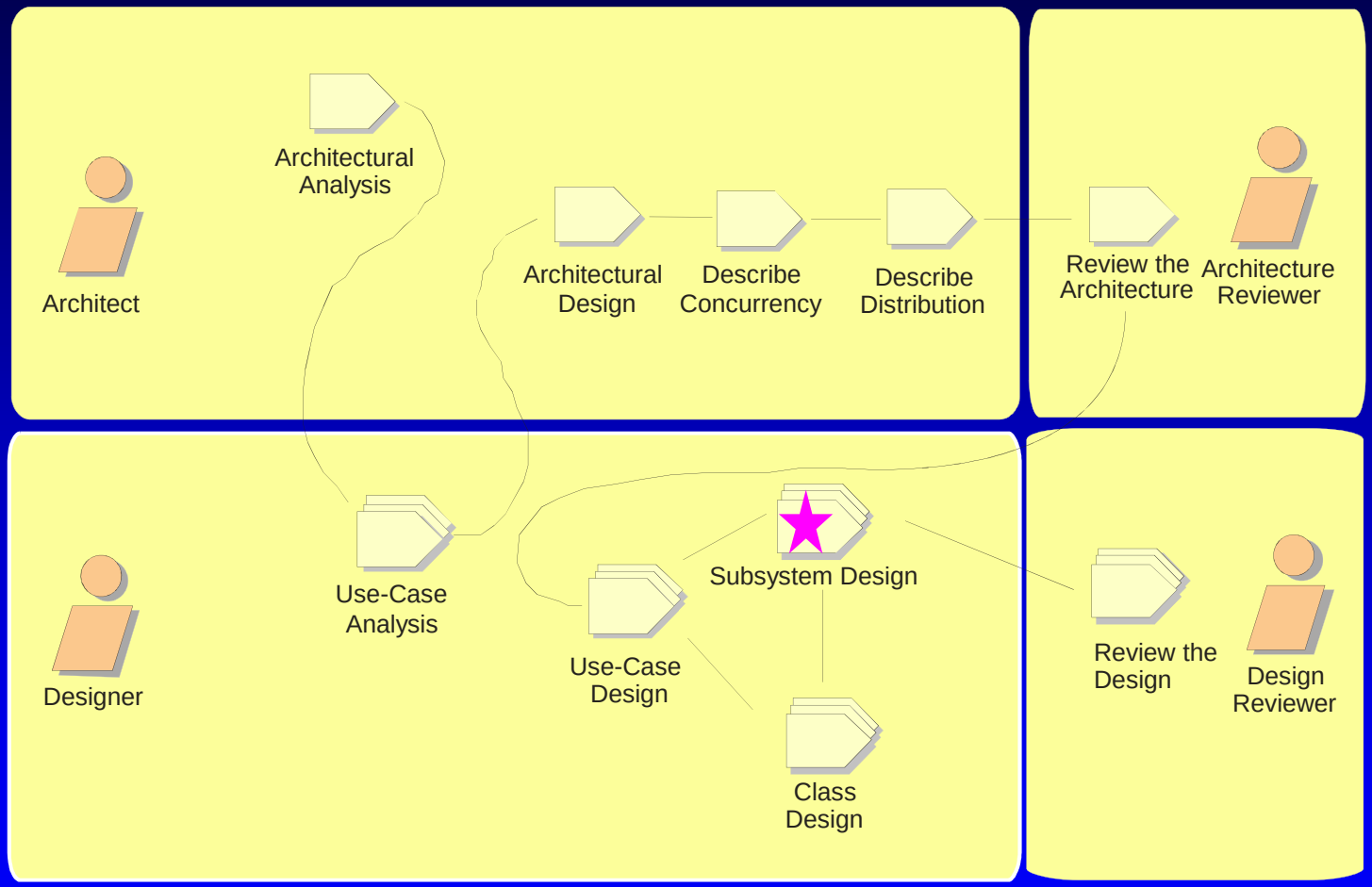
Subsystem Design

FROM Dr. Giuseppe Calavaro, Ratiolal®
TO Students in the DISP, University of Roma “Tor Vergata”
2003

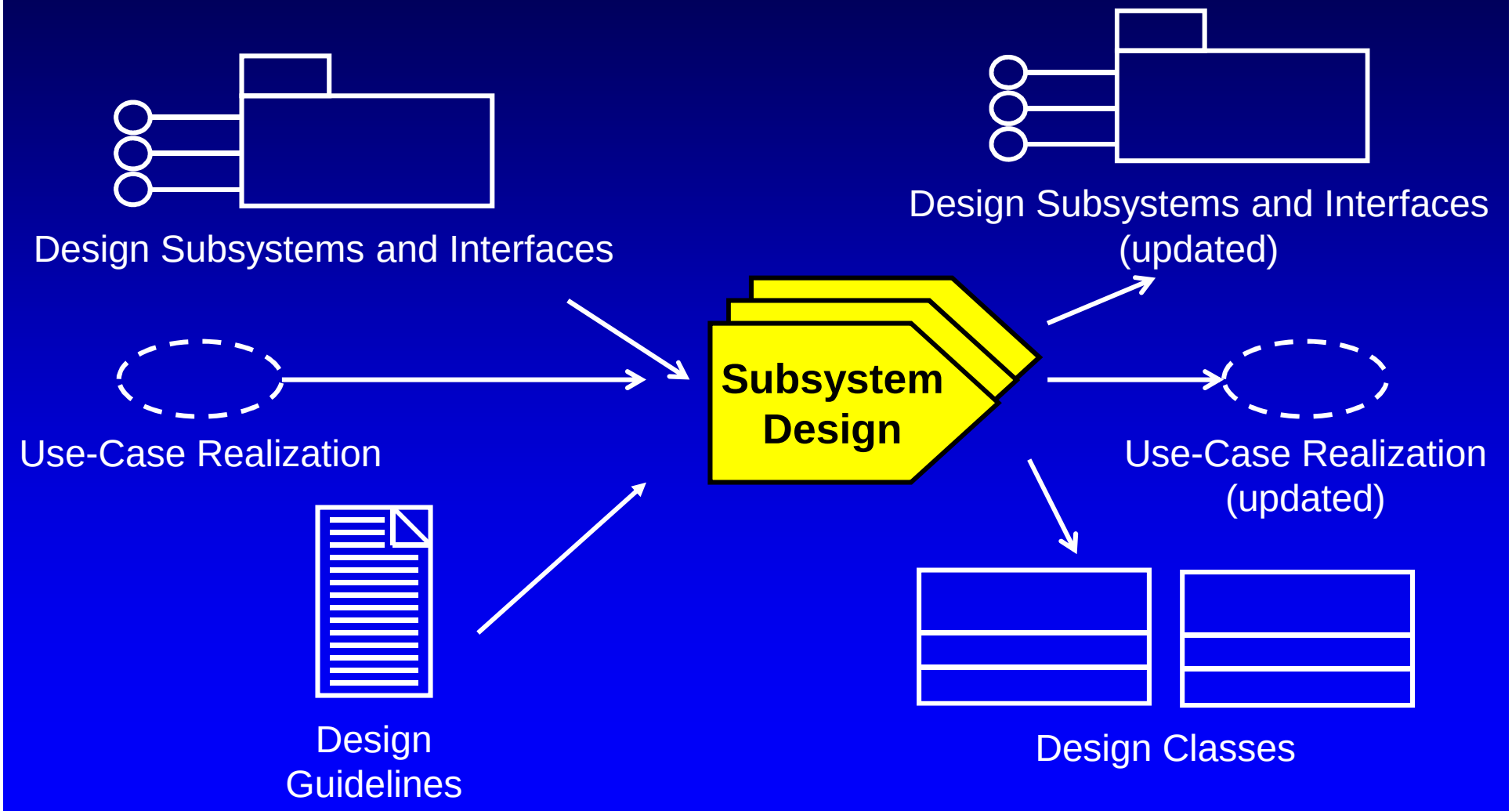
Objectives: Subsystem Design

- Understand the purpose of Subsystem Design and where in the lifecycle it is performed
- Define the behaviors specified in the subsystem's interfaces in terms of collaborations of contained classes
- Document the internal structure of the subsystem
- Determine the dependencies upon elements external to the subsystem

Subsystem Design in Context

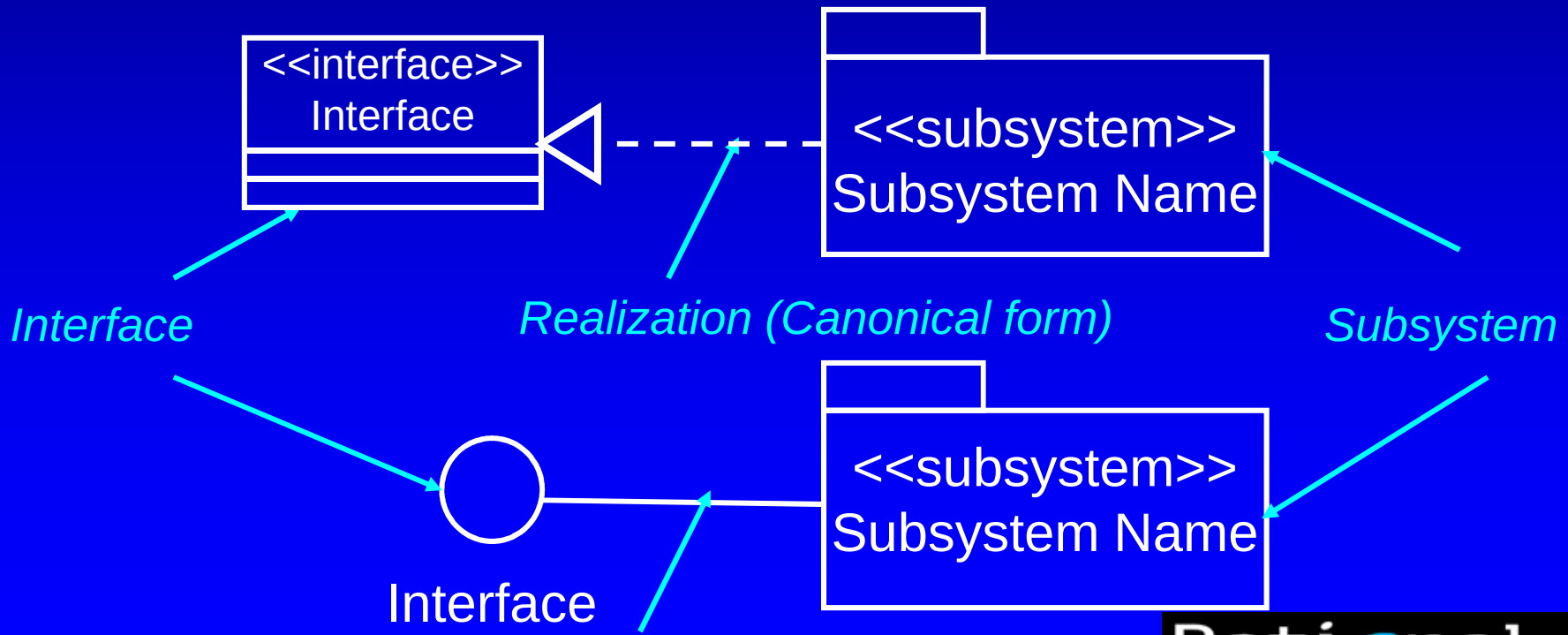


Subsystem Design Overview



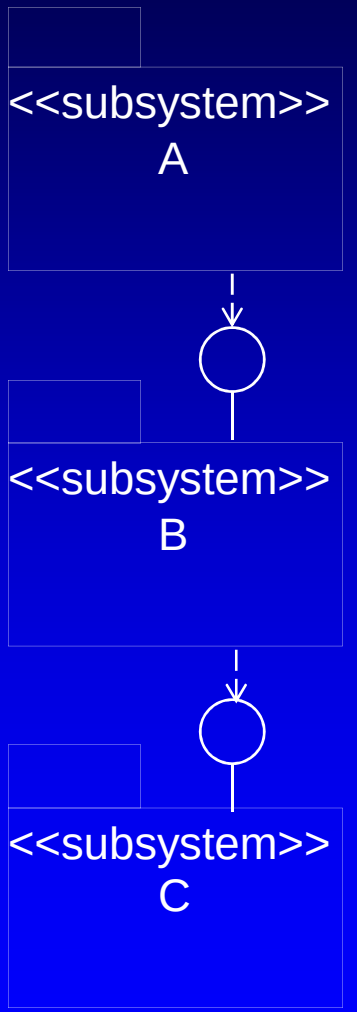
Review: Subsystems and Interfaces

- A “cross between” a package and a class
- Realizes one or more interfaces which define its behavior



Subsystem Guidelines

- Goals
  - Loose coupling
  - Portability, plug-and-play compatibility
  - Insulation from change
  - Independent evolution
- Strong Suggestions
  - Don’t expose details, only interfaces
  - Only depend on other interfaces

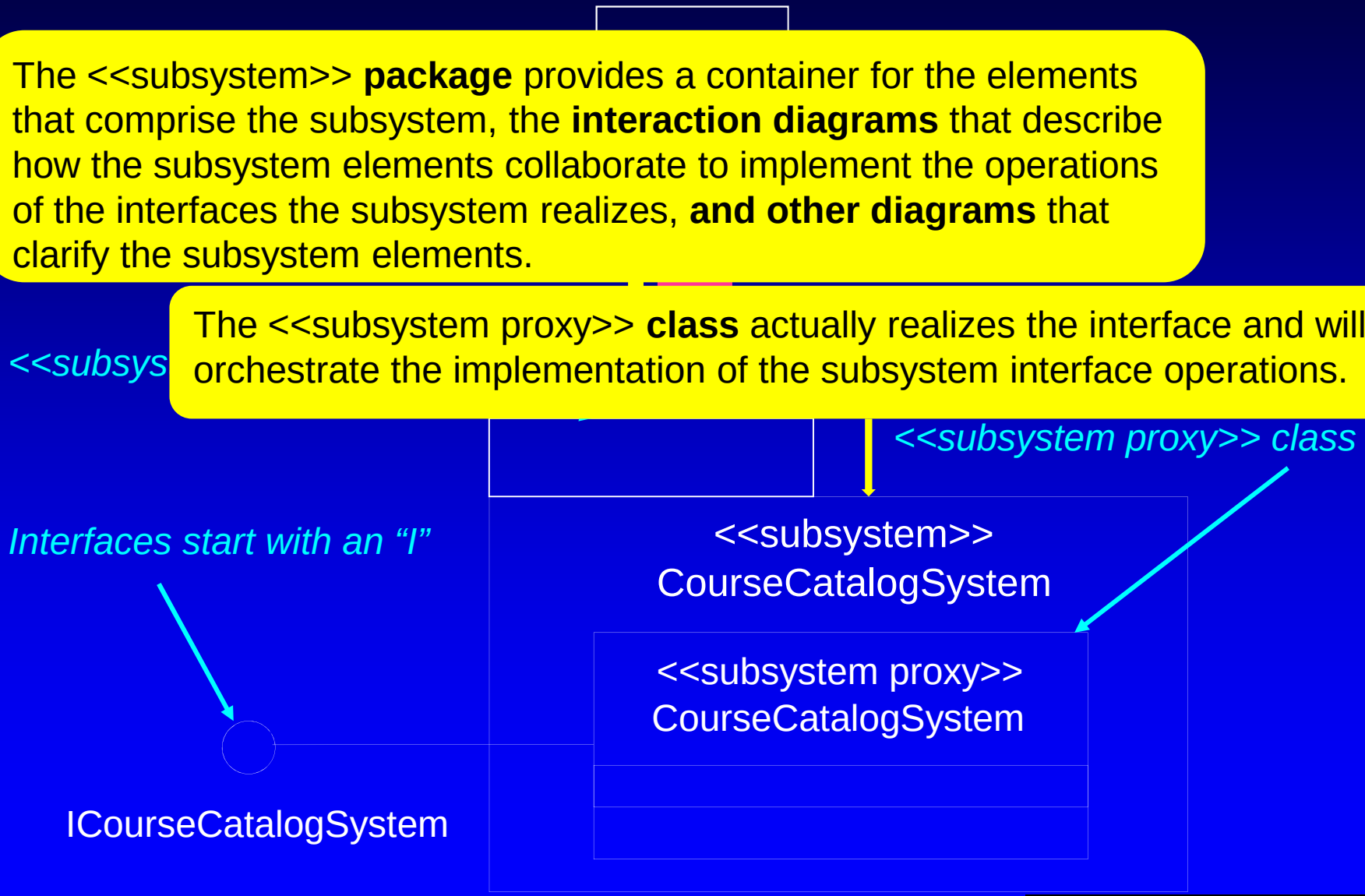


Key is abstraction and encapsulation

Review: Modeling Convention for Subsystems and Interfaces

The <<subsystem>> **package** provides a container for the elements that comprise the subsystem, the **interaction diagrams** that describe how the subsystem elements collaborate to implement the operations of the interfaces the subsystem realizes, **and other diagrams** that clarify the subsystem elements.

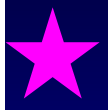
The <<subsystem proxy>> **class** actually realizes the interface and will orchestrate the implementation of the subsystem interface operations.



Subsystem Design Steps

- Distribute Subsystem behavior to Subsystem Elements
- Document Subsystem Elements
- Describe Subsystem Dependencies
- Checkpoints

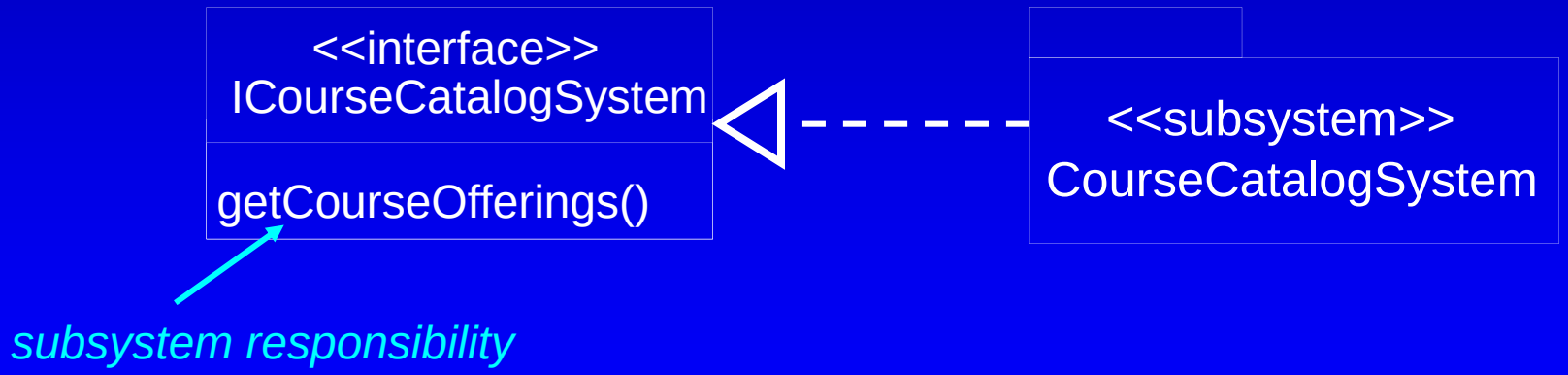
Subsystem Design Steps



- Distribute Subsystem behavior to Subsystem Elements
- Document Subsystem Elements
- Describe Subsystem Dependencies
- Checkpoints

Subsystem Responsibilities

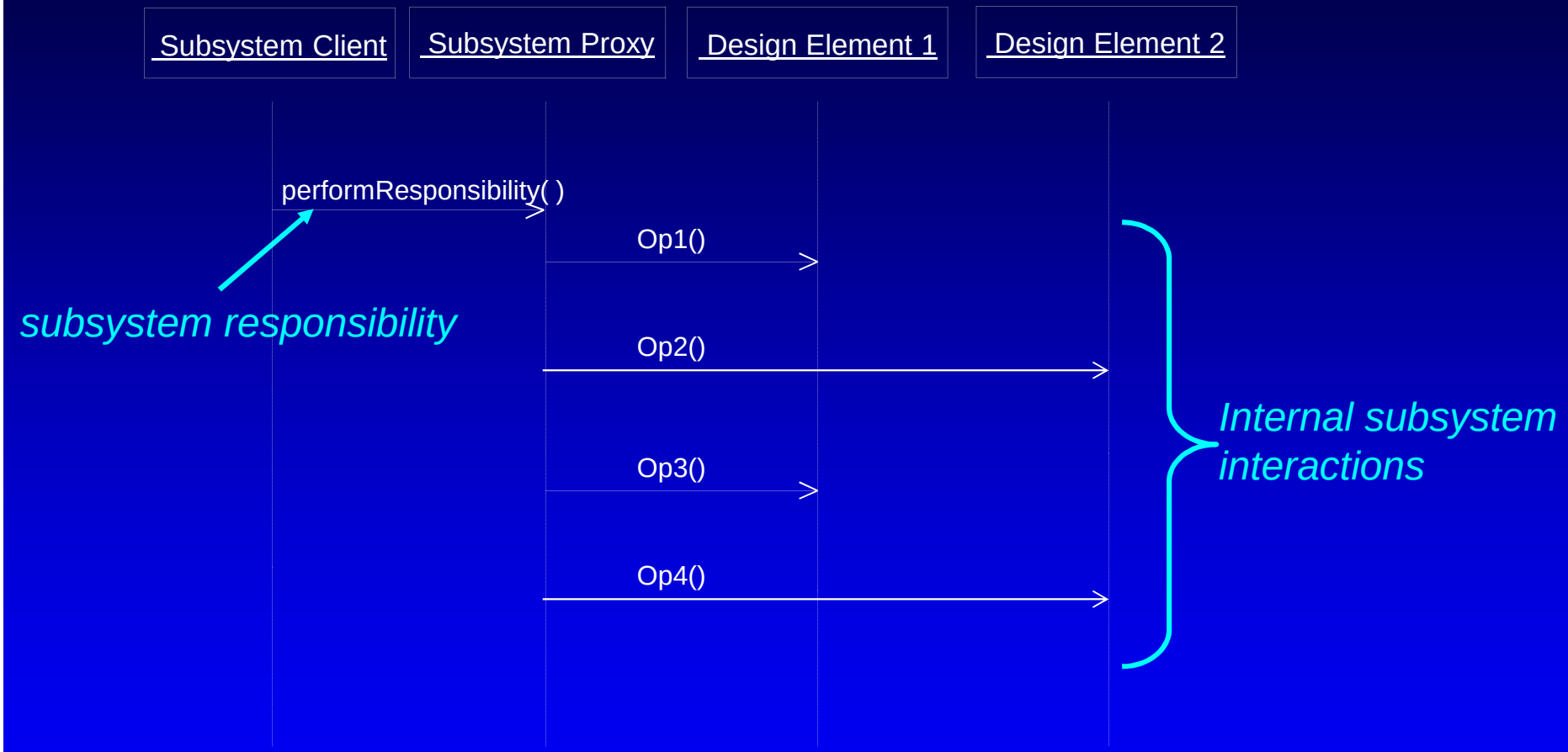
- Subsystem responsibilities defined by interface operations
- Interface operations may be realized by
  - Internal class operations
  - Internal subsystem operations



Distributing Subsystem Responsibilities

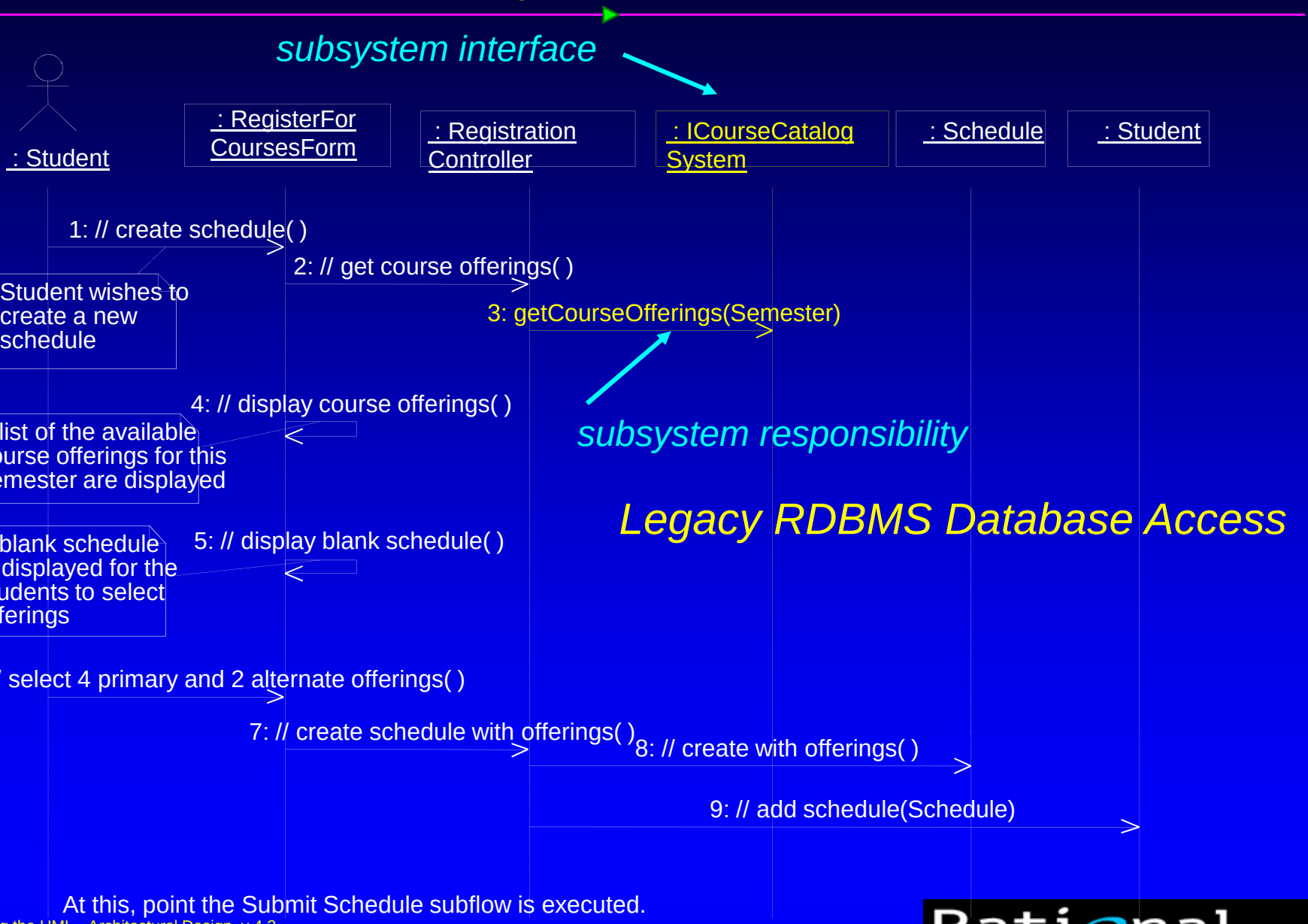
- Identify new, or reuse existing, design elements (e.g., classes and/or subsystems)
- Be careful to **avoid having effectively the same class in two different subsystems**.
  Existence of such a class implies that the subsystem boundaries may not be well-drawn.
- Document design element collaborations in “interface realizations”
- Diagram are owned by the subsystem, and are used to design the internal behavior of the subsystem. The diagrams are essential for subsystems with complex internal designs. It also enables the subsystem behavior to be easily understood, hopefully rendering it reusable across contexts.
- These internal interaction diagrams should incorporate any applicable mechanisms initially identified in Architectural Design (e.g., persistence, distribution, etc.)
- Identify new, or reuse existing, design elements (e.g., classes and/or subsystems), as needed

Modeling Convention: Subsystem Interaction Diagrams



Subsystem interface not shown

Example: CourseCatalogSystem Subsystem In Context



Incorporating the Architectural Mechanisms: Persistency

- Analysis-Class-to-Architectural-Mechanism Map from Use-Case Analysis

Analysis Class	Analysis Mechanism(s)
Student	Persistency, Security
Schedule	Persistency, Security
CourseOffering	*Persistency, Legacy Interface*
Course	*Persistency, Legacy Interface*
RegistrationController	Distribution

OODBMS Persistency

RDBMS Persistency

OODBMS Persistency was discussed in Use-Case Design

Review: Incorporating JDBC: Steps

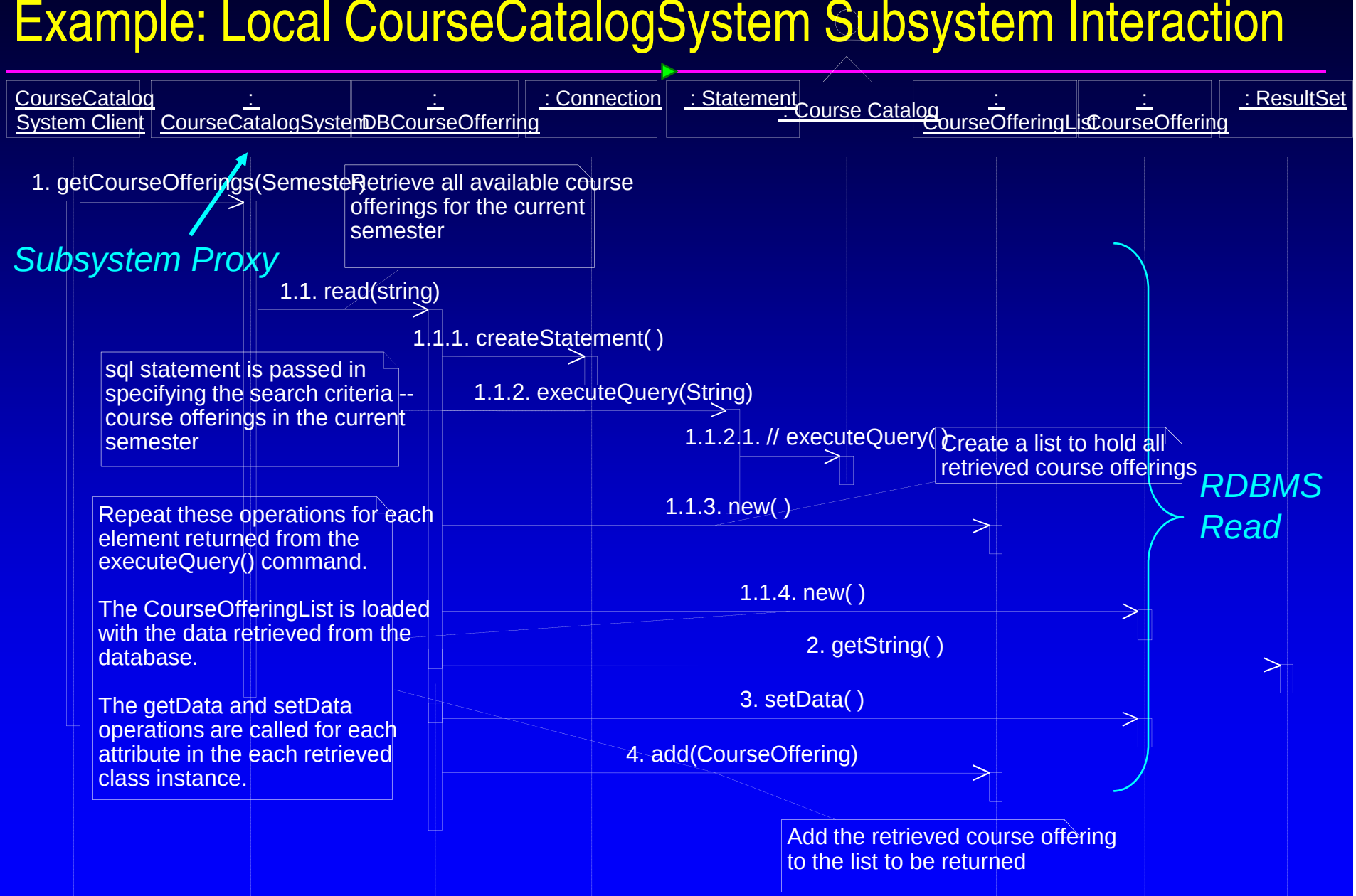
- Provide access to the class libraries needed to implement JDBC
  - √ *Provide java.sql package*
- Create the necessary DBClasses
  - *One DBClass per persistent class*
  - *Course Offering persistent class => DBCourseOffering*

(continued)

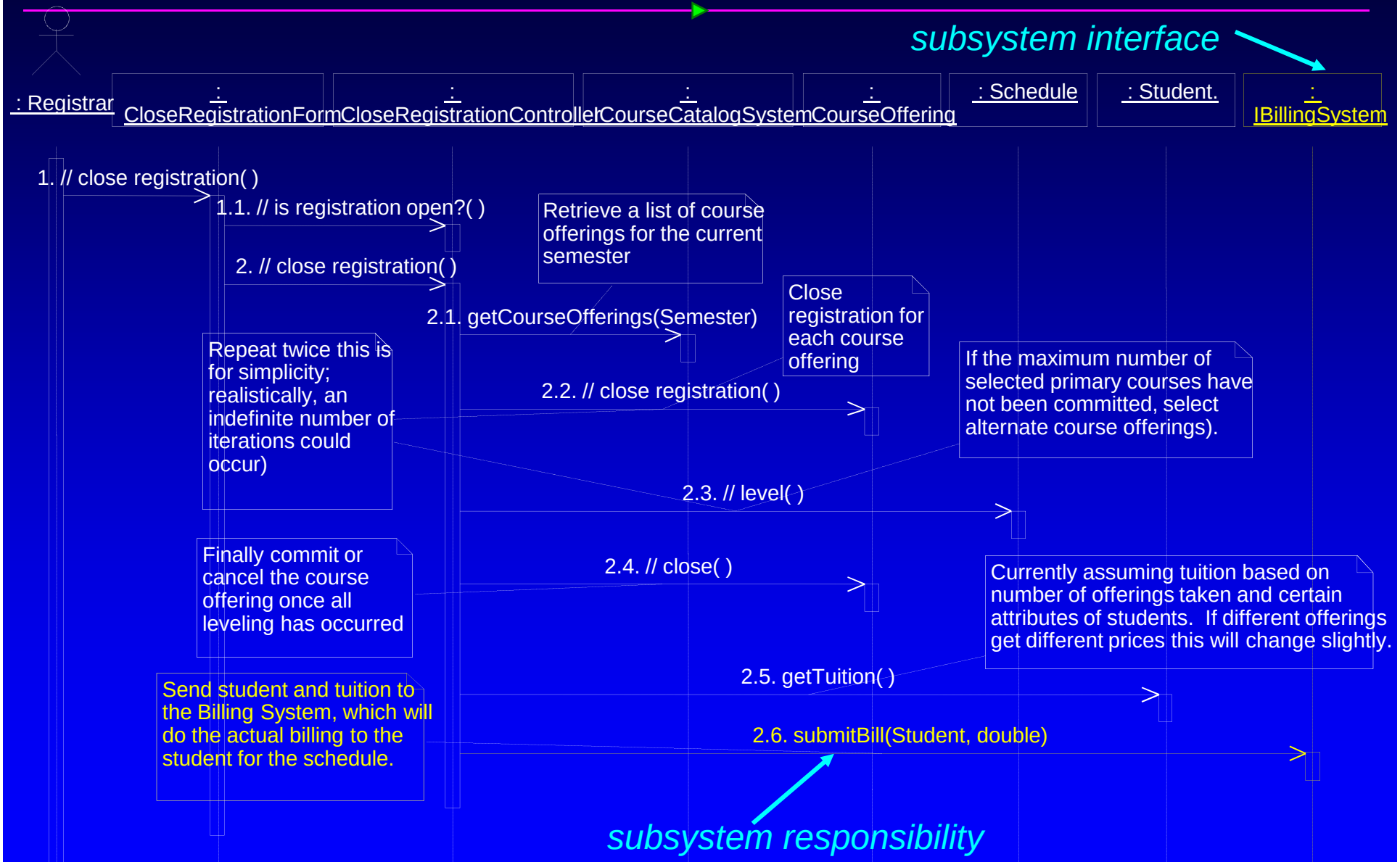
Review: Incorporating JDBC: Steps (contd.)

- Incorporate DBClasses into the design
  - Allocate to package/layer
    - *DBCourseOffering placed in CourseCatalogSystem subsystem*
  - Add relationships from persistency clients
    - *Persistency clients are the CourseCatalogSystem subsystem clients*
- Create/Update interaction diagrams that describe:
  - Database initialization
  - Persistent class access: Create, Read, Update, Delete

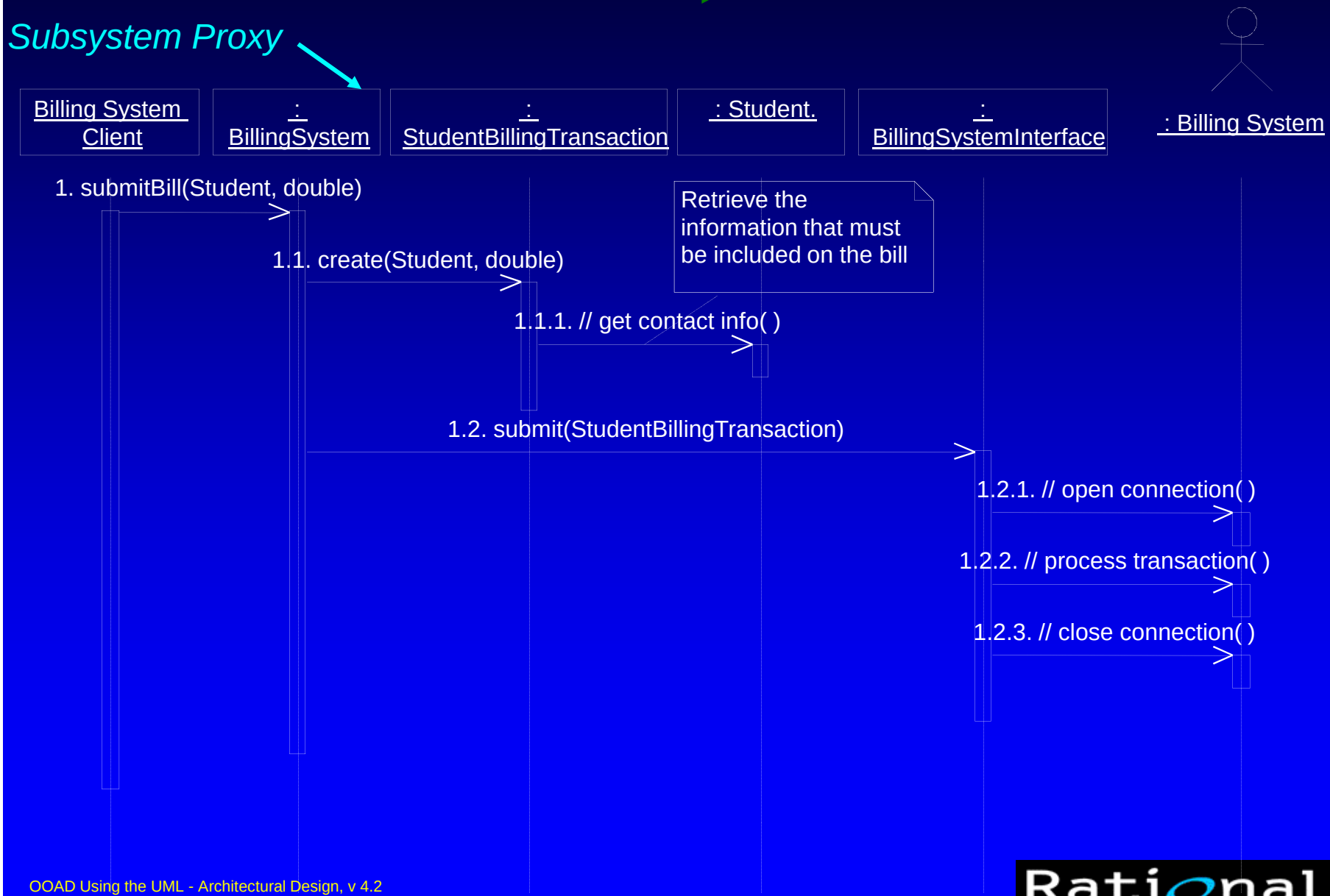
Example: Local CourseCatalogSystem Subsystem Interaction



Example: Billing System Subsystem In Context



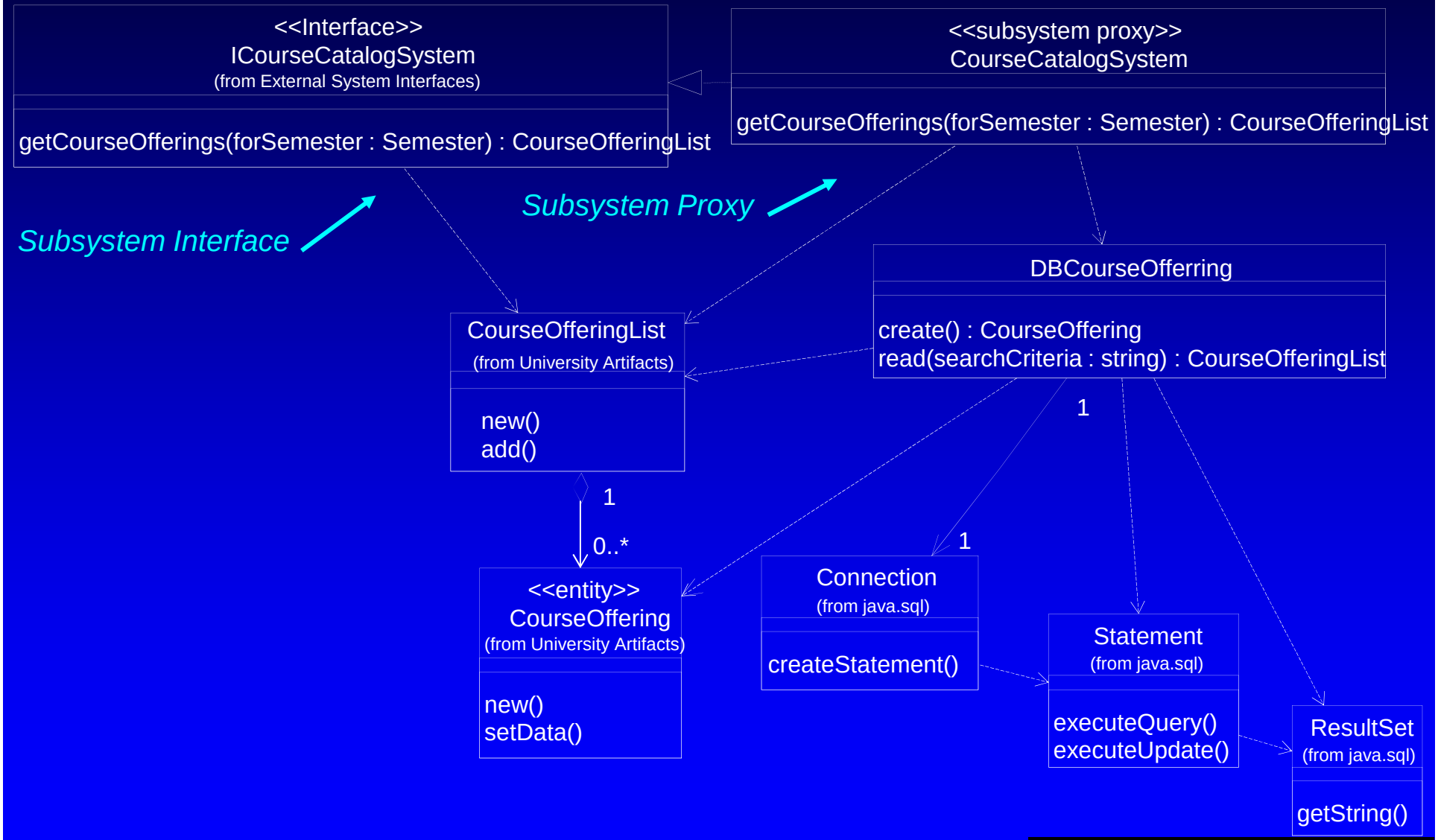
Example: Local BillingSystem Subsystem Interaction



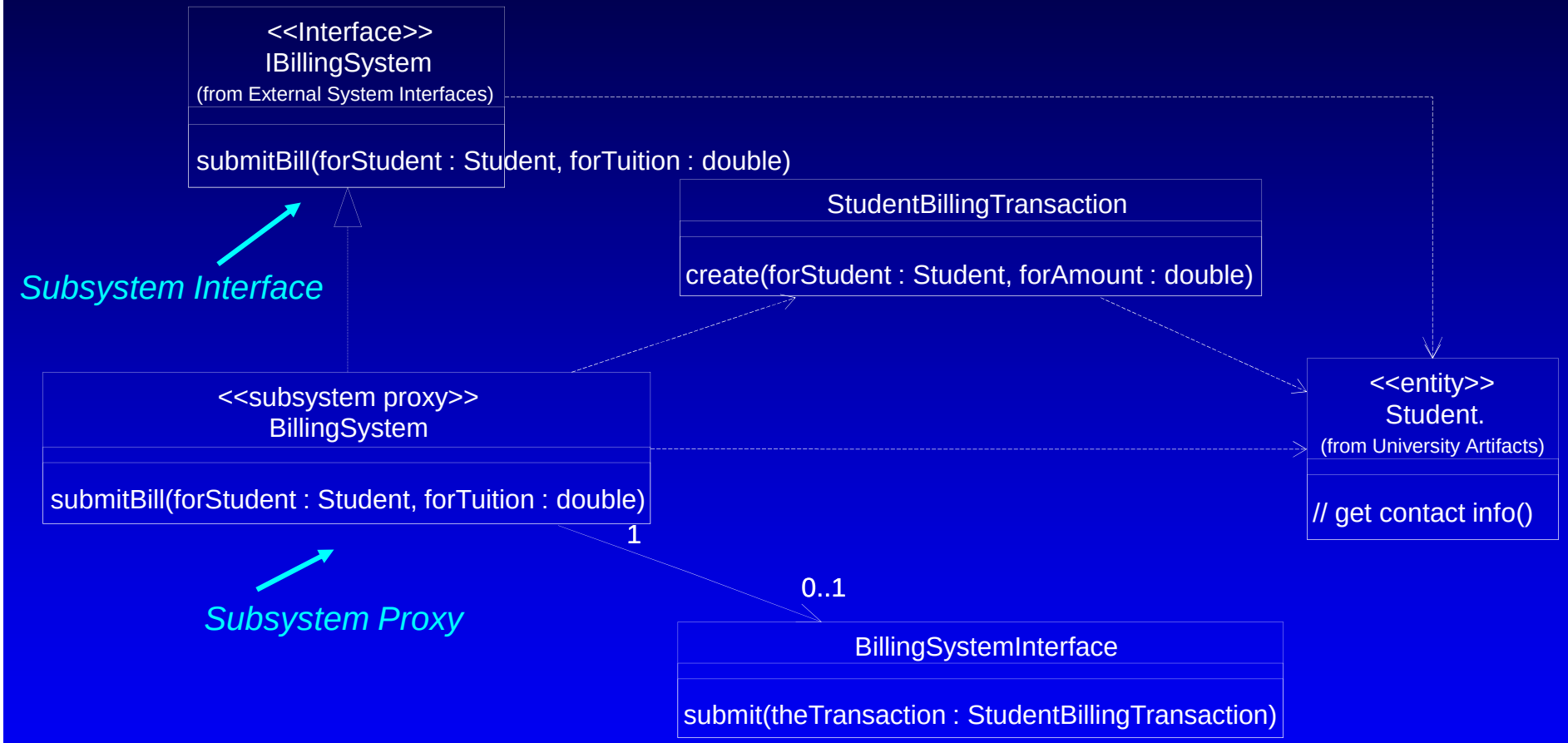
Subsystem Design Steps

- Distribute Subsystem behavior to Subsystem Elements
- Document Subsystem Elements
- Describe Subsystem Dependencies
- Checkpoints

Example: CourseCatalogSystem Subsystem Elements



Example: Billing System Subsystem Elements

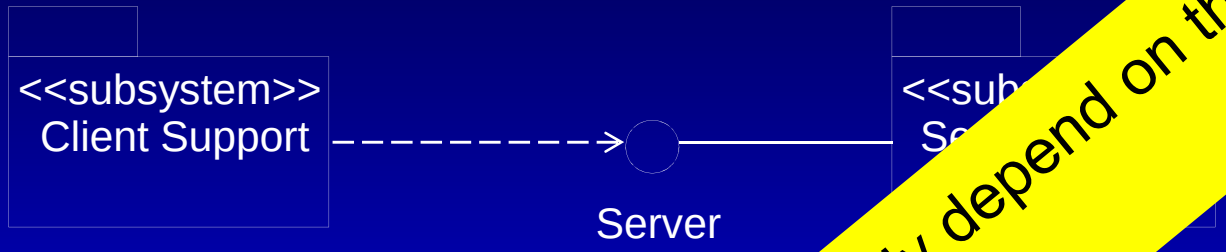


Subsystem Design Steps

- Distribute Subsystem behavior to Subsystem Elements
- Document Subsystem Elements
- Describe Subsystem Dependencies
- Checkpoints

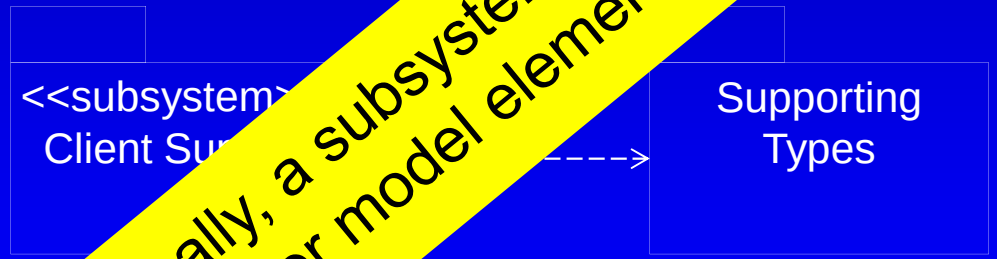
Describing Subsystem Dependencies

- Subsystem dependency on a sub



Flexible

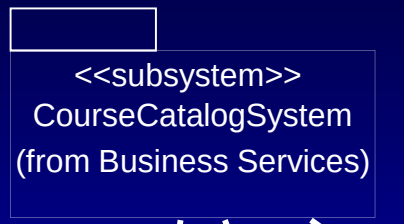
- Subsystem depend a package



Use with care

Ideally, a subsystem should only depend on the **interfaces** of other model elements.

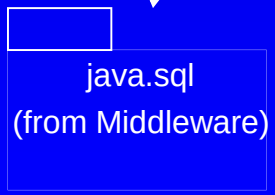
Example: CourseCatalogSystem Subsystem Dependencies



Access to the subsystem interface itself (ICourseCatalogSystem).

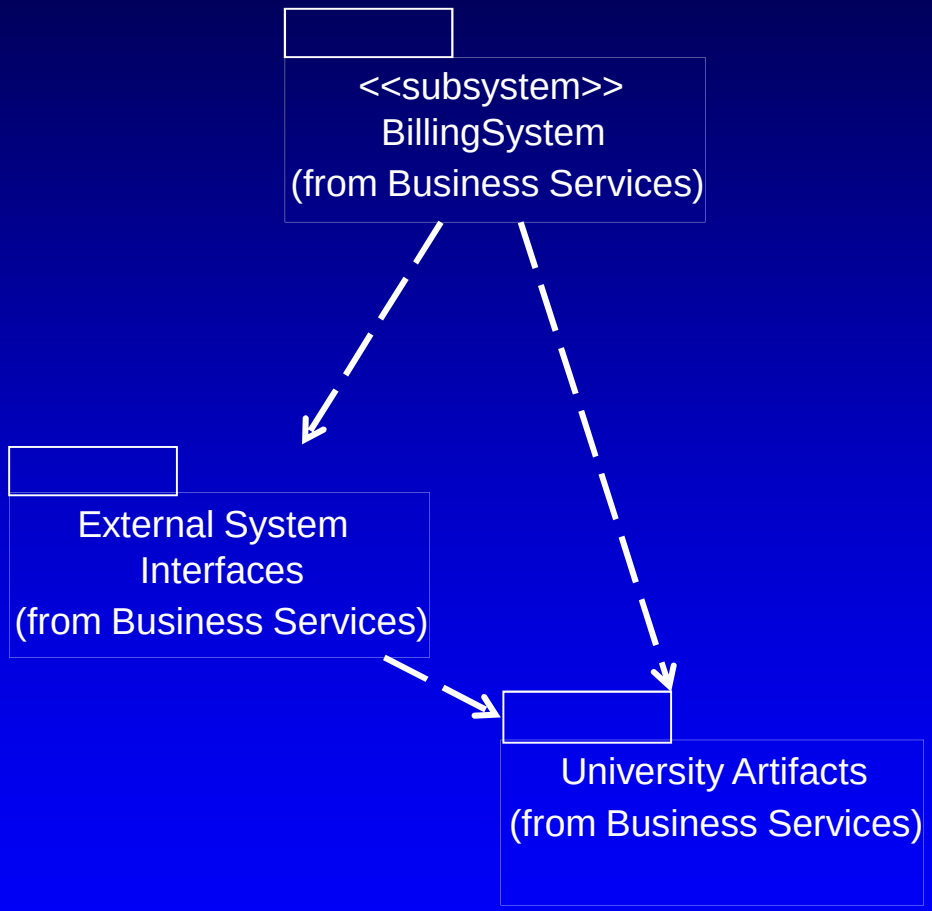
MEMO
A subsystem cannot be reused without the packages it depends on

Access the core types of the Course Registration System



Access the DBMS

Example: BillingSystem Subsystem Dependencies



Subsystem Design Steps

- Distribute Subsystem behavior to Subsystem Elements
- Document Subsystem Elements
- Describe Subsystem Dependencies
- Checkpoints

Checkpoints: Design Subsystems

- Is a realization association defined for each interface offered by the subsystem?
- Is a dependency association defined for each interface used by the subsystem?
- Ensure that none of the elements within the subsystem have public visibility.
- Is each operation on an interface realized by the subsystem documented in a interaction diagram? If not, is the operation realized by a single class, so that it is easy to see that there is a simple 1:1 mapping between the class operation and the interface operation?

Review: Subsystem Design

- What is the purpose of Subsystem Design?
- How many interaction diagrams should be produced during Subsystem Design?
- Why should dependencies on a subsystem be on the subsystem interface?

Exercise: Subsystem Design

- Given the following:
  - The defined subsystems, their interfaces and their relationships with other design elements (the subsystem context diagrams)
  - Patterns of use for the architectural mechanisms

(continued)

Exercise: Subsystem Design (cont.)

- Identify the following for a particular subsystem(s):
  - The design elements contained within the subsystem and their relationships
  - The applicable architectural mechanisms
  - The interactions needed to implement the subsystem interface operations

(continued)

Exercise: Subsystem Design (cont.)

- Produce the following diagrams for a particular subsystem(s):
  - "Interface realizations"
    - Interaction diagram for each interface operation
    - Class diagram containing the subsystem design elements that realize the interface responsibilities and their relationships
  - Class diagram that shows the subsystem and any dependencies on external packag(es) and/or subsystem(s) (subsystem dependencies class diagram)