive module is one whose elements are involved in activities such that output data from one activity serves as input data to the next
- Examples (Repaint a car)
  - Clean car body
  - Fill in holes in car
  - Sand car body
  - Apply primer

Communicational Cohesion

- A method is said to have communicational cohesion when it does more than one unrelated thing on the same data
- A communicational cohesive module is one whose elements contribute to activities that use the same input or output data
- Examples (Book)
  - Find title of book
  - Find price of book
  - Find publisher of book
  - Find author of book

Procedural Cohesion

- A method is said to have procedural cohesion when all the routines within the method need to occur in a specified order and the routines don't share data

- A procedurally cohesive module is one whose elements are involved in different and possibly unrelated activities in which control flows from each activity to the next

  - Remember that in a sequentially cohesive module data, not control, flows from one activity to the next

Procedural Cohesion

- Example
  - Clean utensils from previous meal
  - Prepare chicken for roasting
  - Make phone call
  - Take shower
  - Chop vegetables
  - Set table

Temporal Cohesion

- A method is said to have temporal cohesion when all the routines within the method need to occur at the same time, but not necessarily in order
- A temporally cohesive module is one whose elements are involved in activities that are related in time
- Example
  - Put out milk bottles
  - Put out cat
  - Turn off TV
  - Brush teeth

Logical Cohesion

- A method is said to have logical cohesion when the routines within the method are not related, don't share data, and the routine is selected by a flag either passed in as a parameter or, worse, existing outside the method
- A logically cohesive module is one whose elements contribute to activities of the same general category in which the activity or activities to be executed are selected from outside the module
- Example
  - Go by car
  - Go by train
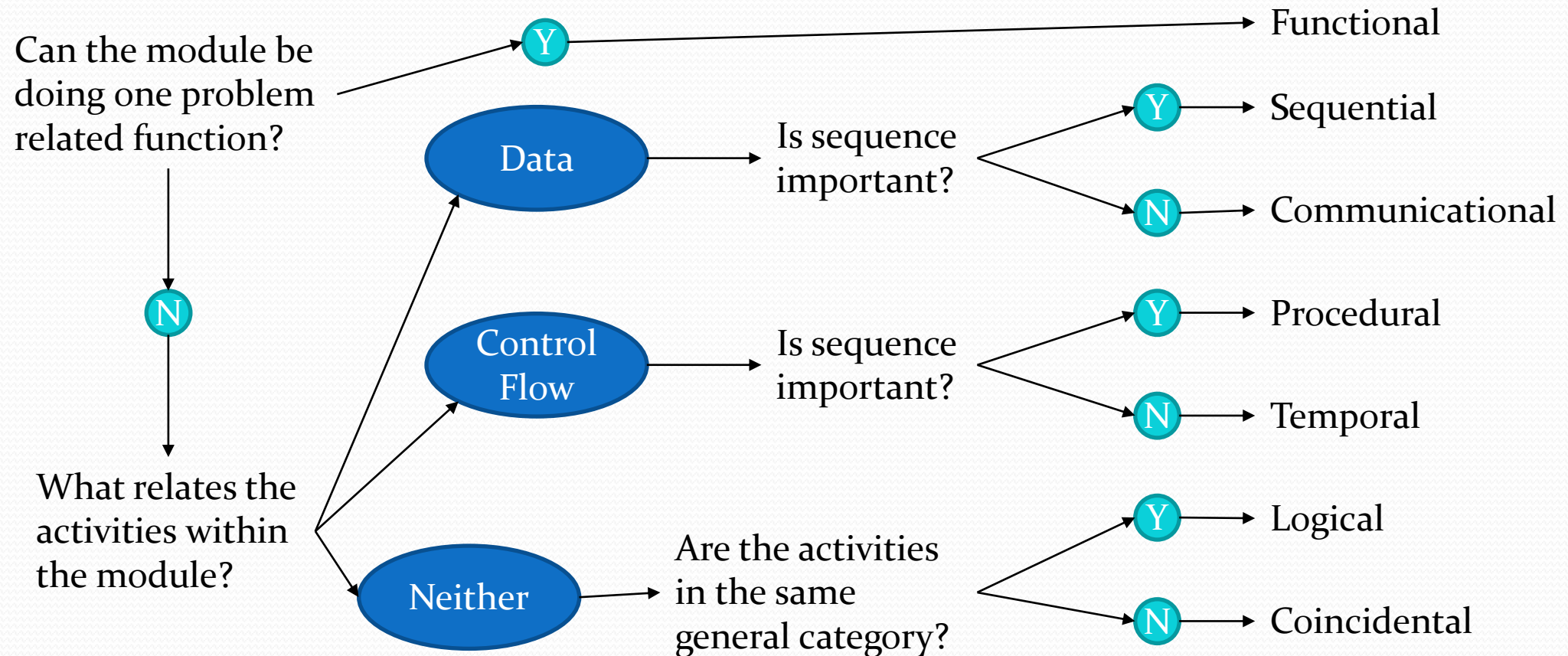 - Go by boat
  - Go by plane

Coincidental Cohesion

- A method is said to have coincidental cohesion when the routines within the method are not related, and don't share data
  - This is a nice term which means the method does not have cohesion (or the cohesion is weak)
- A coincidental cohesive module is one whose elements contribute to activities with no meaningful relationship to one another

Coincidental Cohesion

- Example
  - Fix car
  - Bake cake
  - Walk dog
  - Fill out application form
  - Have a lunch
  - Get out of bed
  - Go to the movies

Decision Tree for Module Cohesion

Can the module be doing one problem related function?

Y → Functional

N →

What relates the activities within the module?

Data → Is sequence important?
- **Y** → Sequential
- **N** → Communicational

Control Flow → Is sequence important?
- **Y** → Procedural
- **N** → Temporal

Neither → Are the activities in the same general category?
- **Y** → Logical
- **N** → Coincidental

Comparison of Level of Cohesion

Cohesion Level	Coupling	Cleanliness of Implementation	Modifiability	Understandability	Effect on Overall System Maintainability
Functional	Good	Good	Good	Good	Good
Sequential	Good	Good	Good	Good	Fairly Good
Communicational	Medium	Medium	Medium	Medium	Medium
Procedural	Variable	Poor	Variable	Variable	Bad
Temporal	Poor	Medium	Medium	Medium	Bad
Logical	Bad	Bad	Bad	Poor	Bad
Coincidental	Bad	Poor	Bad	Bad	Bad

Summary of Module Cohesion

- A module may exhibit any of seven levels of cohesion depending on how the activities within the module are related

- In sequence from best to worst, these seven levels are

  - **Functional**: Elements contribute to a single, problem related activity

  - **Sequential**: Activities within the module are connected in that the output from one serves as the input to another

  - **Communicational**: Activities share the same input or output

  - **Procedural**: Activities share the same procedural implementation

  - **Temporal**: Activities can be carried out at the same time

  - **Logical**: Activities appear to belong to the same general category

  - **Coincidental**: Activities have no relationship to one another