Chapter 13 Subsystem Design

Agenda

- Objectives
- Subsystem Design in Context
- Subsystem Design Steps
- Lab

Objectives: Subsystem Design

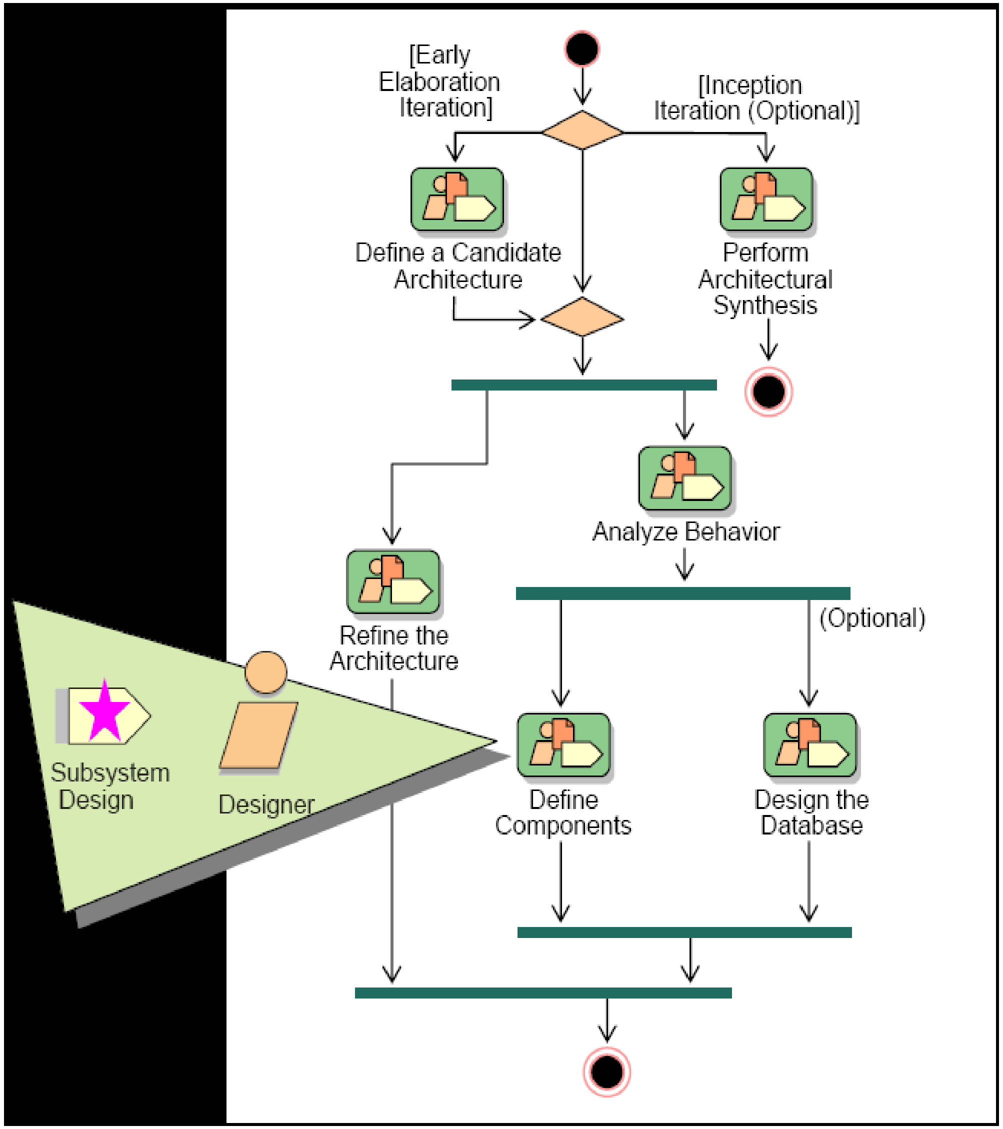
- Describe the purpose of Subsystem Design and where in the lifecycle it is performed
- Define the behaviors specified in the subsystem's interfaces in terms of collaborations of contained classes
- Document the internal structure of the subsystem
- Determine the dependencies upon elements external to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

Chapter 13 Subsystem Design

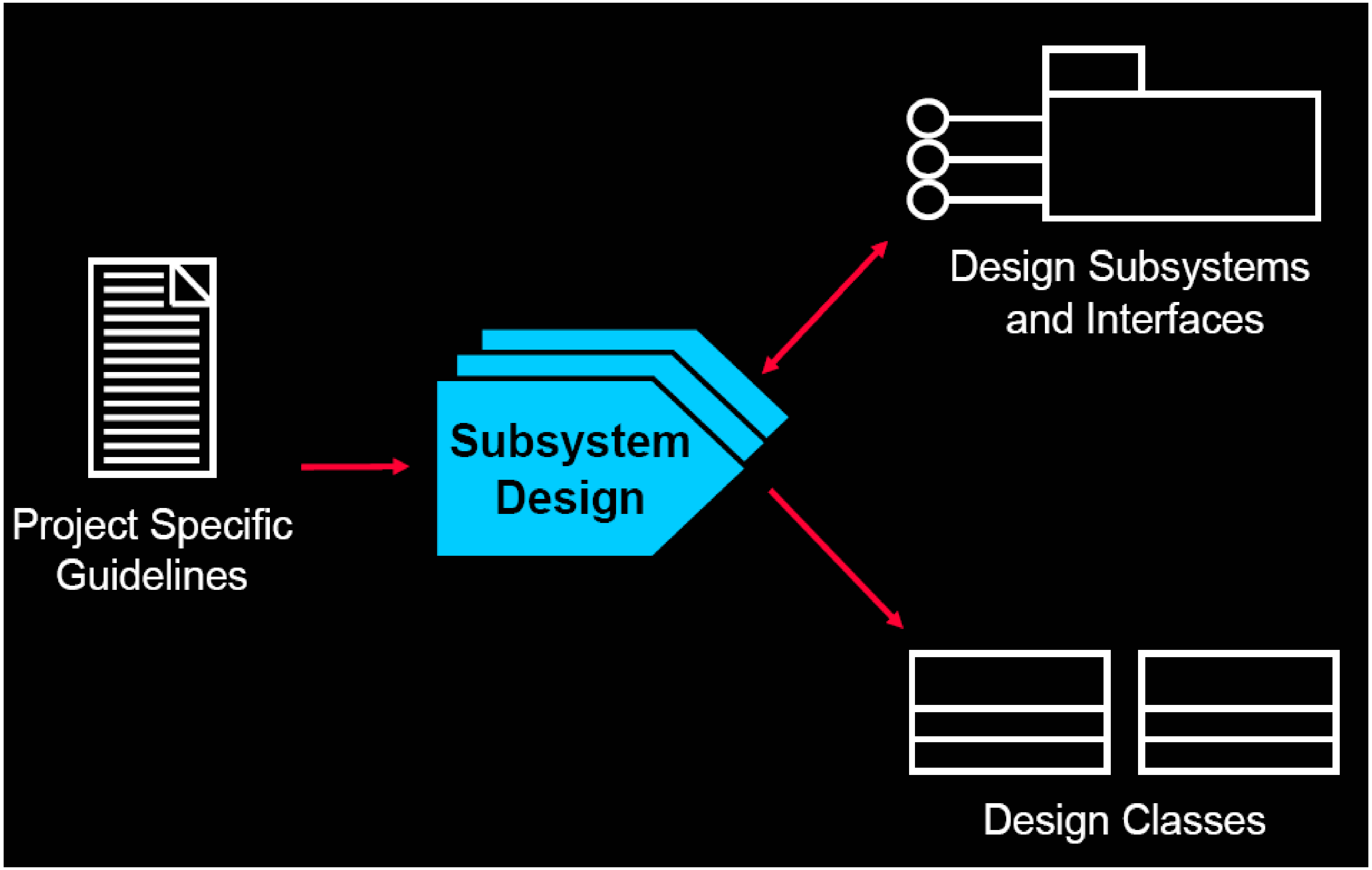
Agenda

- Objectives
- Subsystem Design in Context
- Subsystem Design Steps
- Lab

Subsystem Design in Context


[Early Elaboration Iteration]
[Inception Iteration (Optional)]
Define a Candidate Architecture
Perform Architectural Synthesis
Analyze Behavior
Refine the Architecture
(Optional)
Define Components
Design the Database
Subsystem Design
Designer

Subsystem Design Overview

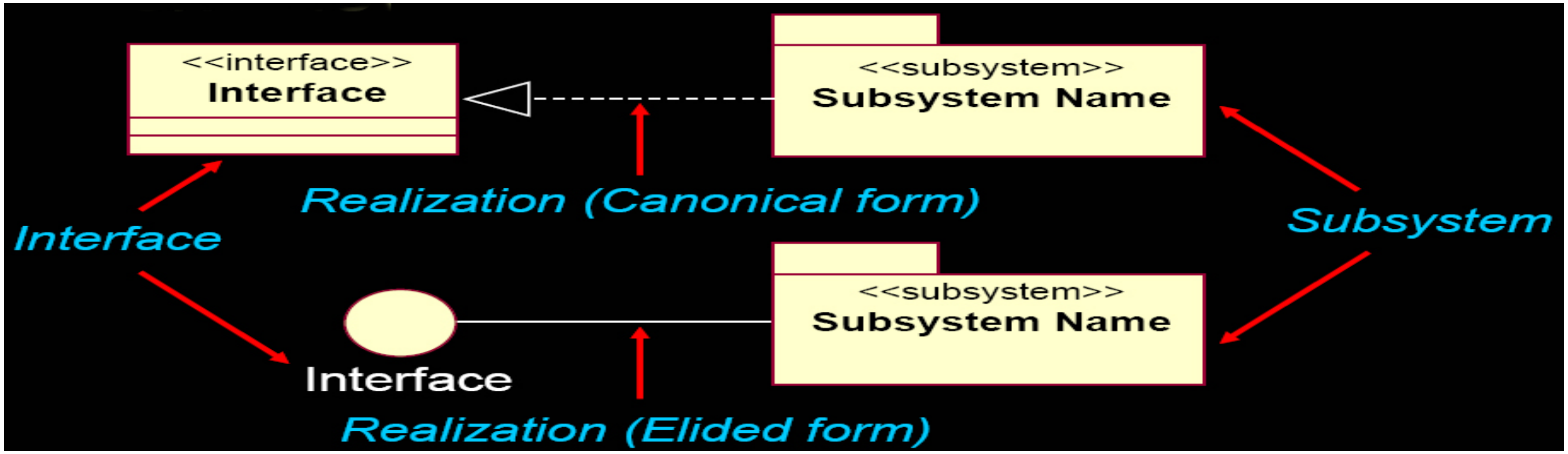


Review: Subsystems and Interfaces

- A Subsystem:

– Is a “cross between” a package and a class

– Realizes one or more interfaces that define its behavior



Subsystem Guidelines

- Goals

  – Loose coupling

□– Portability, plug-and-play compatibility

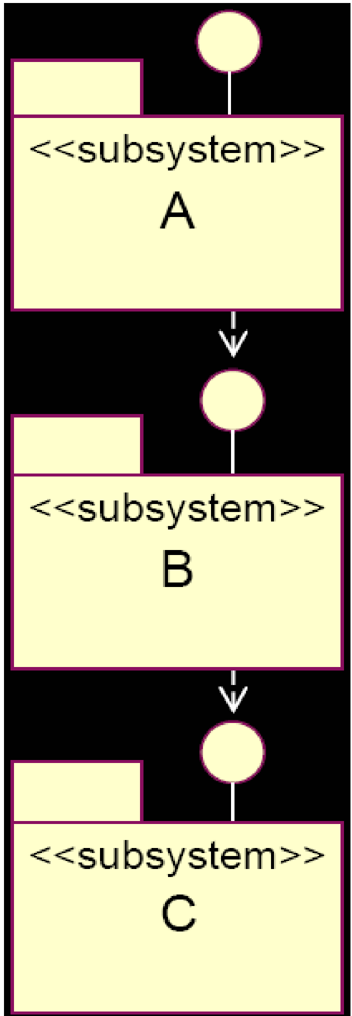
  – Insulation from change

□– Independent evolution

- Strong Suggestions

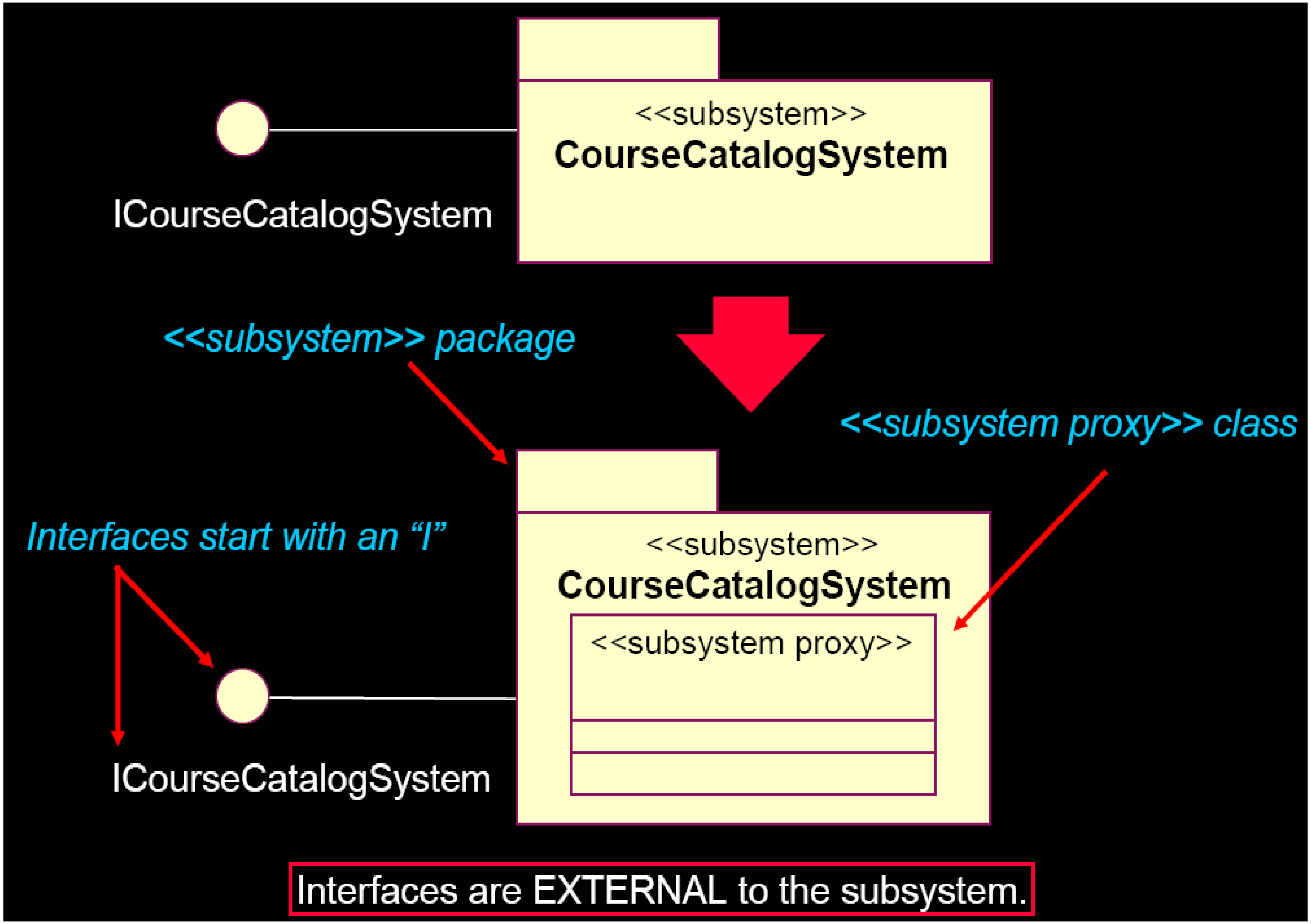
□ – Do not expose details, only interfaces

□ – Depend only on other interfaces



Key is abstraction and encapsulation

Review: Modeling Convention for Subsystems and Interfaces



Interfaces are EXTERNAL to the subsystem.

Chapter 13 Subsystem Design

Agenda

- Objectives
- Subsystem Design in Context
- Subsystem Design Steps
- Lab

Subsystem Design Steps

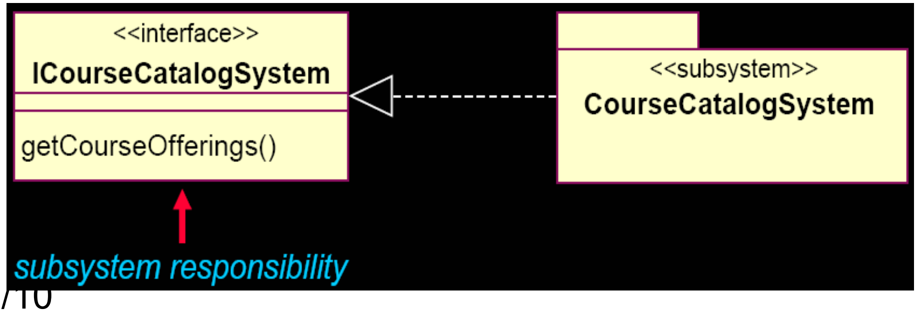
- Distribute subsystem behavior to subsystem elements
- Document subsystem elements
- Describe subsystem dependencies
- Checkpoints

Subsystem Responsibilities

- Subsystem responsibilities defined by interface operations
  - – Model interface realizations
- Interface operations may be realized by

□ – Internal class operations

□ – Internal subsystem operations

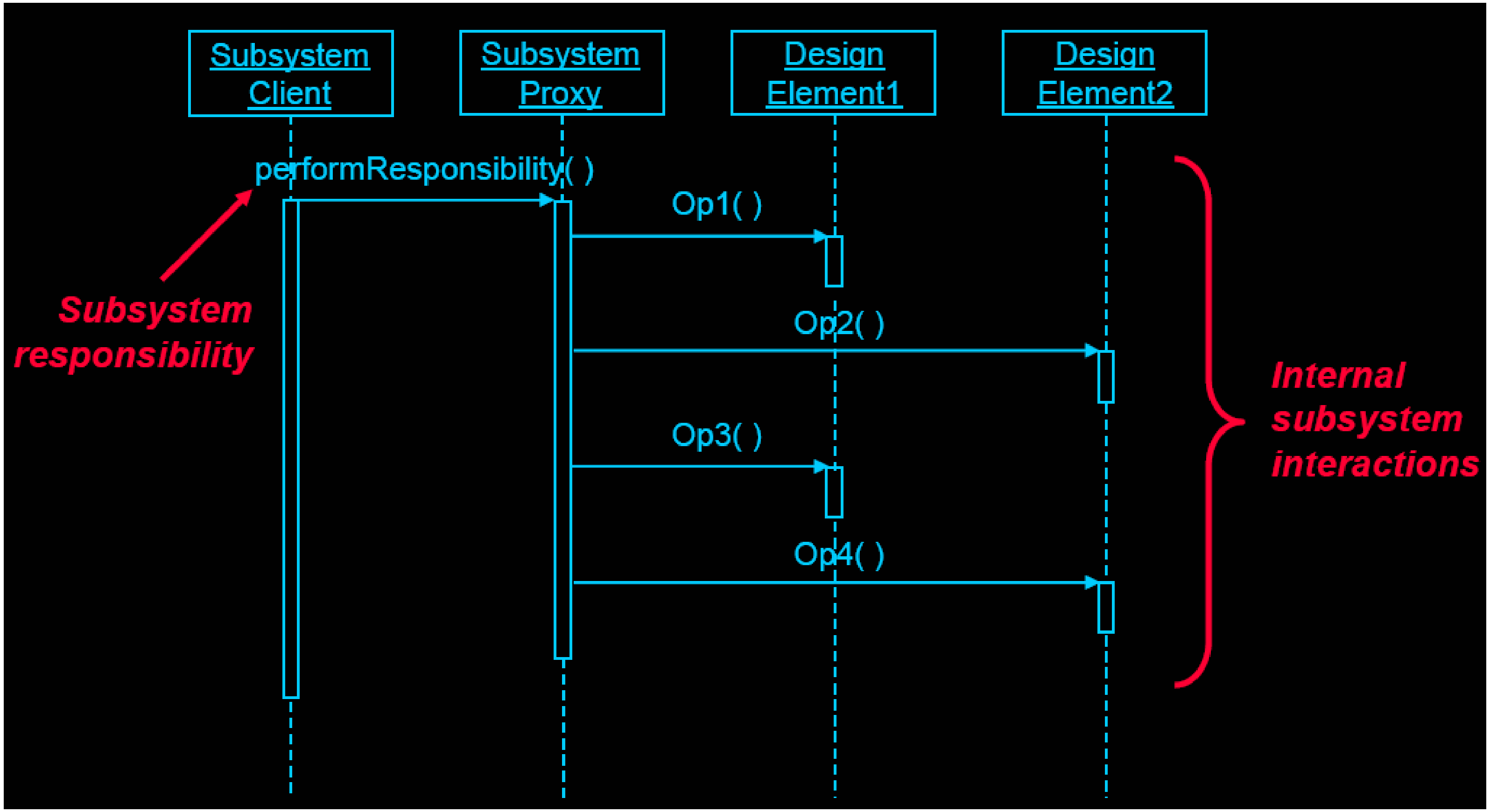


Distributing Subsystem Responsibilities

- Identify new, or reuse existing, design elements (for example, classes and/or 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 diagram(s) containing the required design element relationships
- Revisit "*Identify Design Elements*"

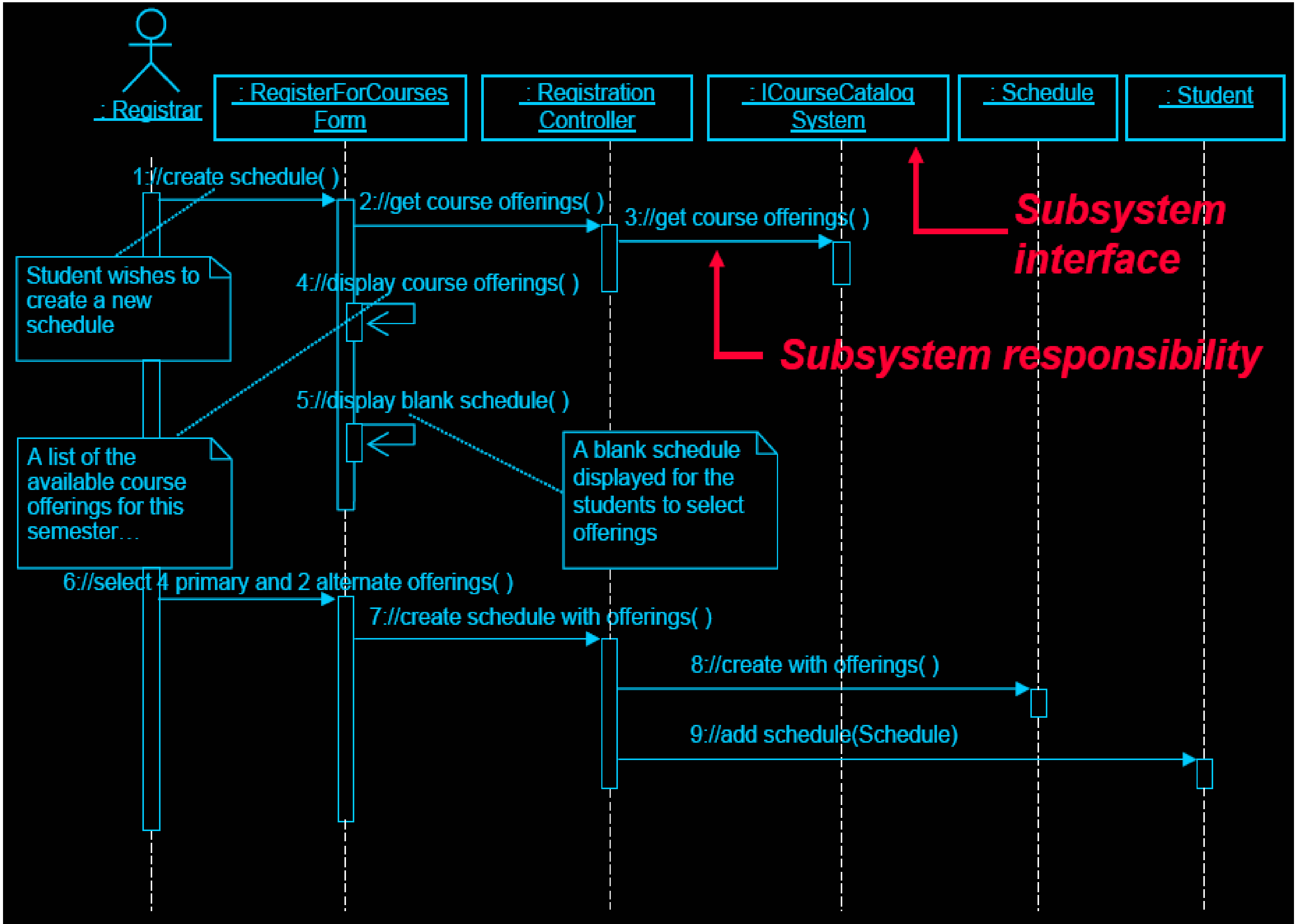
□ – Adjust subsystem boundaries and dependencies, as needed

Modeling Convention: Subsystem Interaction Diagrams



Subsystem interface not shown

Example: Course Catalog System Subsystem in Context

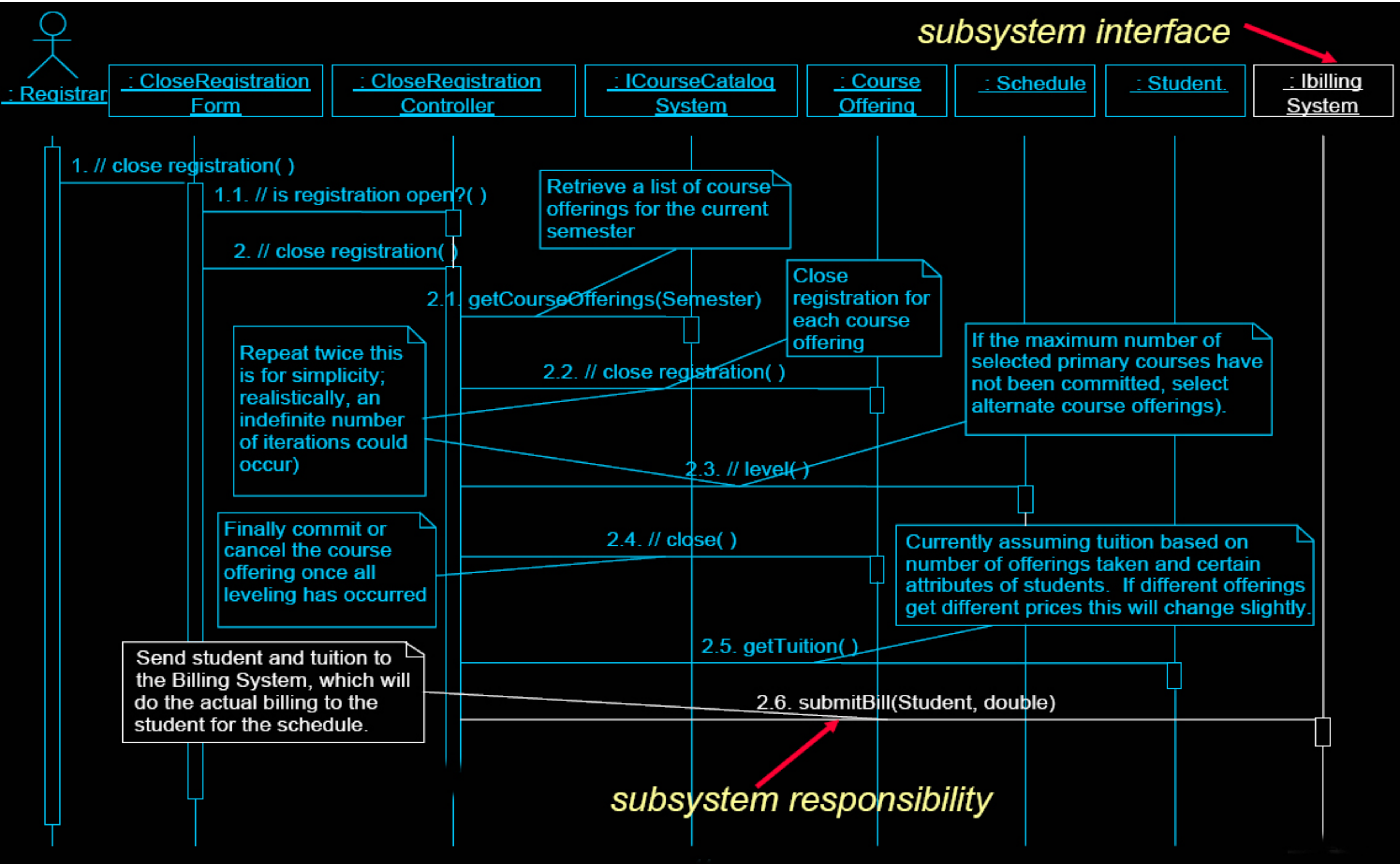

: Registrar
: RegisterForCourses Form
: Registration Controller
: ICourseCatalog System
: Schedule
: Student
1: //create schedule( )
2: //get course offerings( )
3: //get course offerings( )
4: //display course offerings( )
5: //display blank schedule( )
6: //select 4 primary and 2 alternate offerings( )
7: //create schedule with offerings( )
8: //create with offerings( )
9: //add schedule(Schedule)
Student wishes to create a new schedule
A list of the available course offerings for this semester...
A blank schedule displayed for the students to select offerings

Subsystem interface

Subsystem responsibility

Legacy RDBMS Database Access

Example: Billing System Subsystem In Context



subsystem interface

1. // close registration( )

1.1. // is registration open?( )

2. // close registration( )

2.1. getCourseOfferings(Semester)

2.2. // close registration( )

2.3. // level( )

2.4. // close( )

2.5. getTuition( )

2.6. submitBill(Student, double)

Retrieve a list of course offerings for the current semester

Close registration for each course offering

Repeat twice this is for simplicity; realistically, an indefinite number of iterations could occur)

If the maximum number of selected primary courses have not been committed, select alternate course offerings).

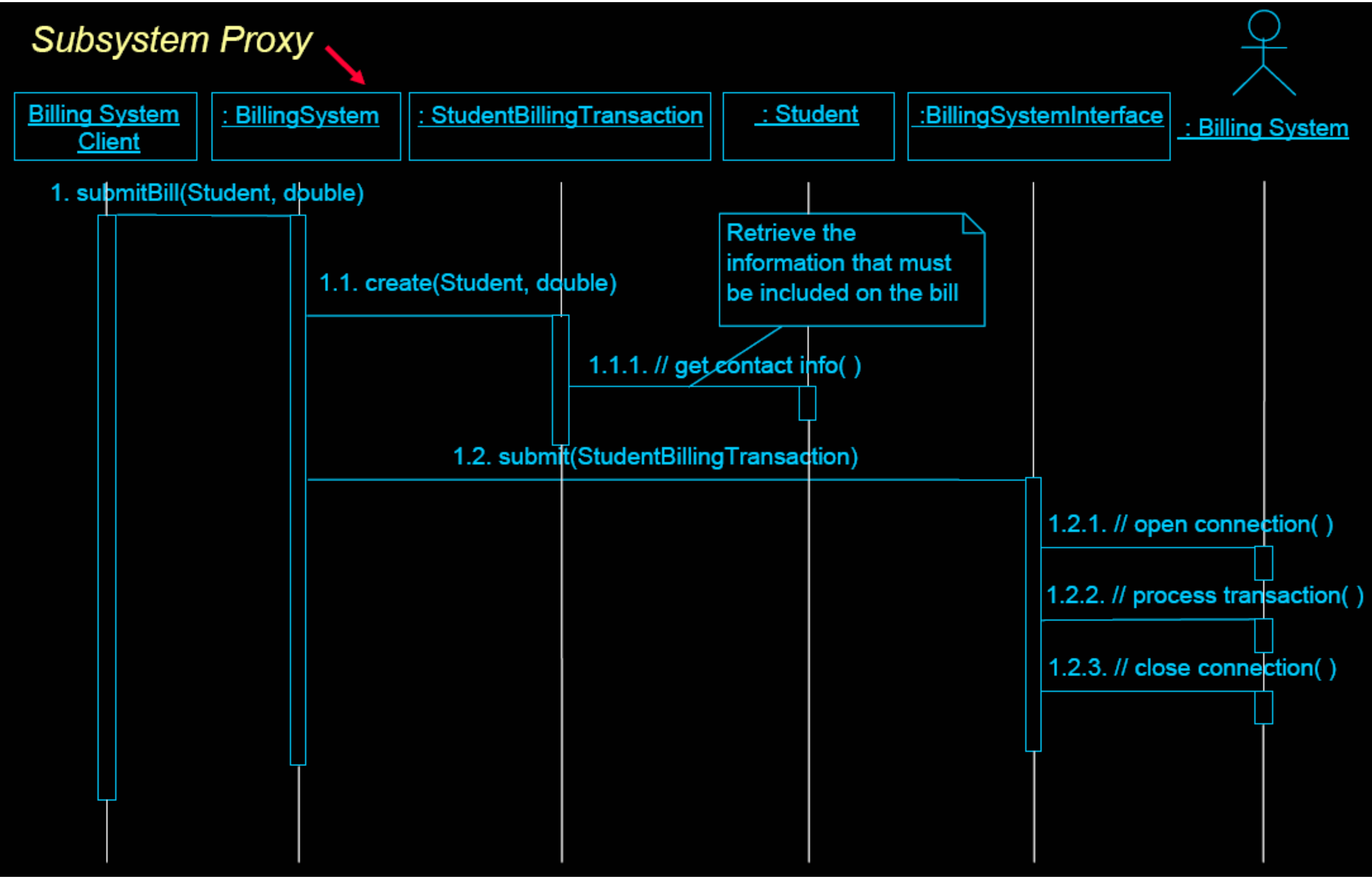
Finally commit or cancel the course offering once all leveling has occurred

Currently assuming tuition based on number of offerings taken and certain attributes of students. If different offerings get different prices this will change slightly.

Send student and tuition to the Billing System, which will do the actual billing to the student for the schedule.

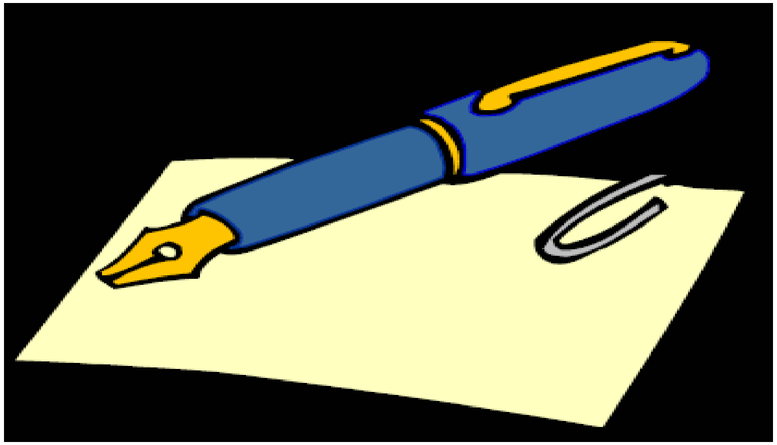
subsystem responsibility

Example: Local Billing System Subsystem Interaction

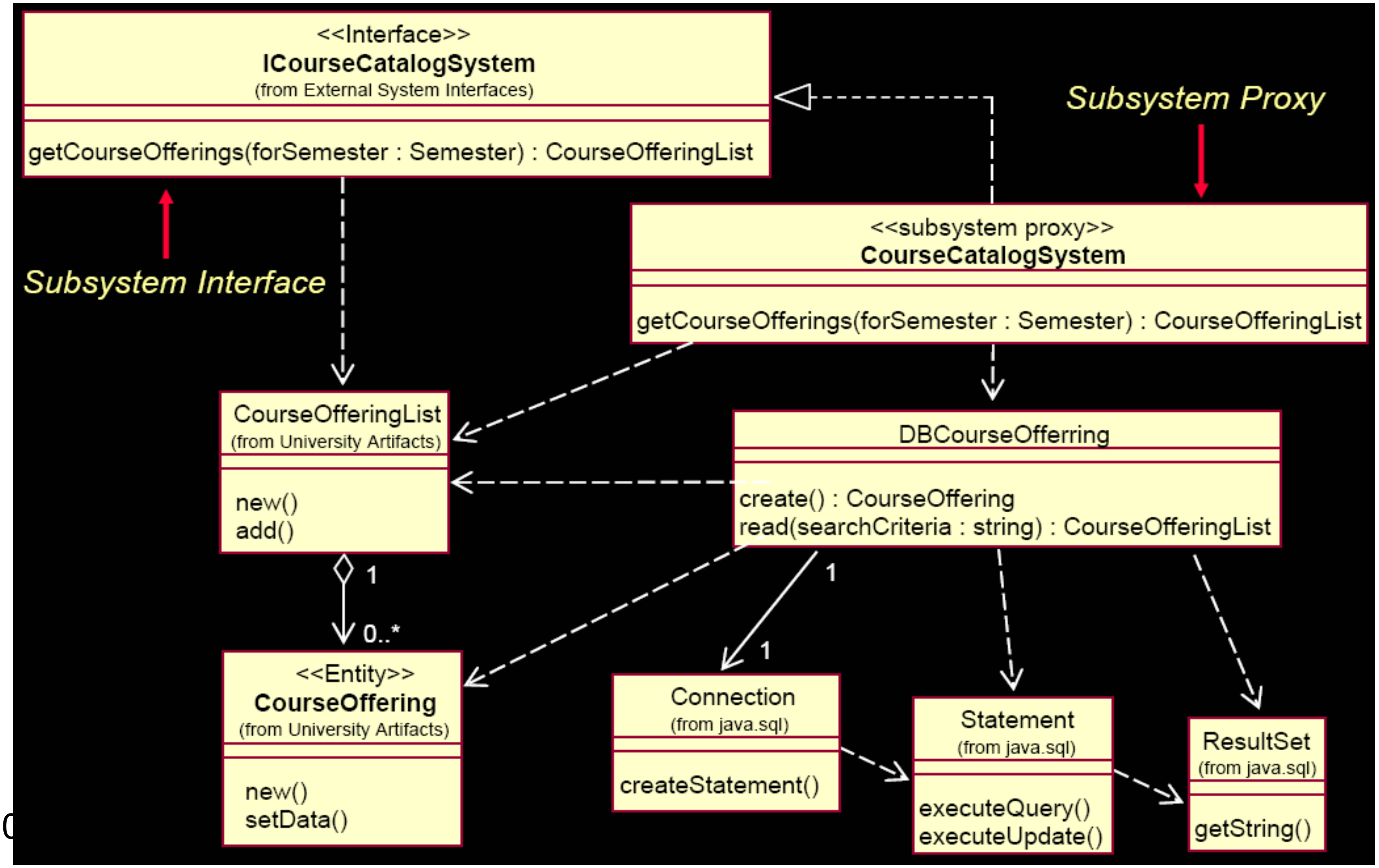

Subsystem Proxy
Billing System Client
: BillingSystem
: StudentBillingTransaction
: Student
:BillingSystemInterface
: Billing System
1. submitBill(Student, double)
1.1. create(Student, double)
Retrieve the information that must be included on the bill
1.1.1. // get contact info( )
1.2. submit(StudentBillingTransaction)
1.2.1. // open connection( )
1.2.2. // process transaction( )
1.2.3. // close connection( )

Subsystem Design Steps

- Distribute subsystem behavior to subsystem elements
- Document subsystem elements
- Describe subsystem dependencies
- Checkpoints



Example: Course Catalog System Subsystem Elements



<<Interface>>
ICourseCatalogSystem
(from External System Interfaces)

getCourseOfferings(forSemester : Semester) : CourseOfferingList

Subsystem Interface

Subsystem Proxy

<<subsystem proxy>>
CourseCatalogSystem

getCourseOfferings(forSemester : Semester) : CourseOfferingList

CourseOfferingList
(from University Artifacts)

new()
add()

DBCourseOfferring

create() : CourseOffering
read(searchCriteria : string) : CourseOfferingList

1

0..*

1

1

<<Entity>>
CourseOffering
(from University Artifacts)

new()
setData()

Connection
(from java.sql)

createStatement()

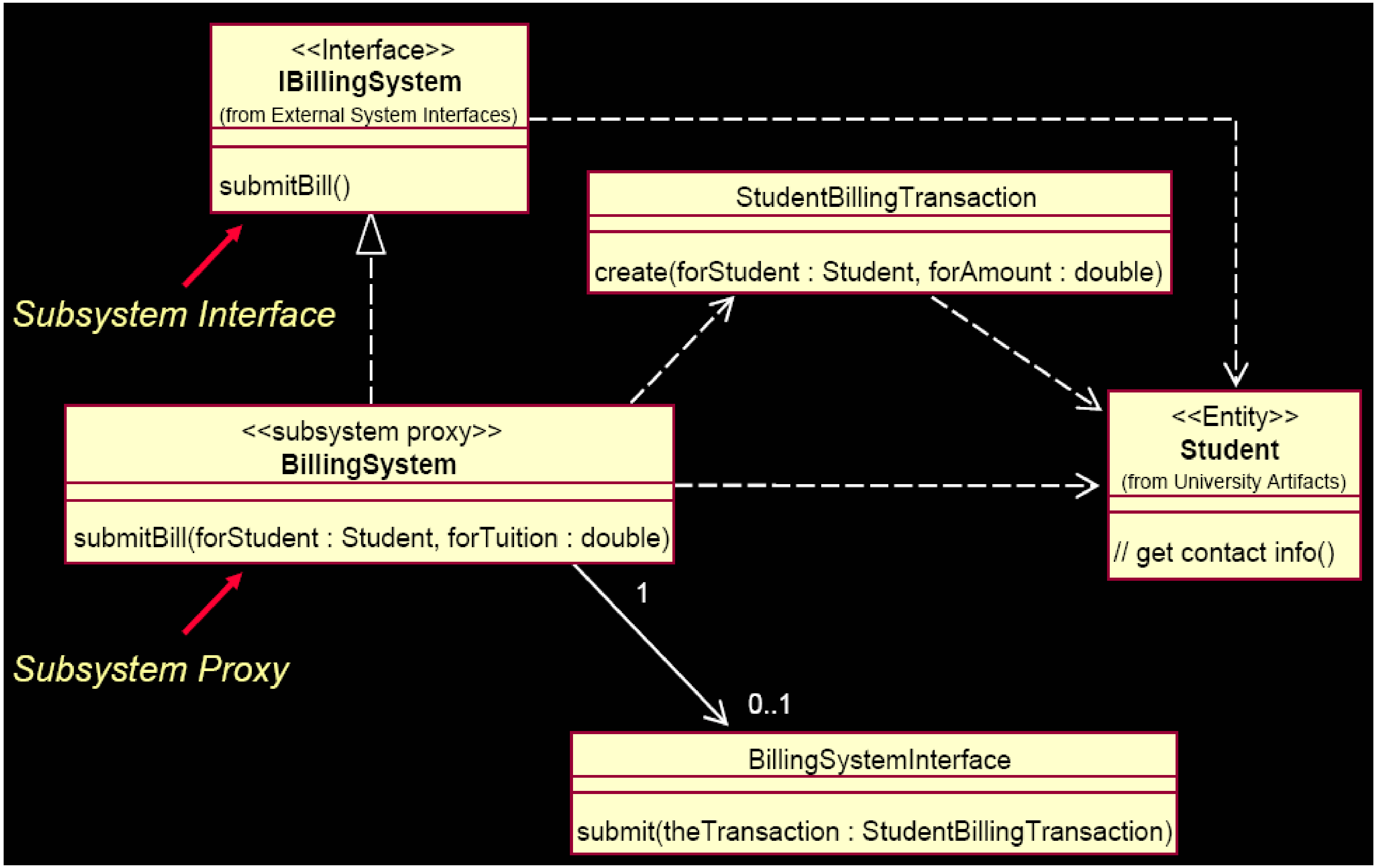
Statement
(from java.sql)

executeQuery()
executeUpdate()

ResultSet
(from java.sql)

getString()

Example: Billing System Subsystem Elements


<<Interface>>
IBillingSystem
(from External System Interfaces)
submitBill()
StudentBillingTransaction
create(forStudent : Student, forAmount : double)
Subsystem Interface
<<subsystem proxy>>
BillingSystem
submitBill(forStudent : Student, forTuition : double)
<<Entity>>
Student
(from University Artifacts)
// get contact info()
1
Subsystem Proxy
0..1
BillingSystemInterface
submit(theTransaction : StudentBillingTransaction)

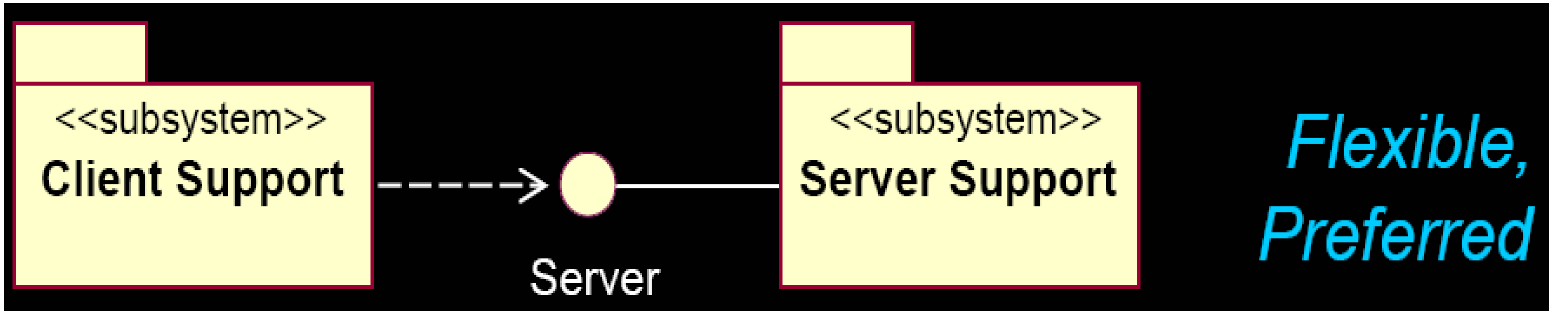
- 接口引用是对象实例的一个引用
- 代理引用是对一系列指向方法的指针的引用
  - 当通过指向方法的指针进行间接调用的同时，还有一个子程序调用了这一系列指向方法的指针。因此，使用代理调用的速度通常要比使用接口调用的速度慢
- 代理和接口的一个重要区别是可以通过使用正确的原型来创建任何方法的代理

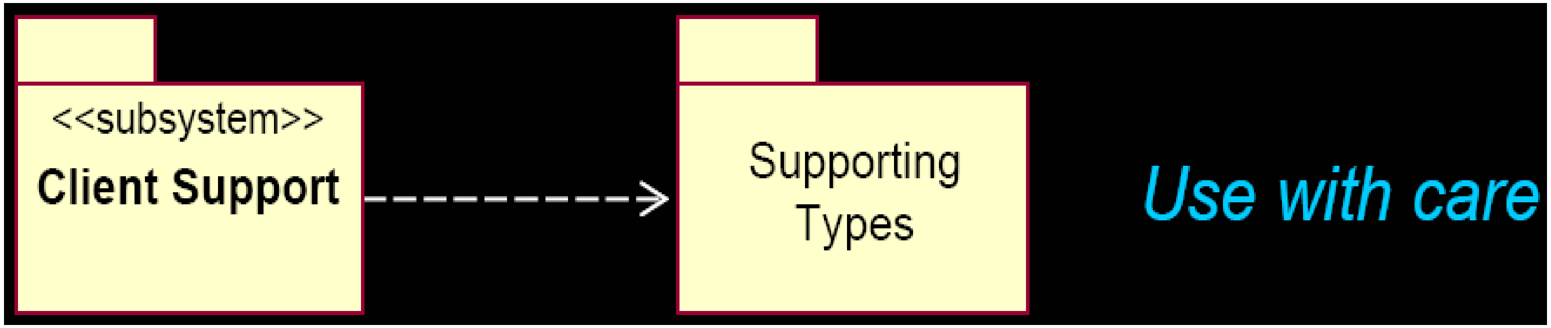
Subsystem Design Steps

- Distribute subsystem behavior to subsystem elements
- Document subsystem elements
- Describe subsystem dependencies
- Checkpoints

Subsystem Dependencies: Guidelines

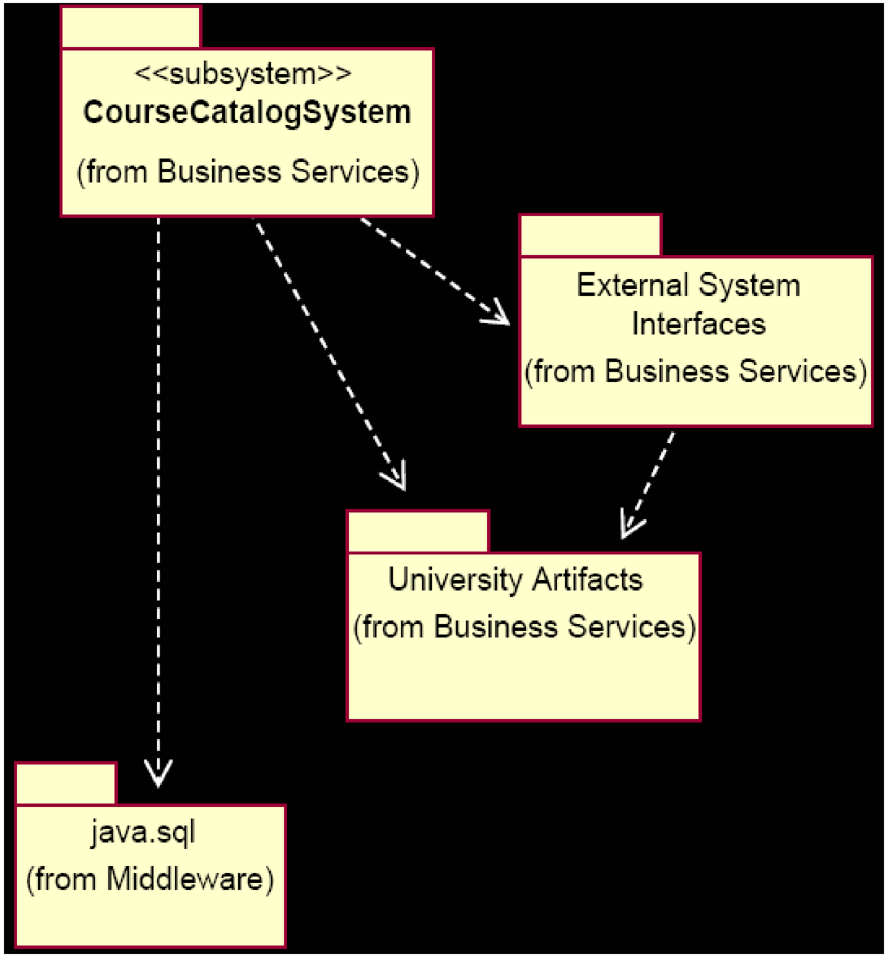
- Subsystem dependency on a subsystem



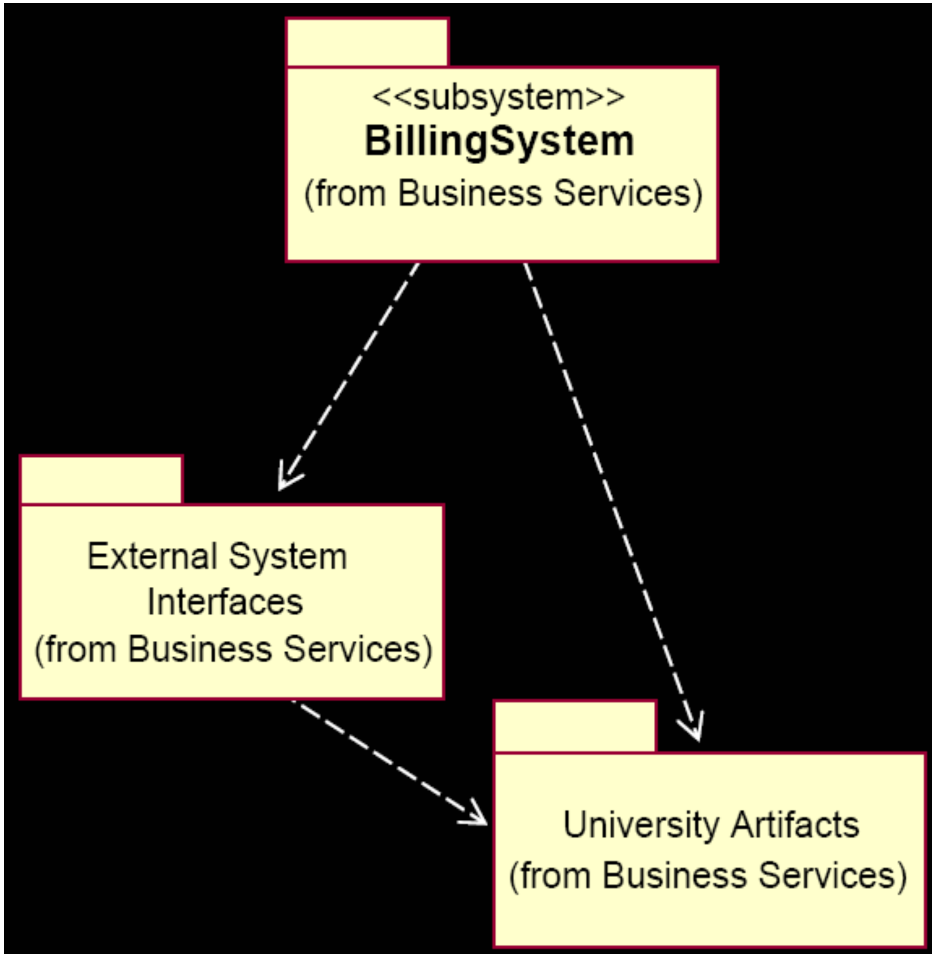


- Subsystem dependency on a package

Example: CourseCatalogSystem Subsystem Dependencies



Example: Billing System 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?
- Are you sure 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?

作业

- 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?

Chapter 13 Subsystem Design

Agenda

- Objectives
- Subsystem Design in Context
- Subsystem Design Steps
- Lab

Lab: Subsystem Design

- Given the following:
  - The defined subsystems, their interfaces and their relationships with other design elements (the subsystem context diagrams)

Lab: Subsystem Design (cont.)

- Identify the following for a particular subsystem(s):

– The design elements contained within the subsystem and their relationships

– The interactions needed to implement the subsystem interface operations

Lab: 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 package(s) and/or subsystem(s) (subsystem dependencies class diagram)

Lab: Review

- Compare your Subsystem Interface Realizations
  - Have all the main and/or 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?

2020/9/10