



IBM Software Group

Mastering OOAD: UML 1.x to 2.0 Migration

Module 3: Subsystem Design

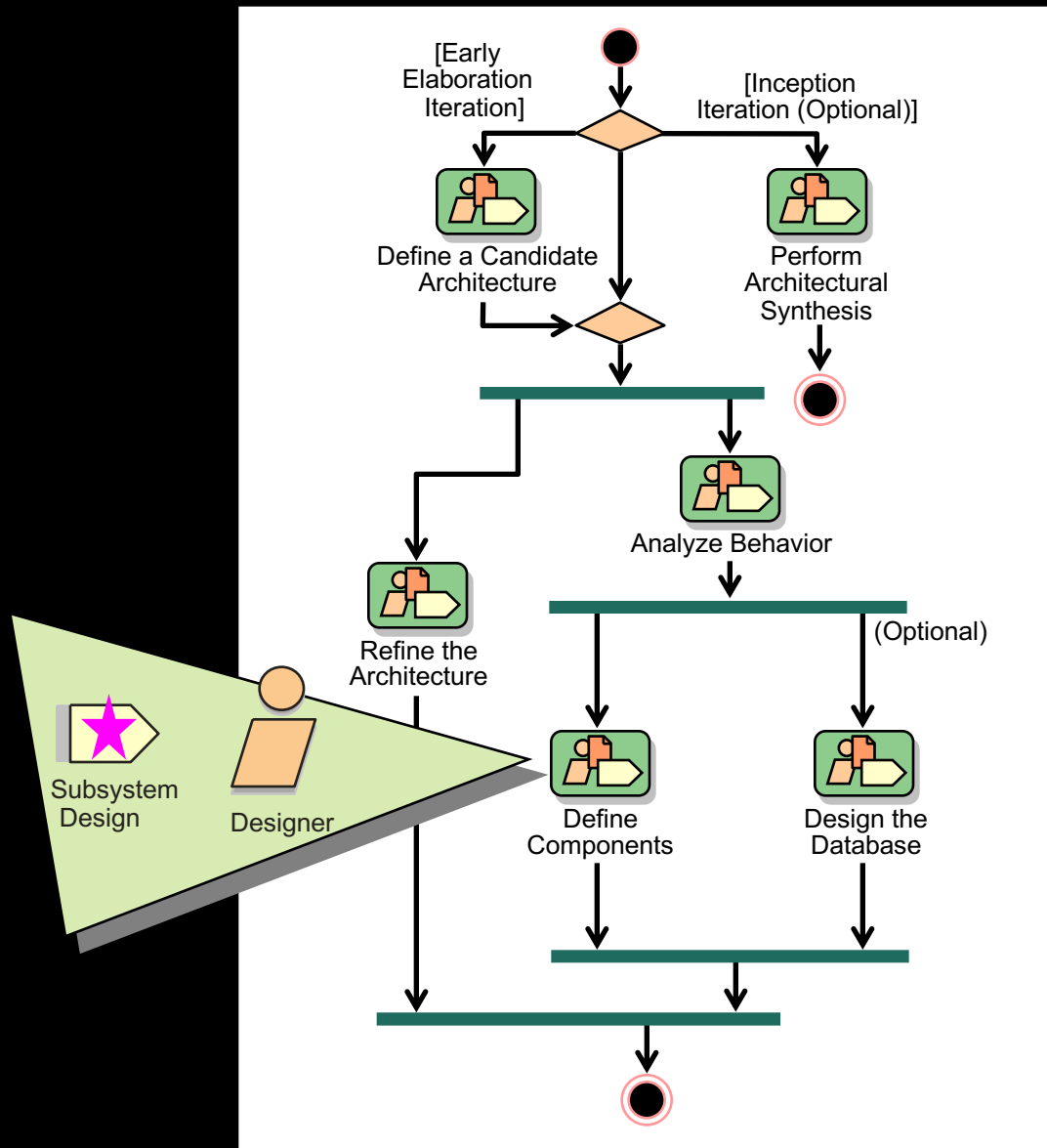
Rational software



Objectives: Subsystem Design

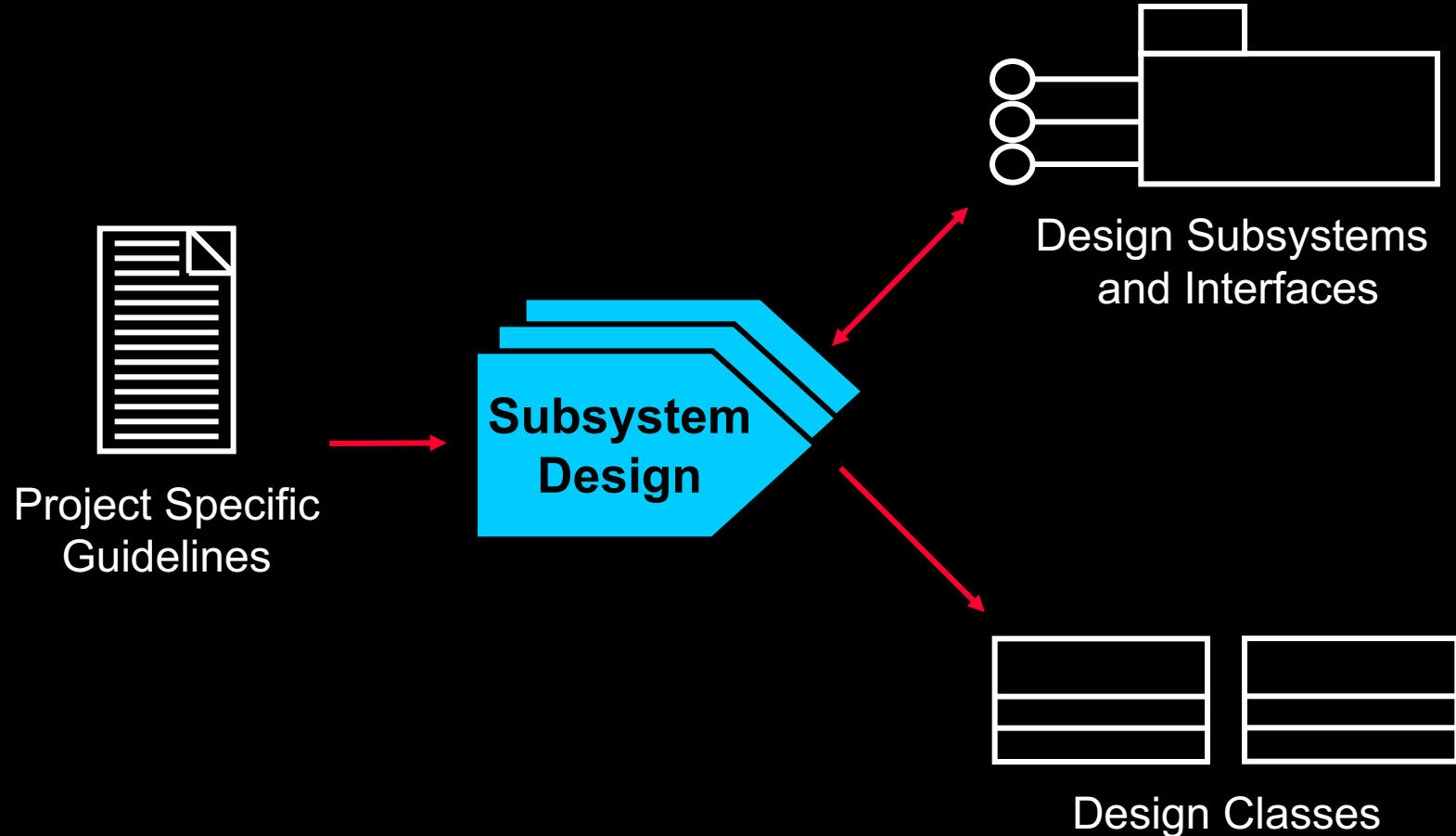
- ◆ Describe how the subsystem's behaviors are distributed to internal elements
- ◆ Explain how to document and model the internal structure of a subsystem
- ◆ Define relationships to external elements, upon which the subsystem might be dependent

Subsystem Design in Context



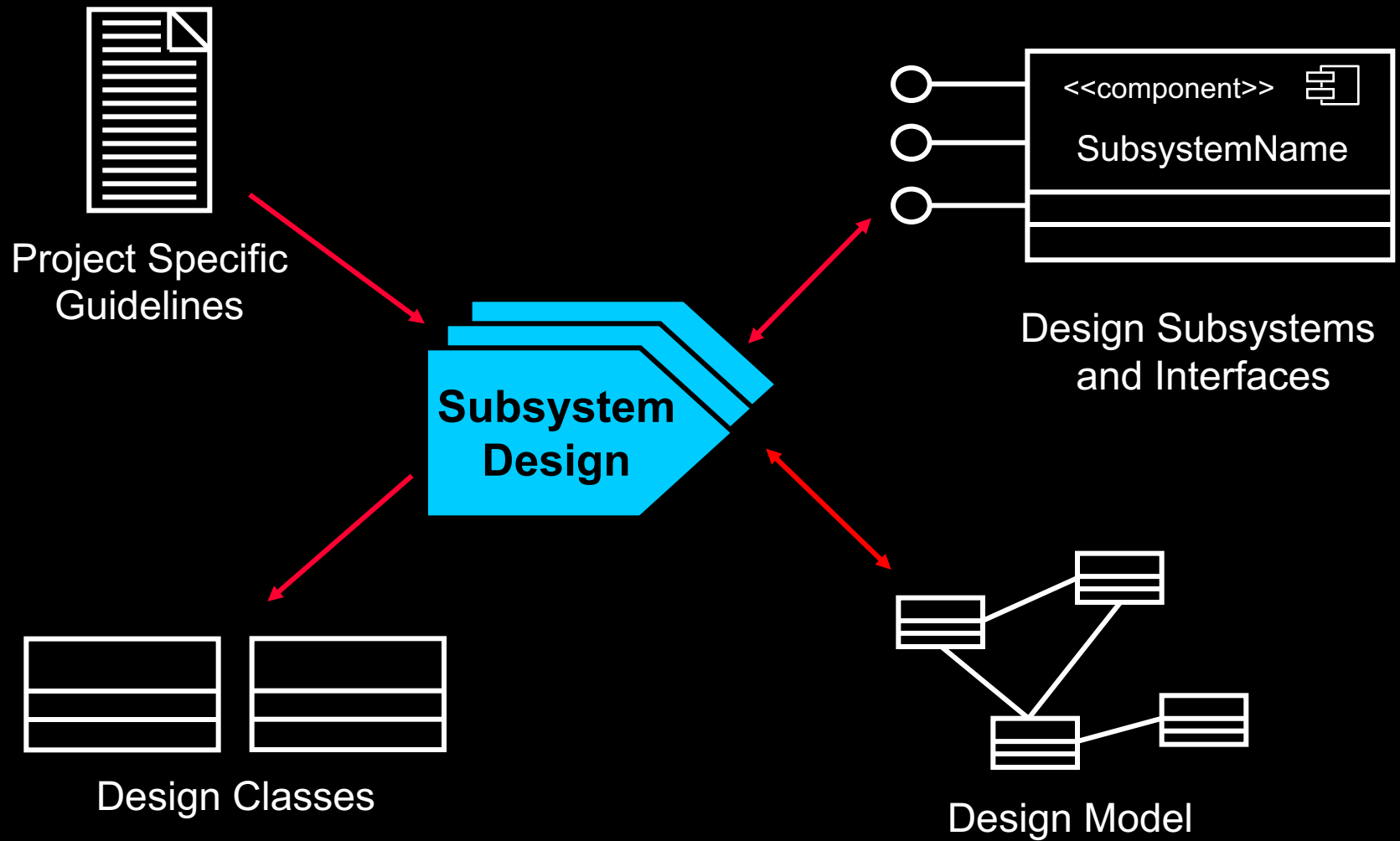
Subsystem Design Overview

(1.x)



Subsystem Design Overview

(2.0)

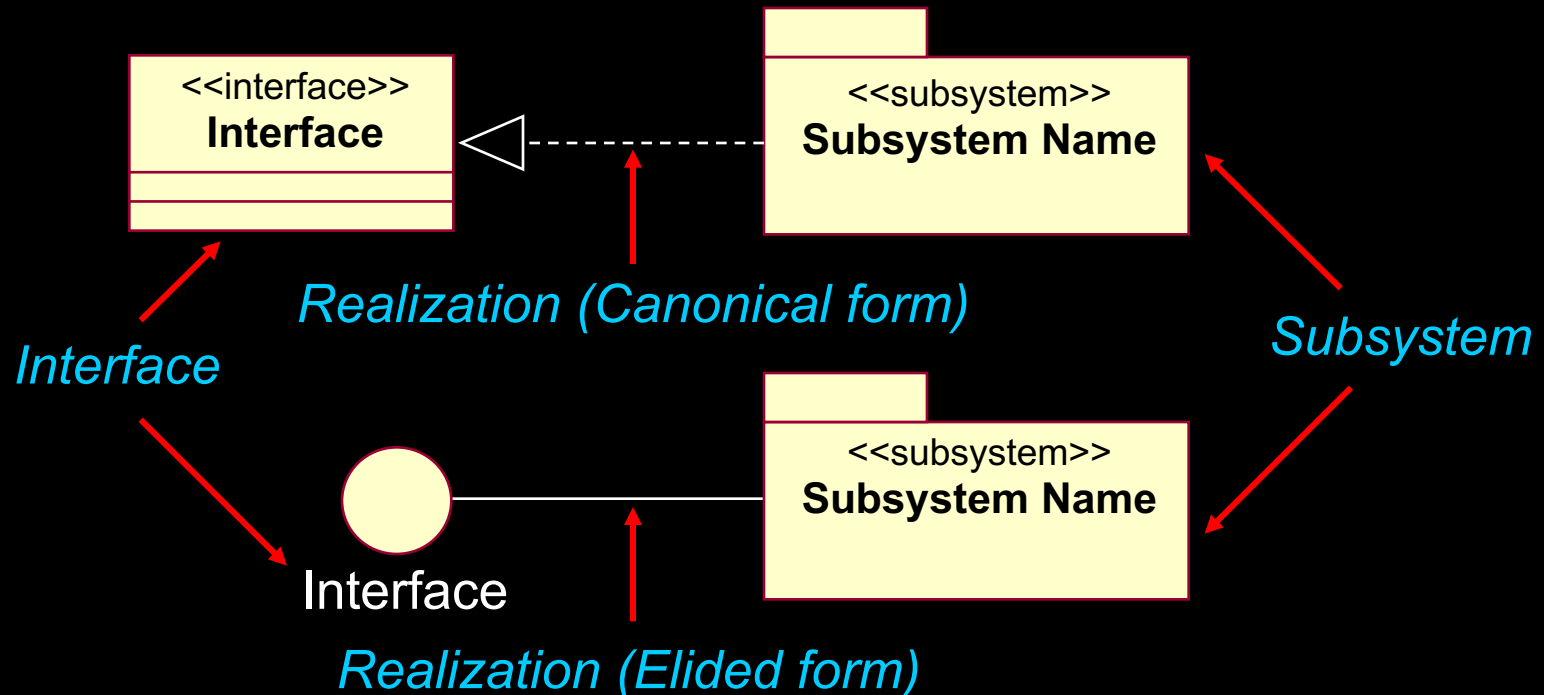


Review: Subsystems and Interfaces

(1.x)

A Subsystem:

- ♦ Is a cross between a package and a class
- ♦ Realizes one or more interfaces that define its behavior

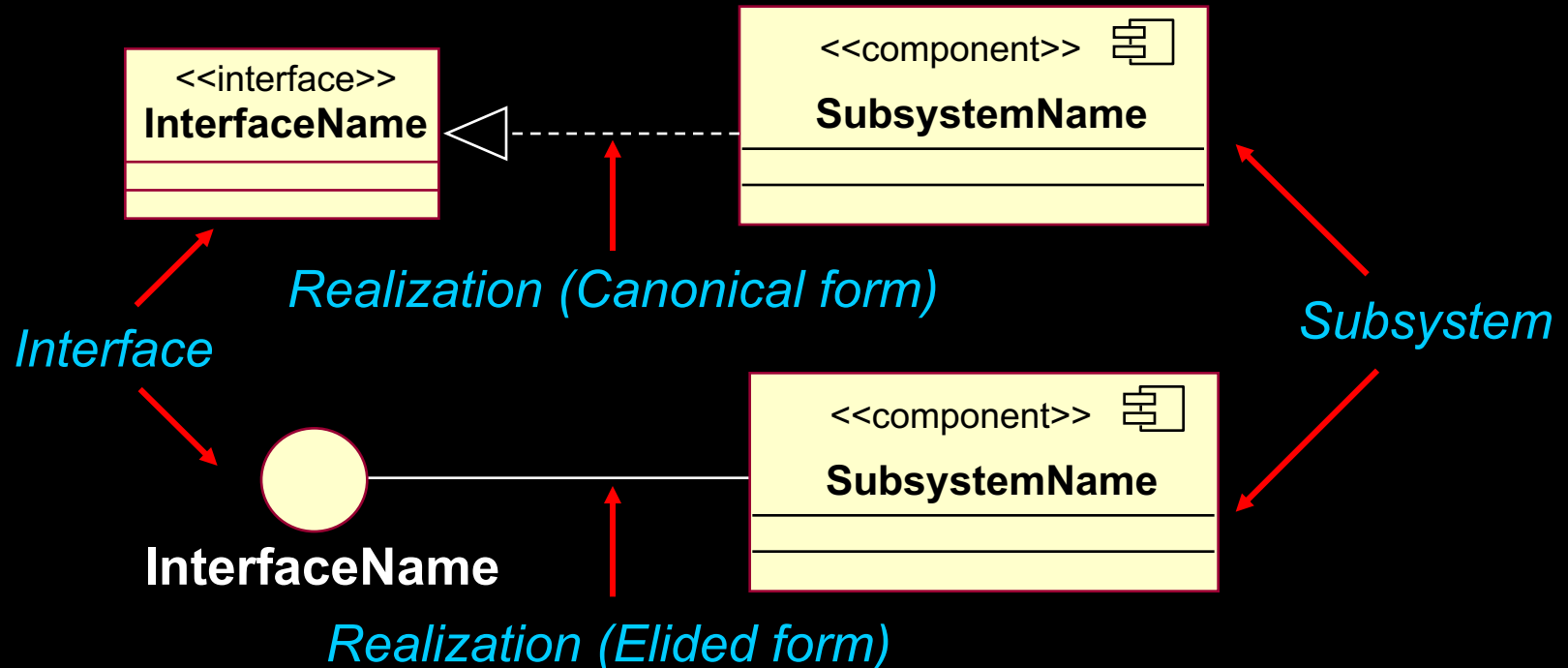


Review: Subsystems and Interfaces

(2.0)

A Subsystem:

- ♦ Is a first class element modeled as a component
- ♦ Realizes one or more interfaces that define its behavior



Subsystem Guidelines

(1.x)

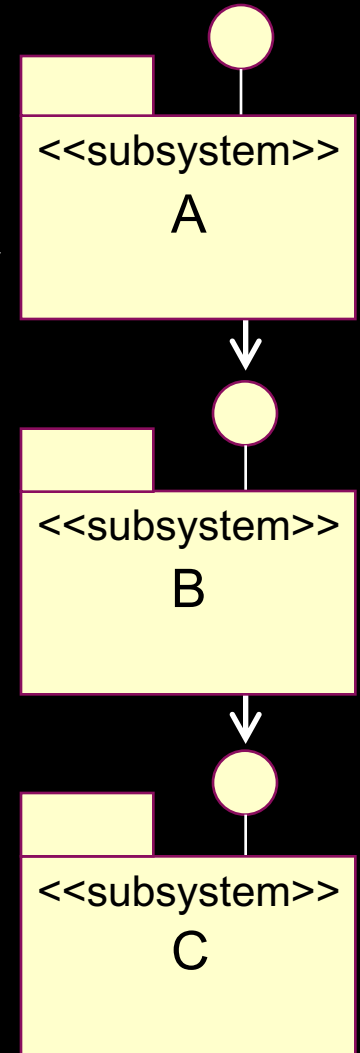
◆ Goals

- Loose coupling
- Portability, plug-and-play compatibility
- Insulation from change
- Independent evolution

◆ Strong Suggestions

- Do not expose details, only interfaces
- Depend only on other interfaces

Key is abstraction and encapsulation



Subsystem Guidelines

(2.0)

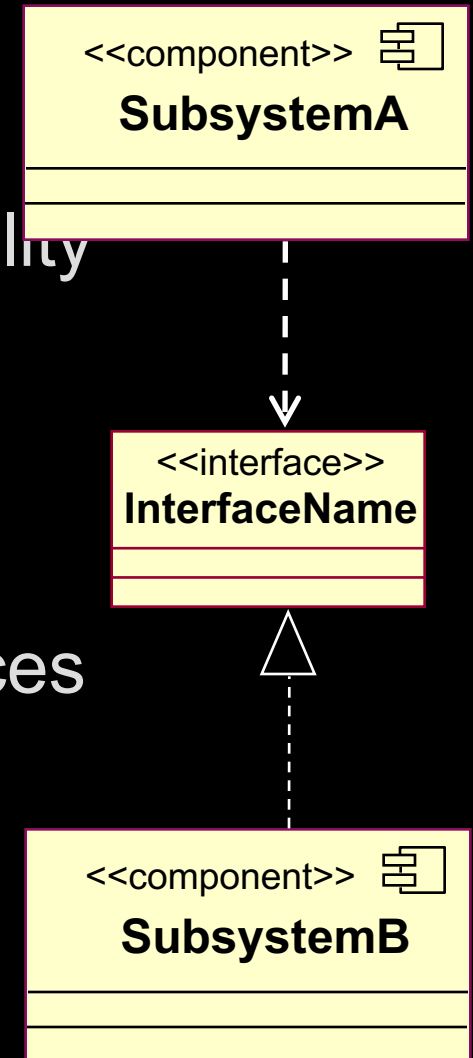
◆ Goals

- Loose coupling
- Portability, plug-and-play compatibility
- Insulation from change
- Independent evolution

◆ Strong Suggestions

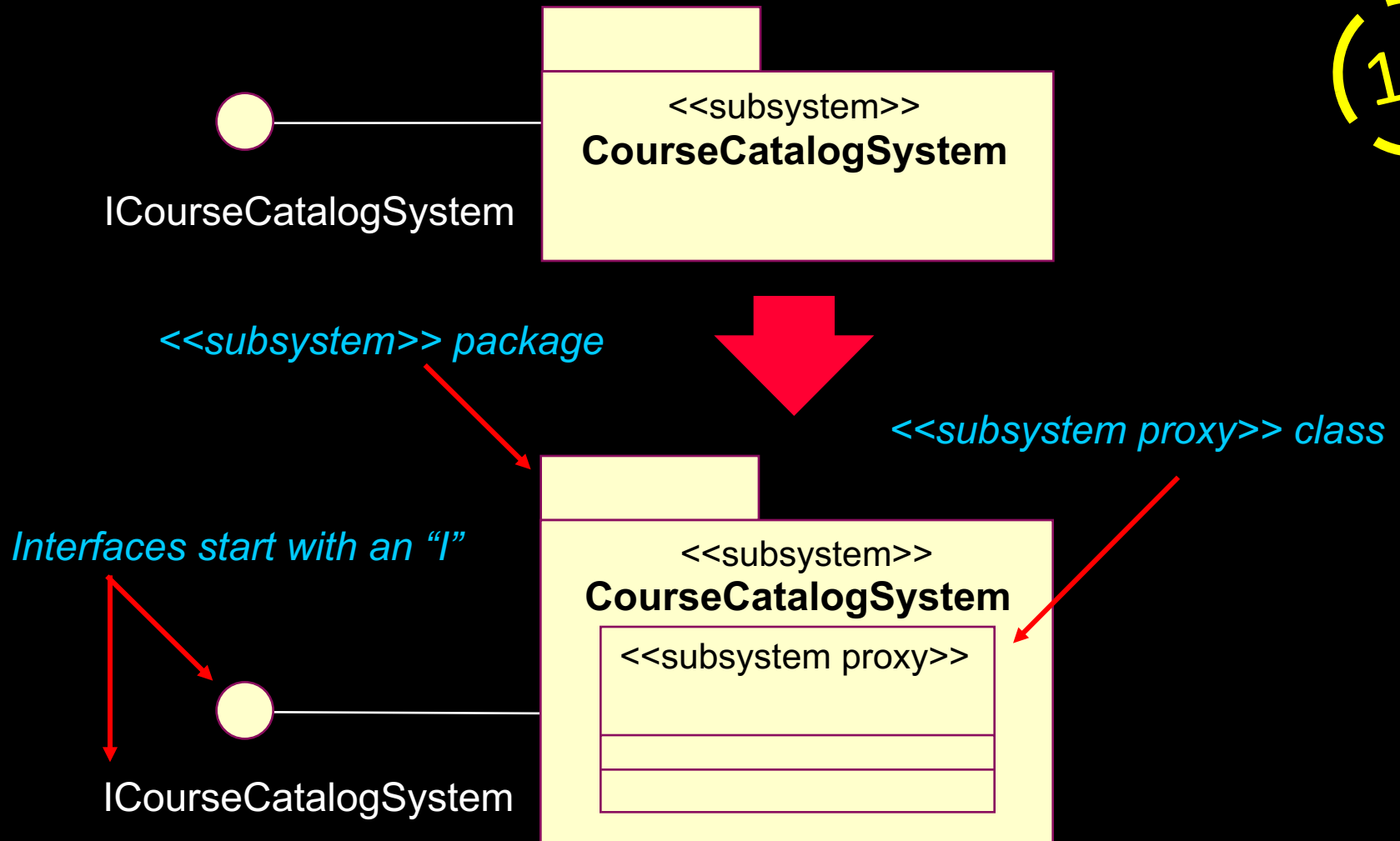
- Do not expose details, only interfaces
- Depend only on other interfaces

Key is abstraction and encapsulation



Review: Modeling Convention for Subsystems and Interfaces

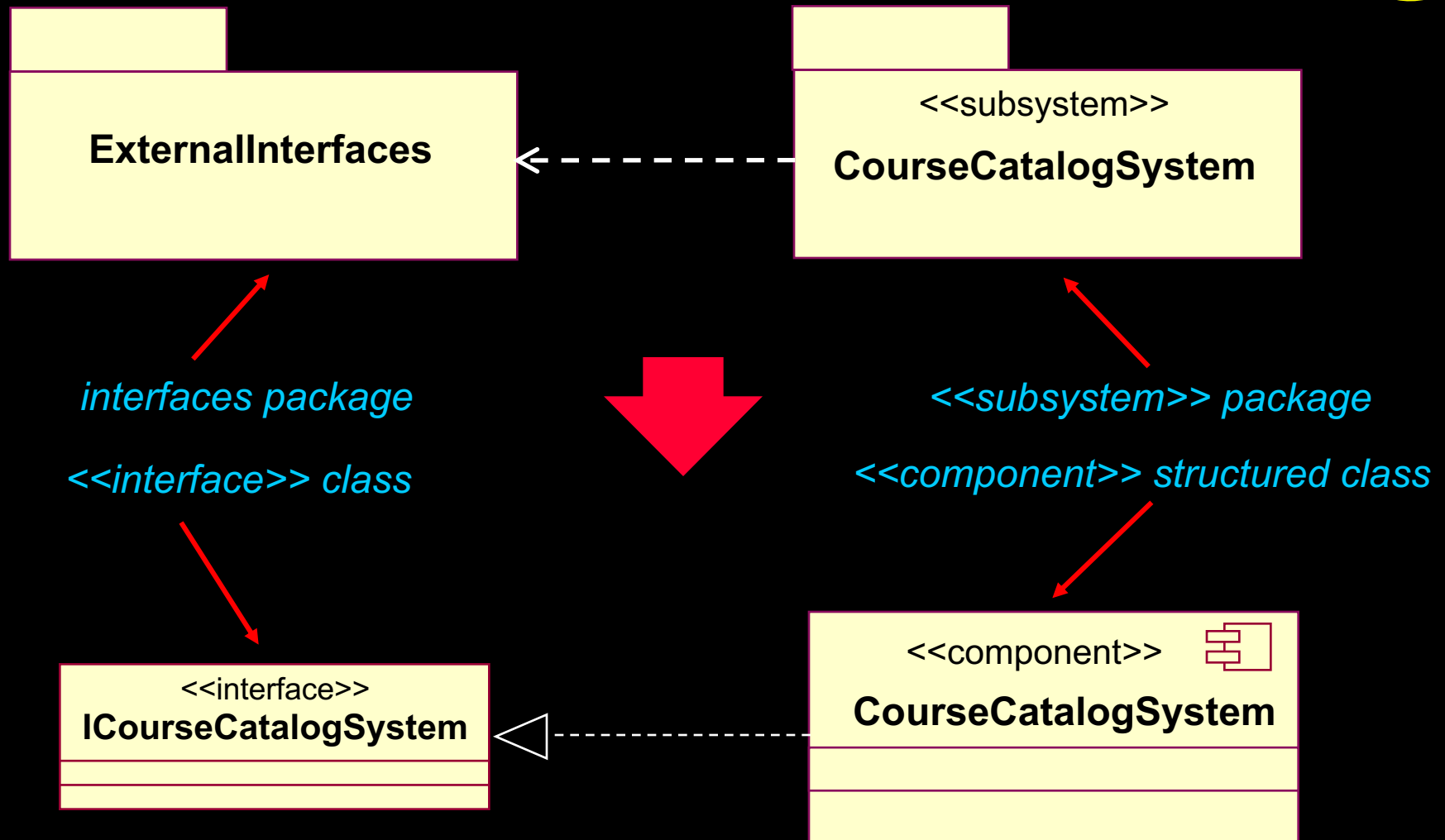
(1.x)



Interfaces are EXTERNAL to the subsystem.

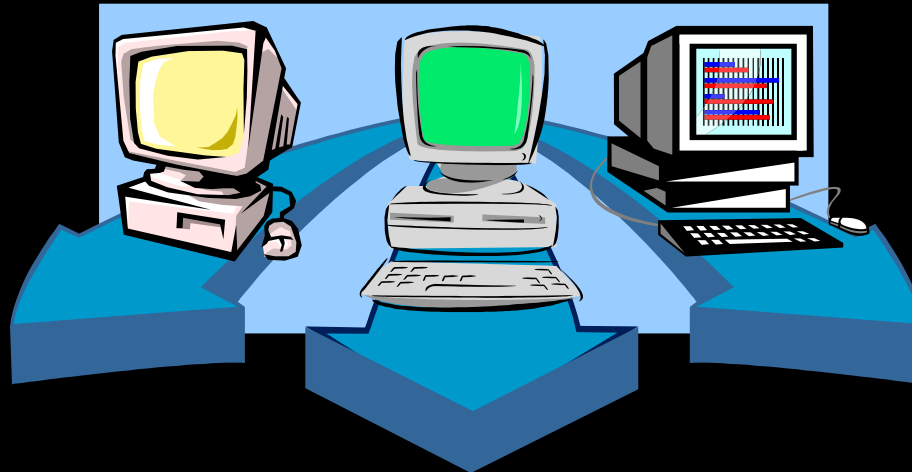
Modeling Convention for Subsystems and Interfaces

(2.0)

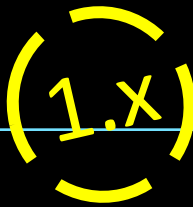


Subsystem Design Steps

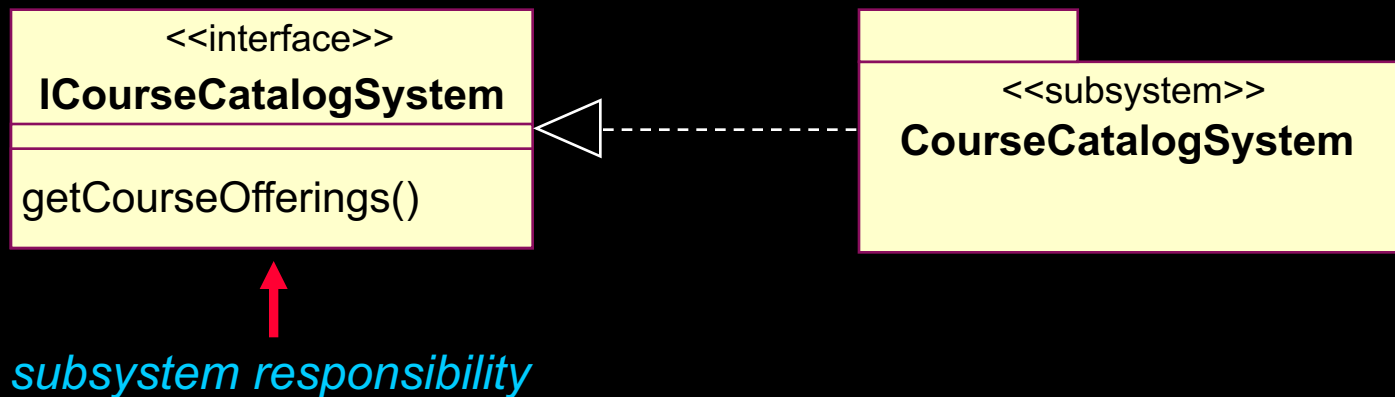
- ★ ♦ Distribute subsystem behavior to subsystem elements
- ♦ Document subsystem elements
- ♦ Describe subsystem dependencies



Subsystem Responsibilities



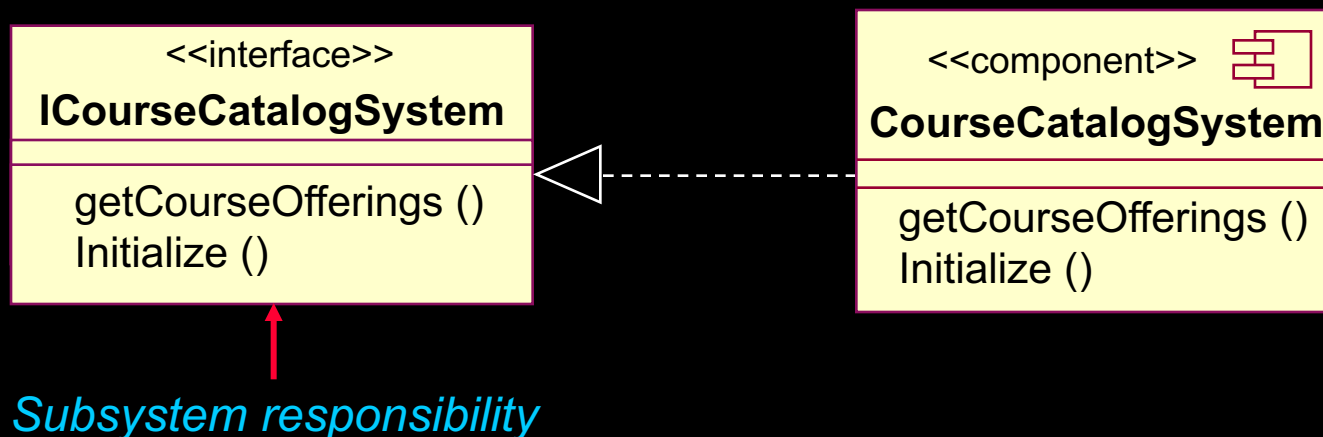
- ◆ Subsystem responsibilities defined by interface operations
 - Model interface realizations
- ◆ Interface operations may be realized by
 - Internal class operations
 - Internal subsystem operations



Subsystem Responsibilities

(2.0)

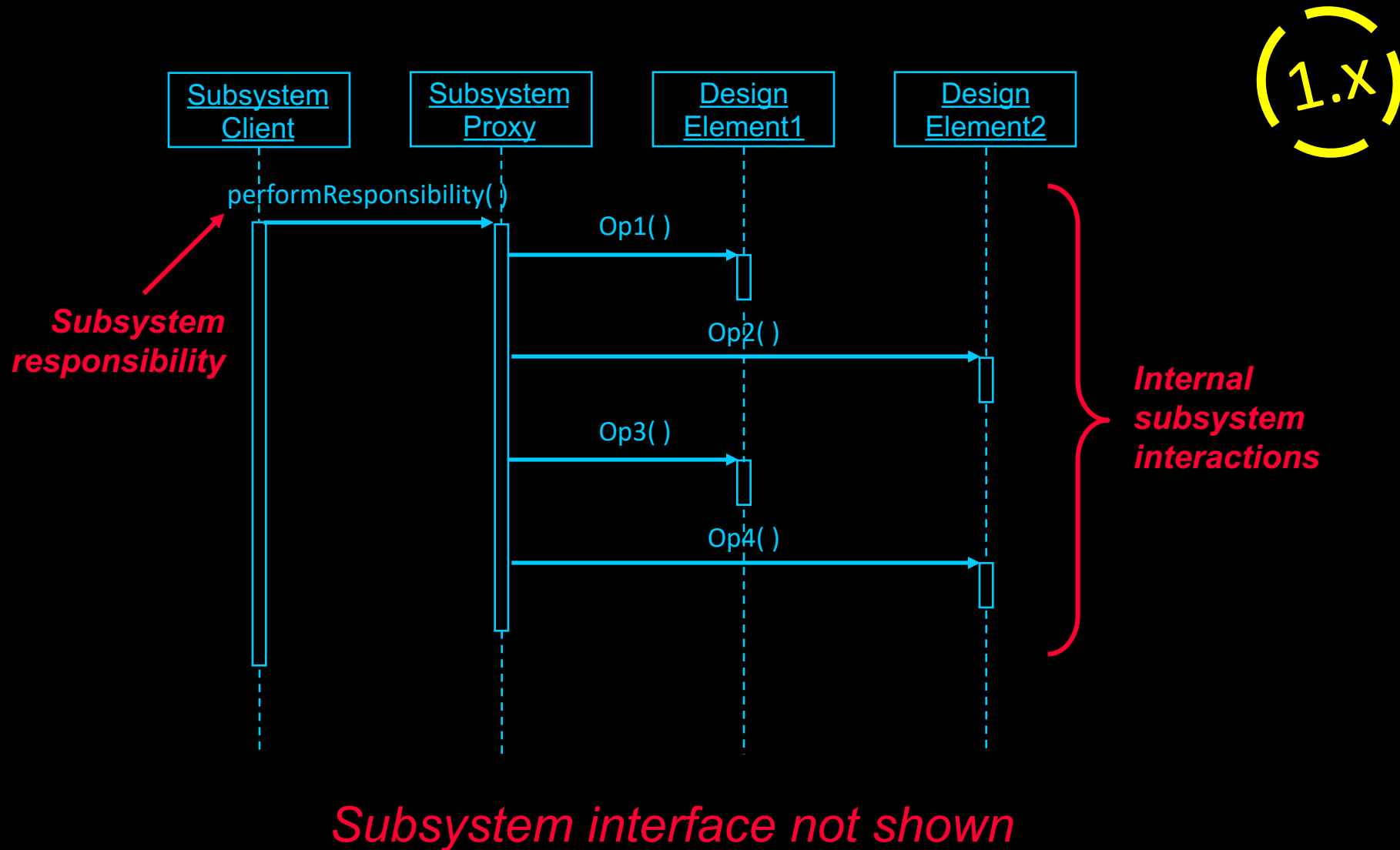
- ◆ Subsystem responsibilities defined by interface operations
 - Model interface realizations
- ◆ Interface operations may be realized by
 - Internal class behavior
 - Subsystem behavior



Distributing Subsystem Responsibilities

- ◆ Identify new, or reuse existing, design elements (for example, classes and subsystems)
- ◆ Allocate subsystem responsibilities to design elements
- ◆ Incorporate applicable mechanisms (for example, persistence, distribution)
- ◆ Document design element collaborations in “interface realizations”
 - One or more interaction diagrams per interface operation
 - Class diagrams containing the required design element relationships
- ◆ Revisit “*Identify Design Elements*”
 - Adjust subsystem boundaries and dependencies, as needed

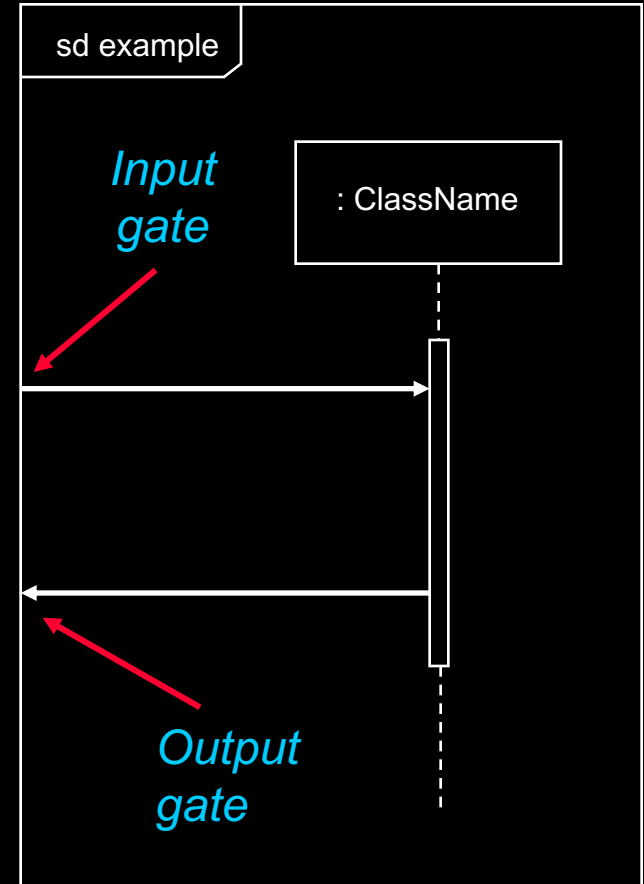
Modeling Convention: Subsystem Interaction Diagrams



What Are Gates?

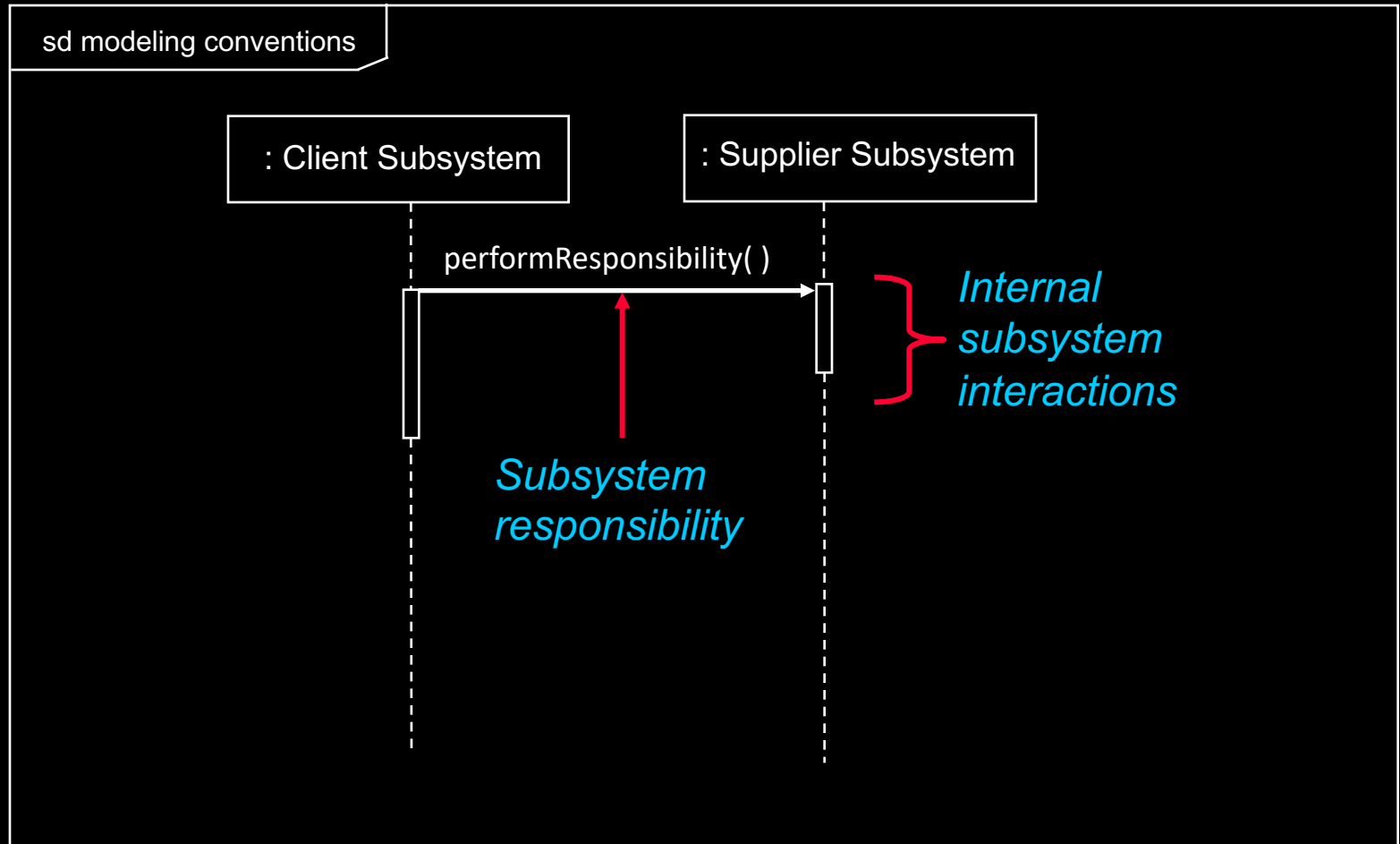
(2.0)

- ◆ A connection point for a message entering or exiting an interaction.
 - A point on the boundary of the sequence diagram
 - The name of the connected message is the name of the gate



Subsystem Interaction Diagrams

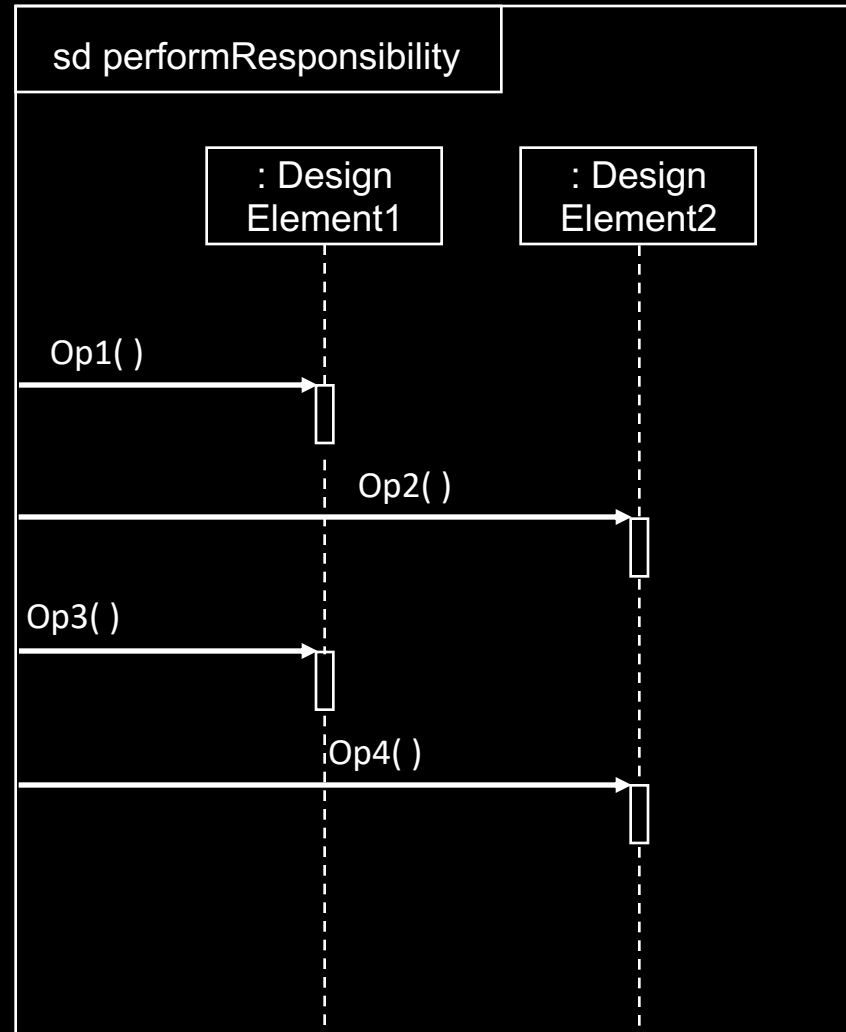
(2.0)



External view of subsystem interactions

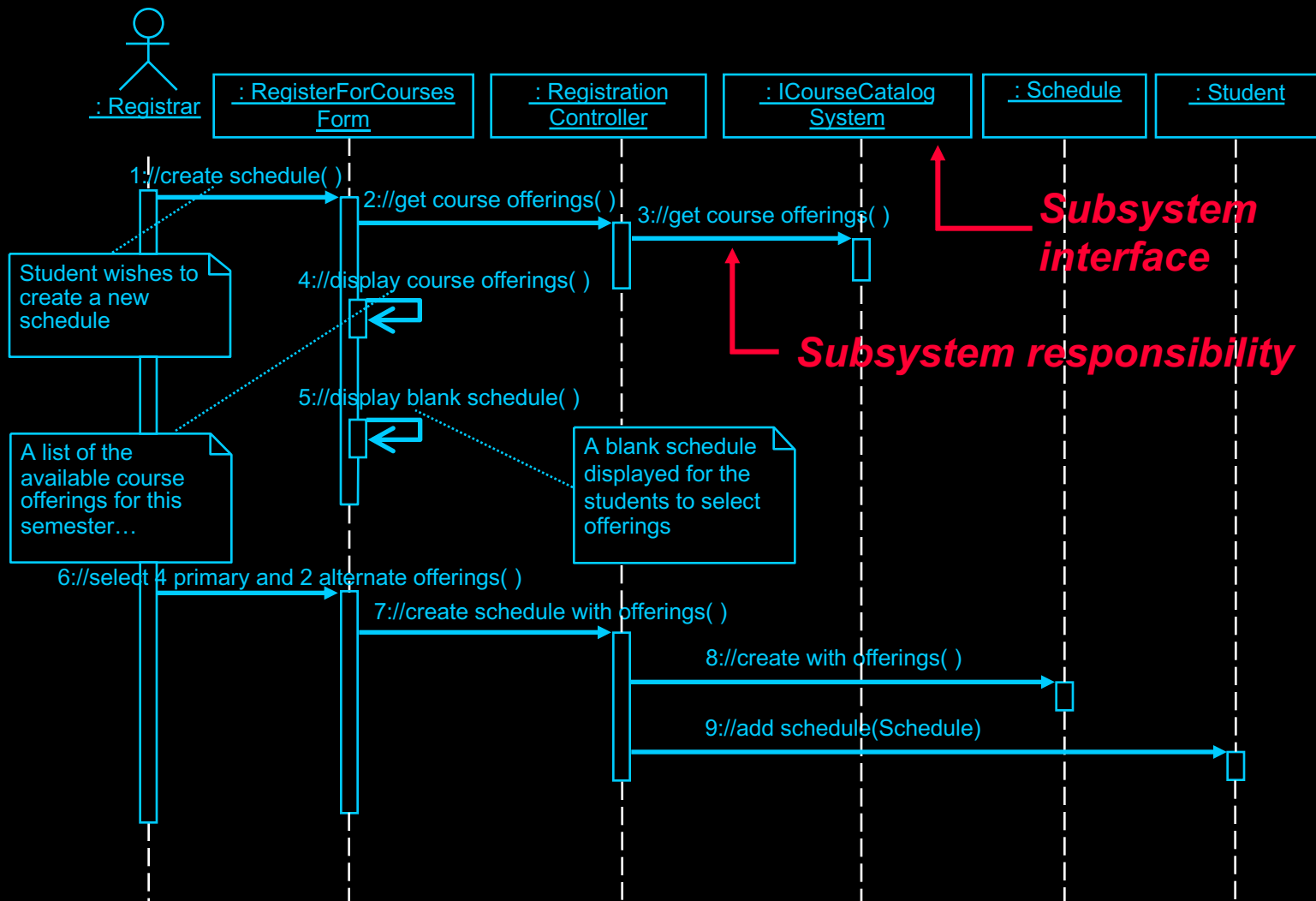
Internal Subsystem Interaction

(2.0)



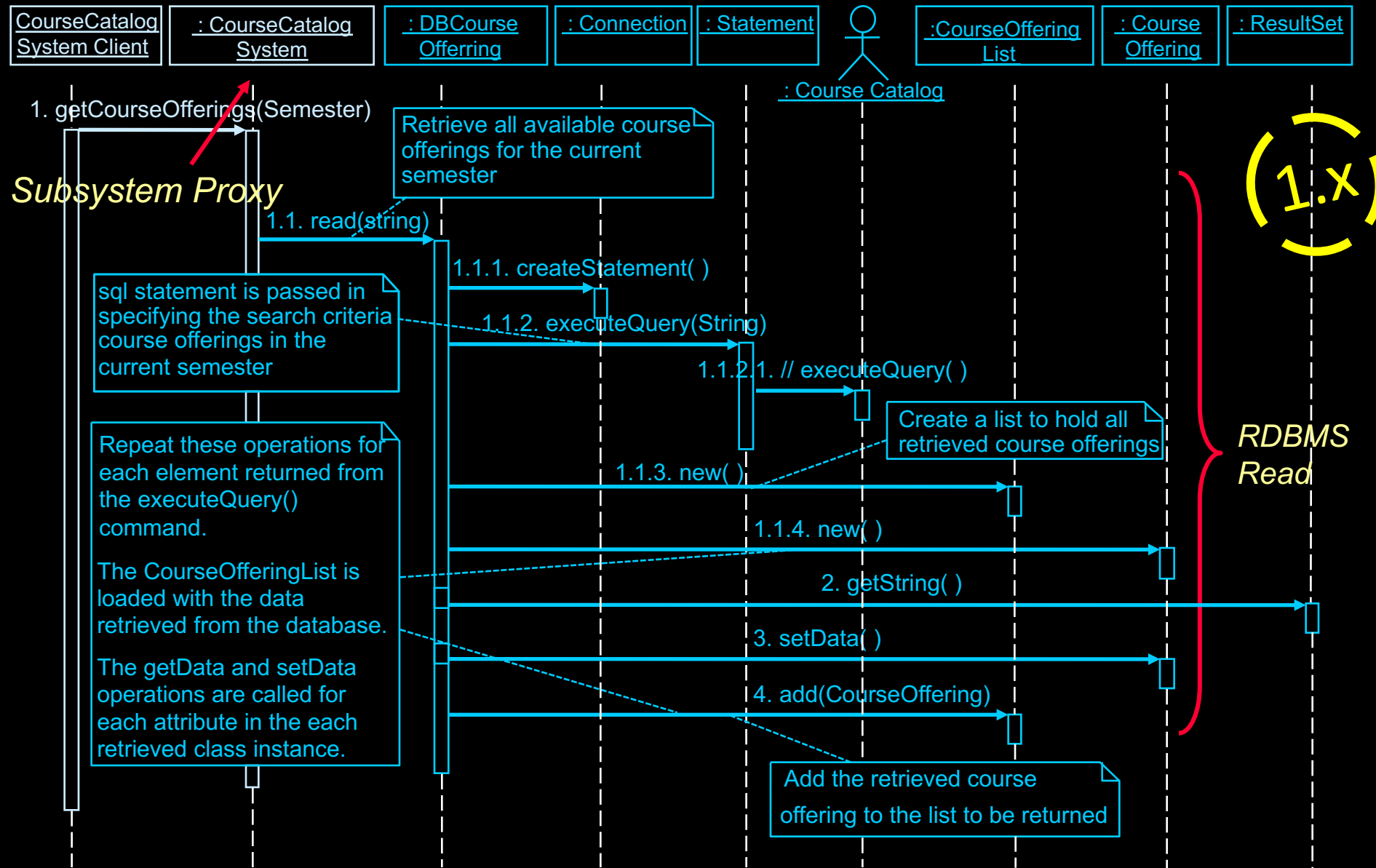
Internal view of Supplier Subsystem

Example: CourseCatalogSystem Subsystem in Context



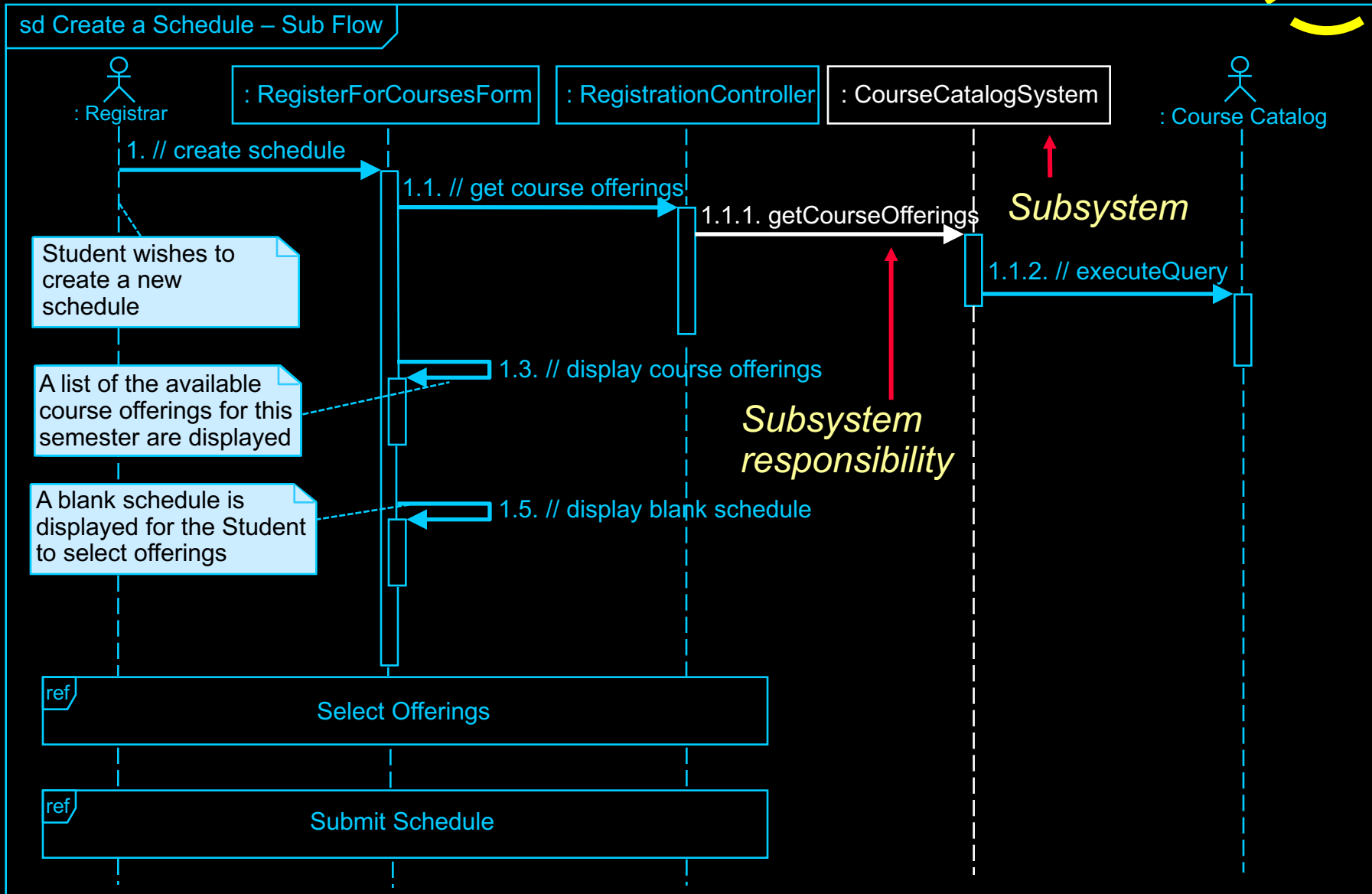
Legacy RDBMS Database Access

Example: Local CourseCatalogSystem Subsystem Interaction



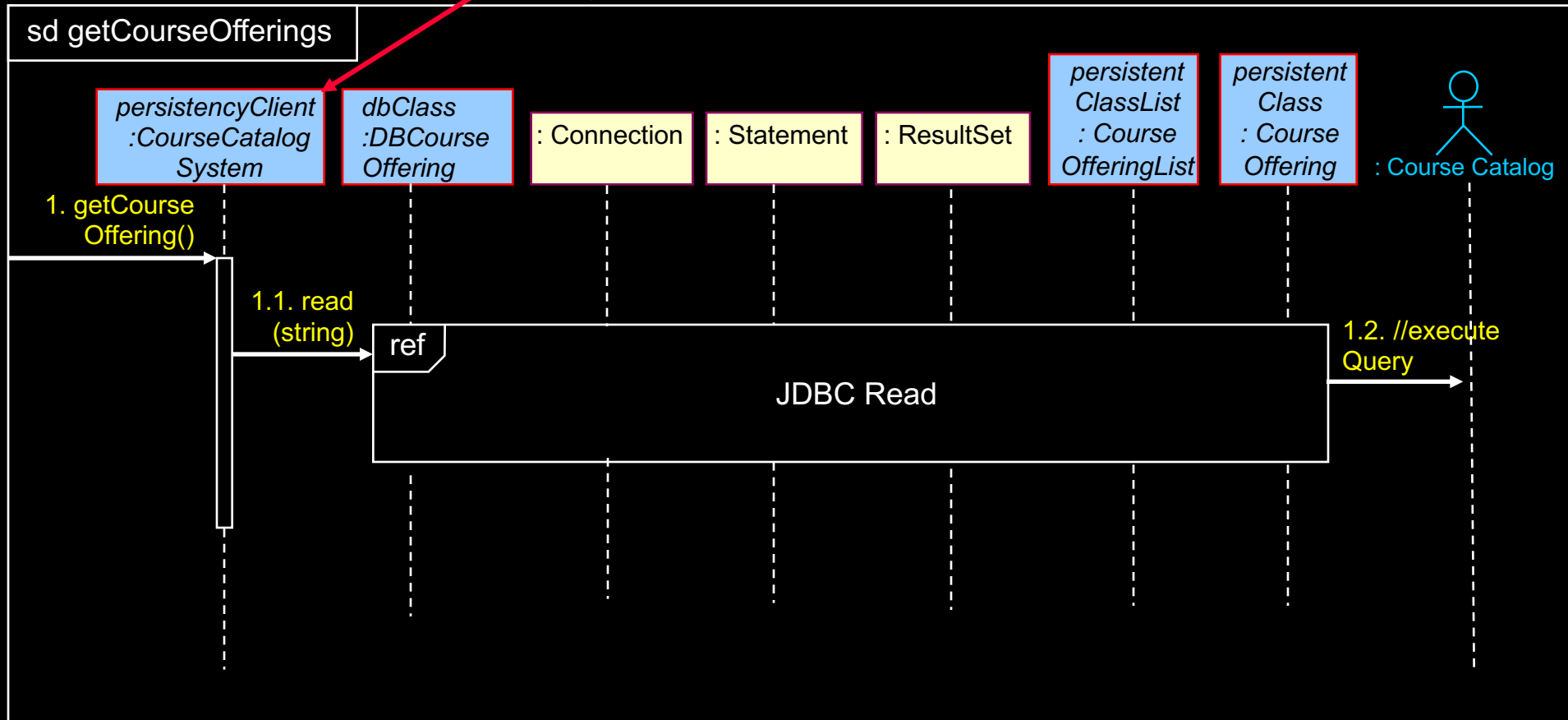
Example: CourseCatalogSystem Subsystem in Context

(2.0)



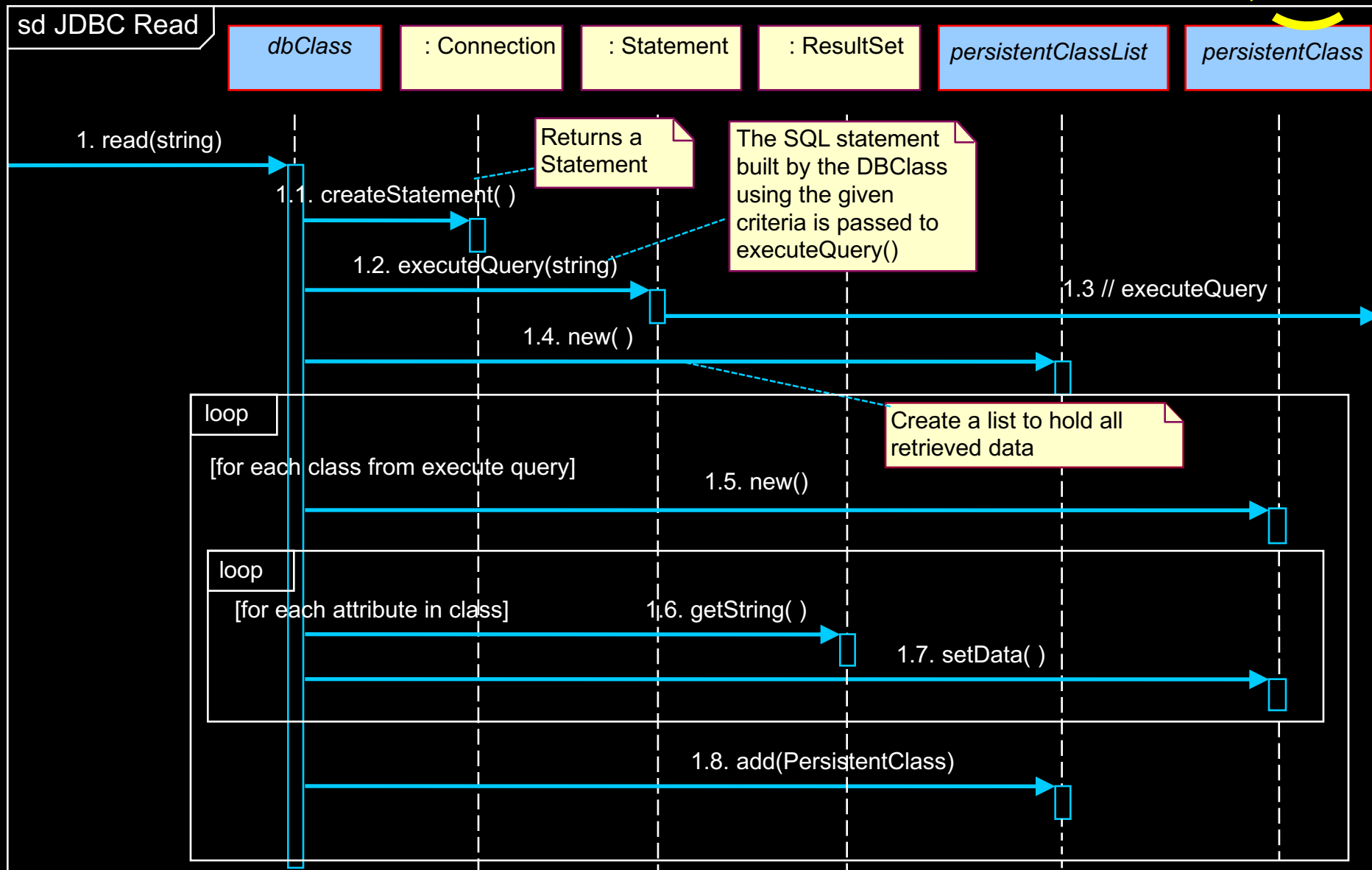
Example: Local CourseCatalogSystem Subsystem Interaction (2.0)

Subsystem



Example: Persistency: RDBMS: JDBC: Read

(2.0)



Subsystem Design Steps

- ◆ Distribute subsystem behavior to subsystem elements

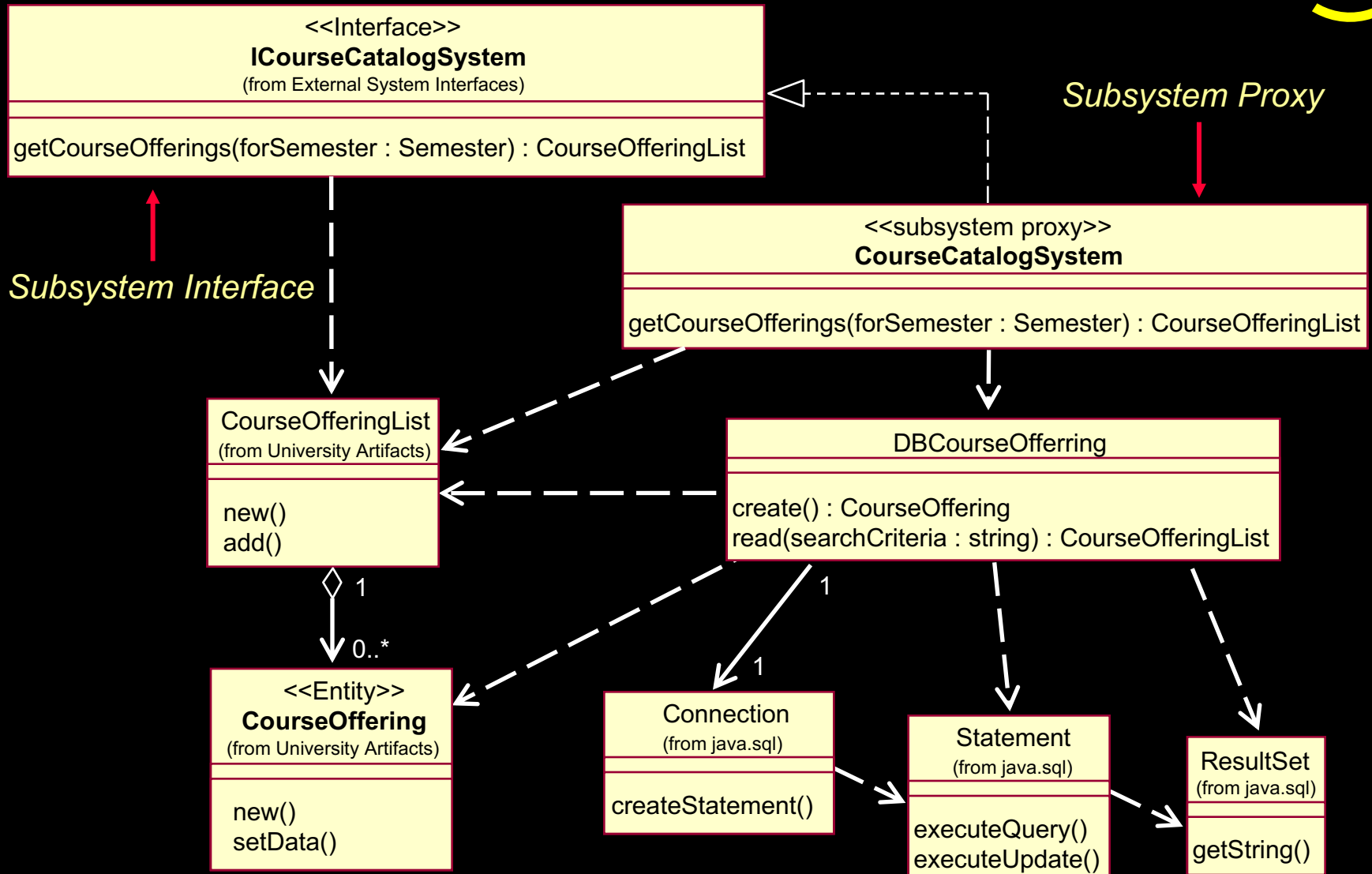
- ★ ◆ Document subsystem elements

- ◆ Describe subsystem dependencies



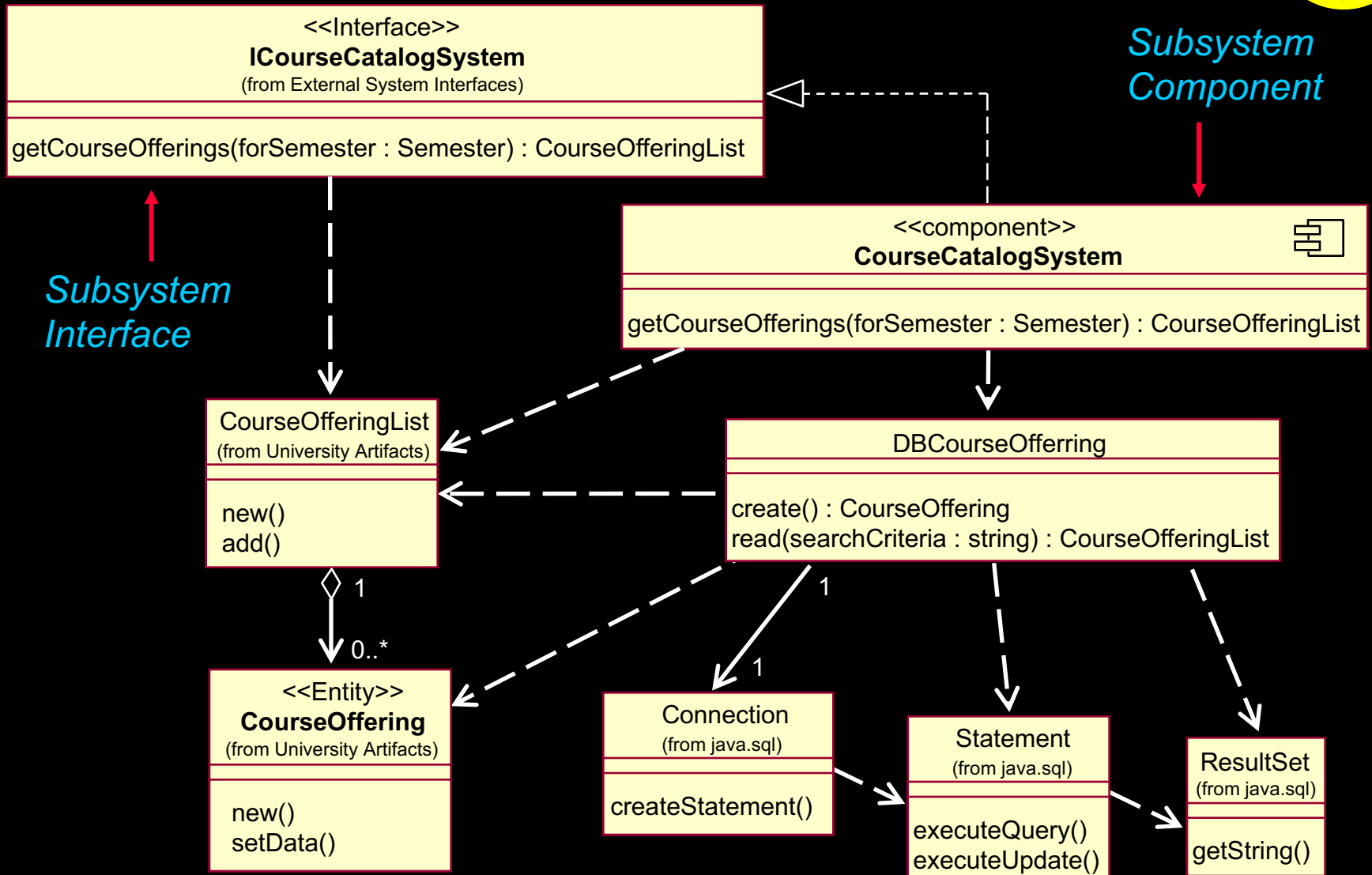
Example: CourseCatalogSystem Subsystem Elements

(1.x)



Example: CourseCatalogSystem Subsystem Elements

(2.0)



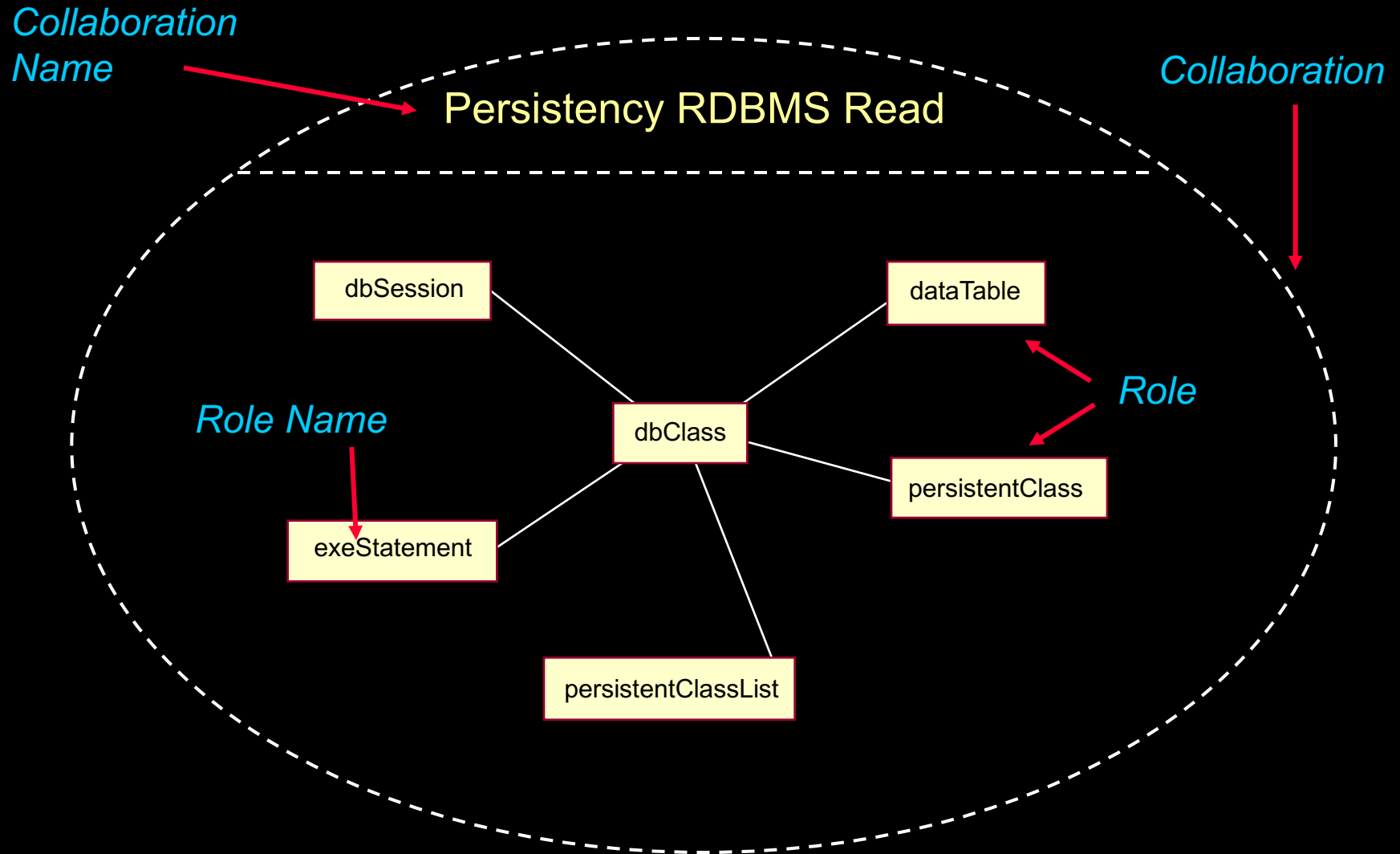
What is a Collaboration?



- ◆ Describes a structure of elements working together to accomplish some desired functionality
 - Typically only incorporates aspects that are deemed relevant to the explanation.
 - Details, such as the identity or precise class of the actual participating instances, are suppressed.
 - Relationships are shown as connectors between the roles which specify communication paths.

Example Collaboration

(2.0)



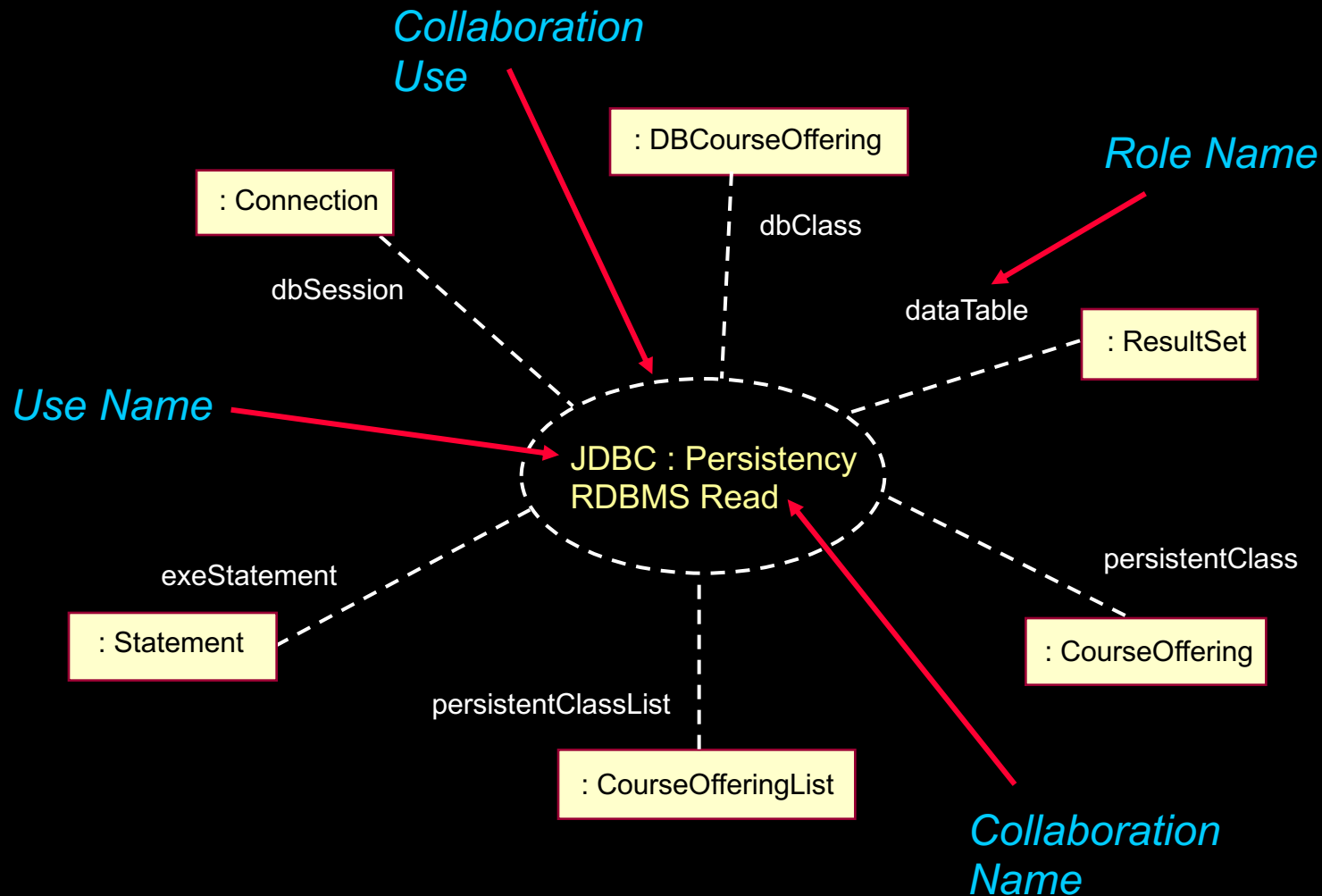
What is a Collaboration Use?



- ◆ Represents the application, or use, of a collaboration within a context
 - A specific situation involving classes or instances fulfilling the roles of the collaboration
- ◆ May appear in the definition of a larger collaboration
 - In this case, roles are bound to roles in the larger collaboration.
 - Roles bound in the larger collaboration are automatically bound to roles of the inner collaboration.

Example Collaboration Use

(2.0)



Review: What Is a Structured Class?

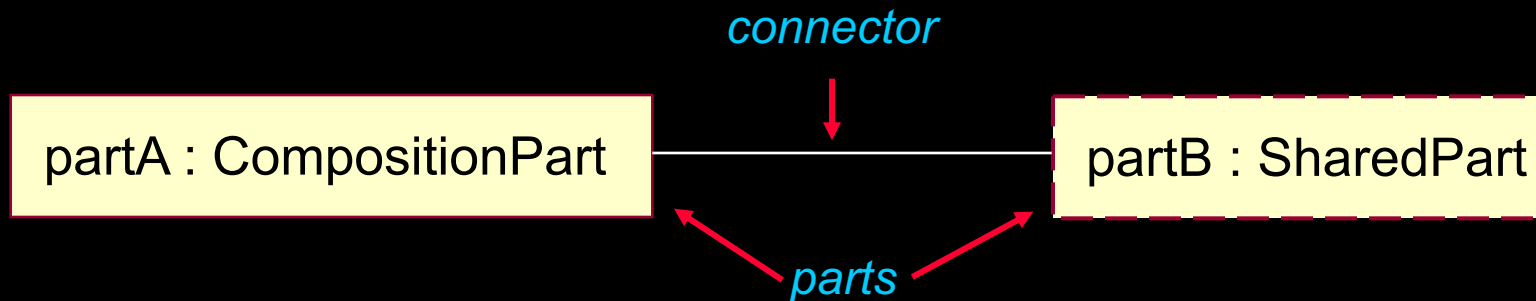
(2.0)

- ♦ A structured class contains parts or roles that form its structure and realize its behavior.
 - Describes the internal implementation structure
- ♦ The parts themselves may also be structured classes.
 - Allows hierarchical structure to permit a clear expression of multilevel models.
- ♦ A connector is used to represent an association in a particular context.
 - Represents communications paths among parts

Structured Class Notation

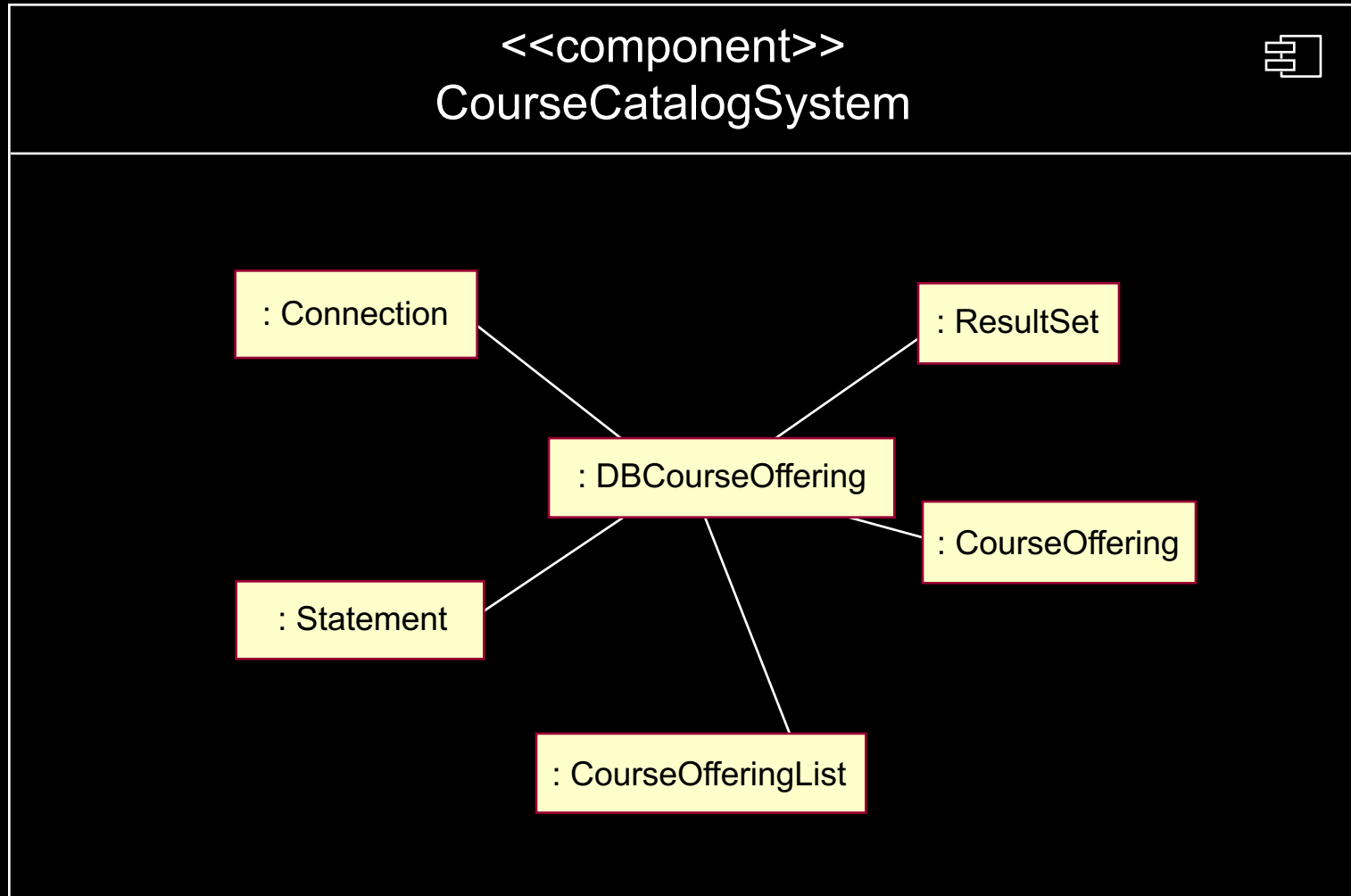
(2.0)

- ♦ A part or role is shown by using the symbol for a class (a rectangle)
 - A composite aggregation represents an owned part and is represented by a solid rectangle.
 - A shared aggregation represents an external part (one not owned by the enclosing whole) and is represented by a dashed rectangle.



Example: Composite Structure Diagram

(2.0)



What Is a Port?

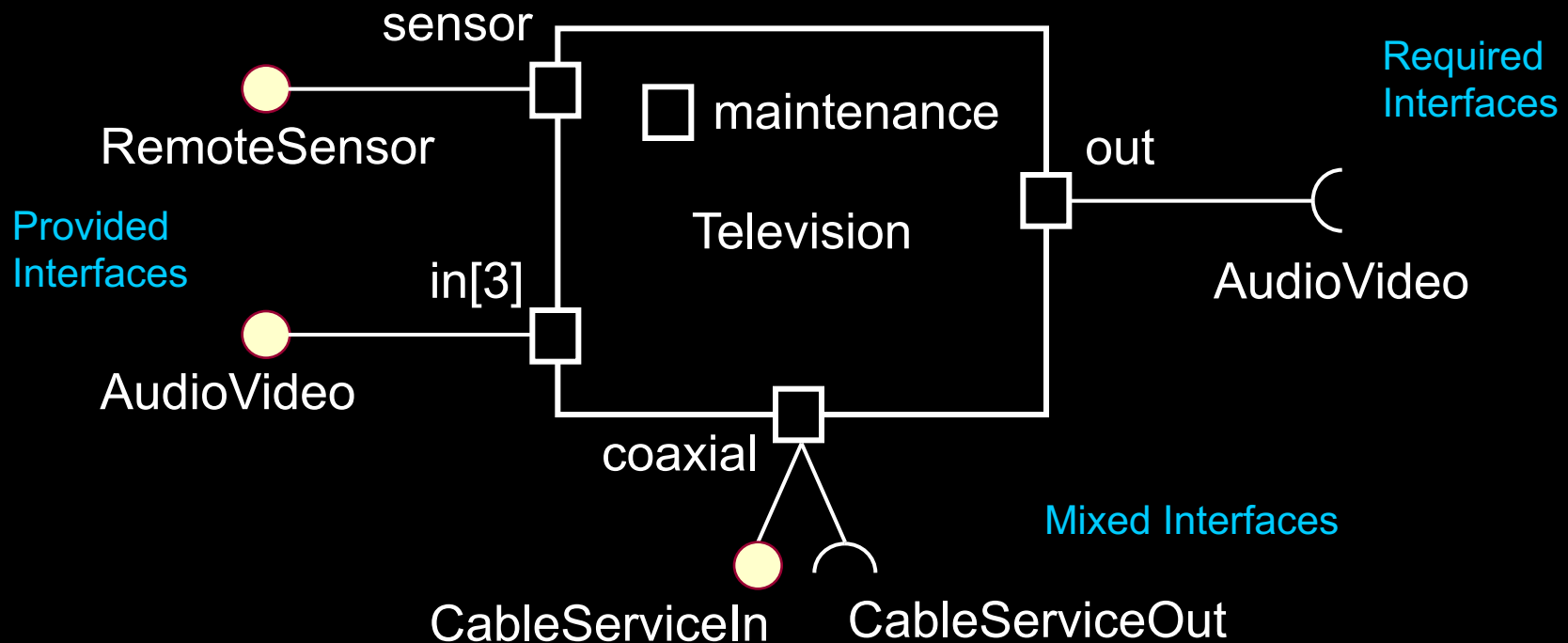


- ◆ A port is a structural feature that encapsulates the interaction between the contents of a class and its environment.
 - Port behavior is specified by its provided and required interfaces.
- ◆ A port permits the internal structure to be modified without affecting external clients.
 - External clients have no visibility to internals.
- ◆ A class may have a number of ports.
 - Each port has a set of provided and required interfaces.

Port Notation

(2.0)

- ♦ A public port is shown as a small square *straddling* the boundary of a classifier.
- ♦ A private port is shown as a small square *inside* the boundary of a classifier.



- ◆ Ports can have different implementation types:
 - Service Port - Is only used for the internal implementation of the class
 - Behavior Port - Requests on the port are implemented directly by the class
 - Delegation Port – Requests on the port are transmitted to internal parts for implementation

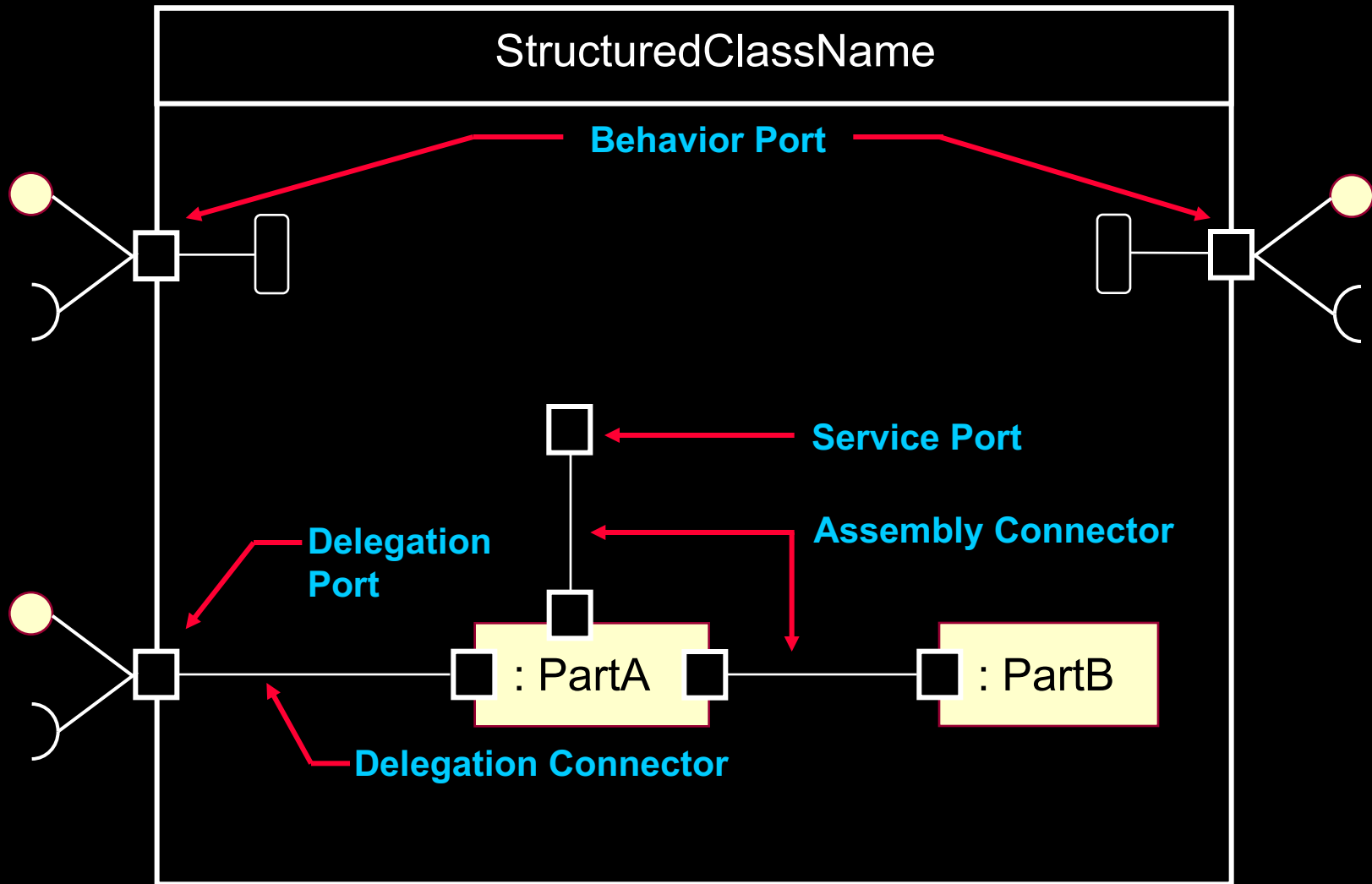
What Is a Connector?

(2.0)

- ◆ A connector models the communication link between interconnected parts. For example:
 - Assembly connectors - Reside between two elements (parts or ports) in the internal implementation specification of a structured class.
 - Delegation connectors - Reside between a delegation port and an internal part in the internal implementation specification of a structured class.

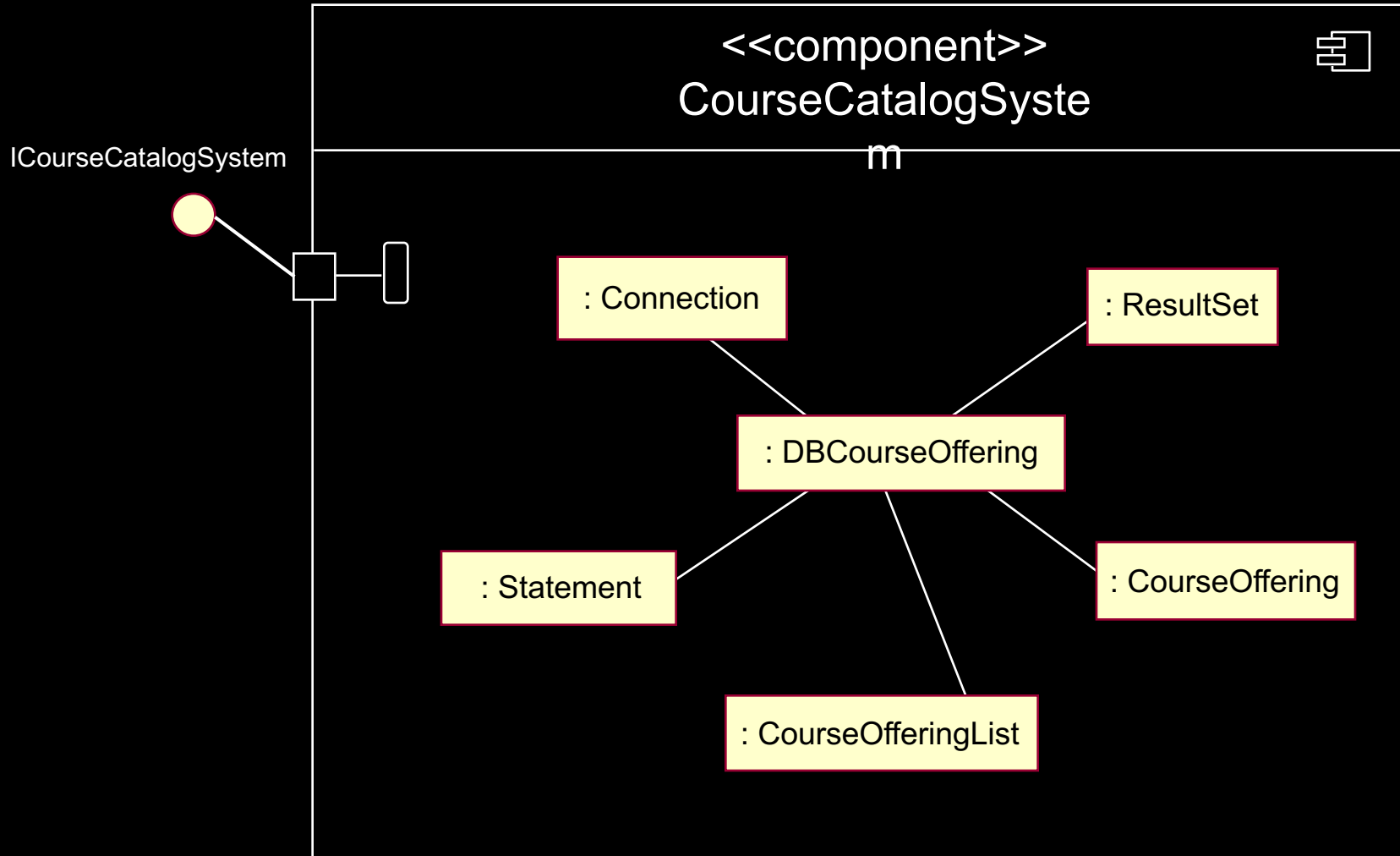
Example: Composite Structure Diagram with Ports

(2.0)



Example Composite Structure Diagram

(2.0)



Subsystem Design Steps

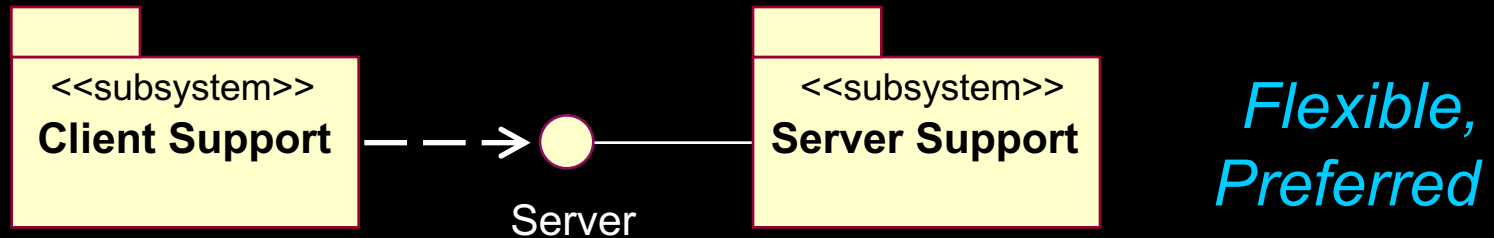
- ◆ Distribute subsystem behavior to subsystem elements
- ◆ Document subsystem elements
- ★ ◆ Describe subsystem dependencies



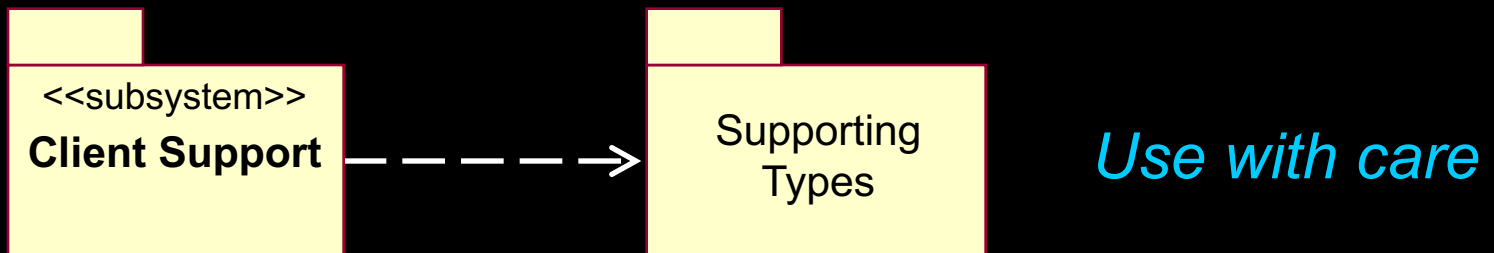
Subsystem Dependencies: Guidelines

(1.x)

◆ Subsystem dependency on a subsystem



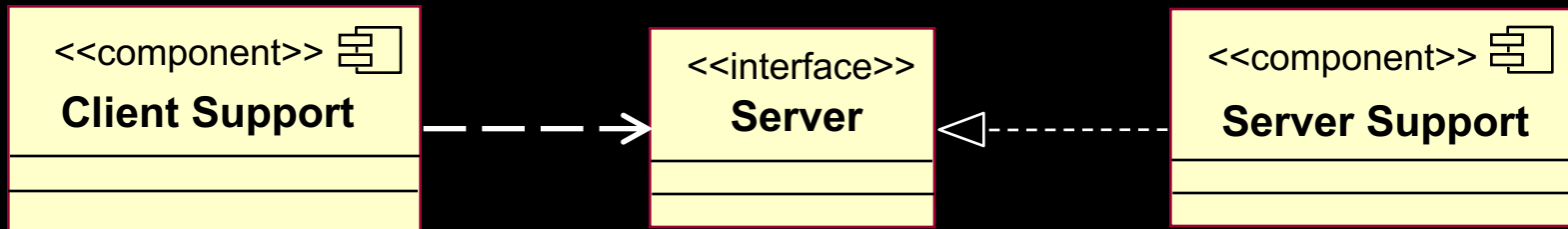
◆ Subsystem dependency on a package



Subsystem Dependencies: Guidelines

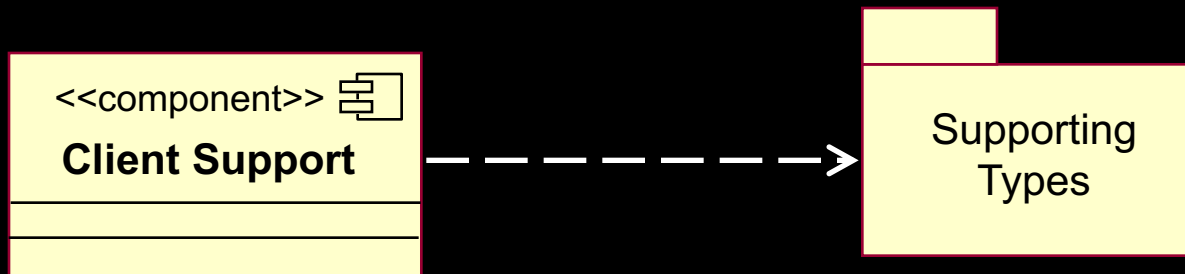
(2.0)

◆ Subsystem dependency on a subsystem



Flexible, Preferred

◆ Subsystem dependency on a package



Use with care

Review: Subsystem Design

- ◆ What are gates?
- ◆ What is the difference between a collaboration and a collaboration use?
- ◆ What is a port and name the different types.
- ◆ Why should dependencies be to the subsystem's interface?



Exercise: Subsystem Design

- ◆ Given the following:
 - The defined subsystems, their interfaces and their relationships with other design elements (the subsystem context diagrams)
 - Payroll Exercise Solution: Identify Design Elements, Subsystem Context Diagrams section



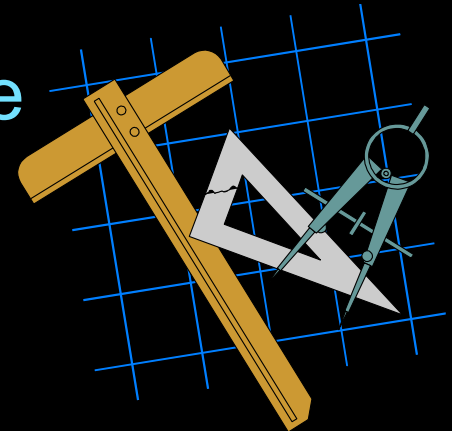
Exercise: Subsystem Design (continued)

- ◆ Identify the following for a particular subsystems:
 - The design elements contained within the subsystem and their relationships
 - The interactions needed to implement the subsystem interface operations



Exercise: Subsystem Design (continued)

- ◆ Produce the following for a particular subsystem(s):
 - “Interface realizations”
 - Interaction diagram for each interface operation
 - Composite structure diagram containing the subsystem design elements that realize the interface responsibilities and their communication paths



Exercise: Review

- ♦ Compare your Subsystem Interface Realizations
 - ♦ Have all the main and subflows for the interface operations been handled?
 - ♦ Has all behavior been distributed among the participating design elements?
 - ♦ Has behavior been distributed to the right design elements?
 - ♦ Are there any messages coming from the interfaces?

