Chapter 12 Use-Case Design

Agenda

- Objectives
- Use-Case Design in Context
- Use-Case Design Steps
- Labs

Objectives: Use-Case Design

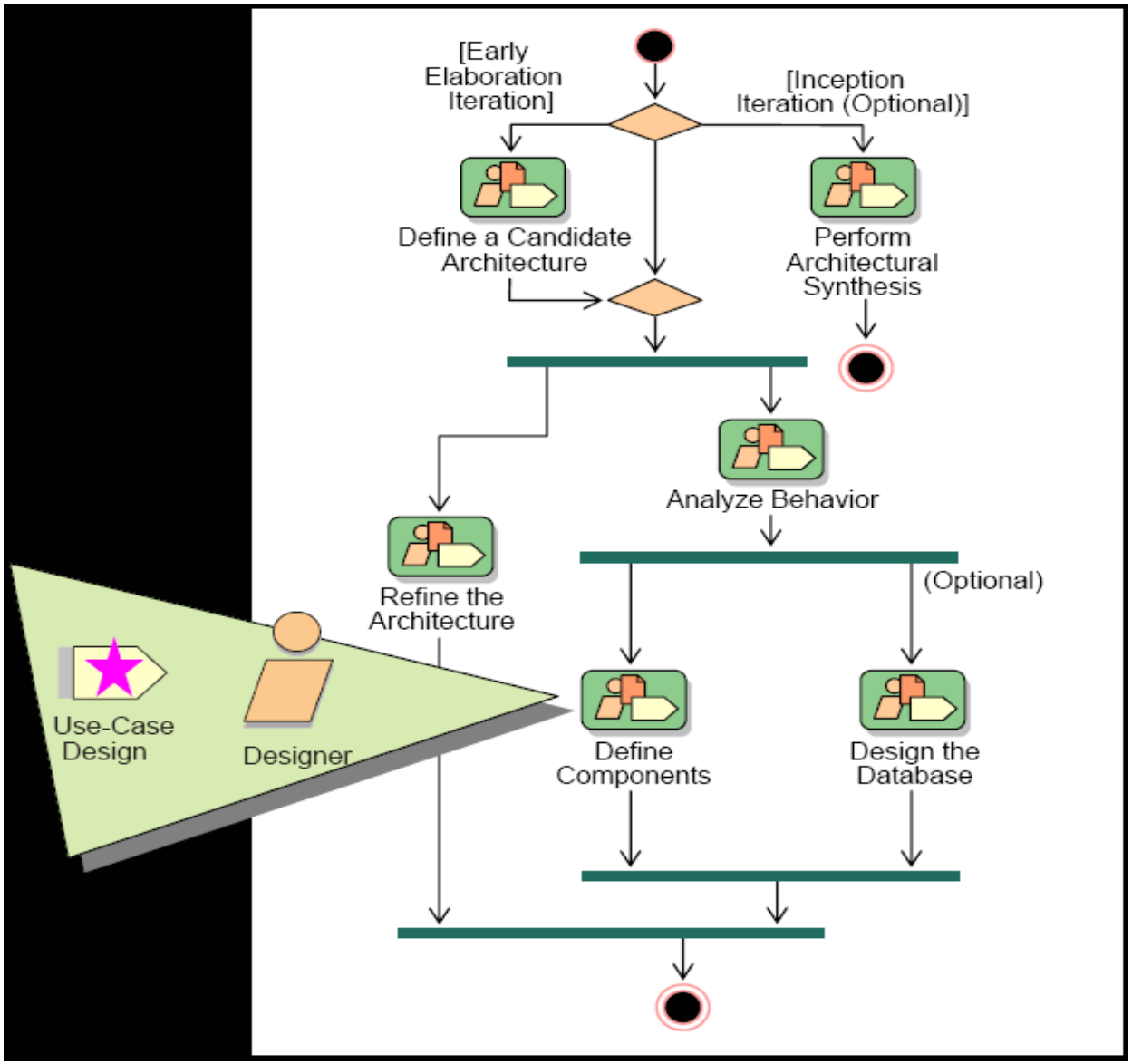
- Define the purpose of Use-Case Design and when in the lifecycle it is performed
- Verify that there is consistency in the use case implementation
- Refine the use-case realizations from Use-Case Analysis using defined Design Model elements

Chapter 12 Use-Case Design

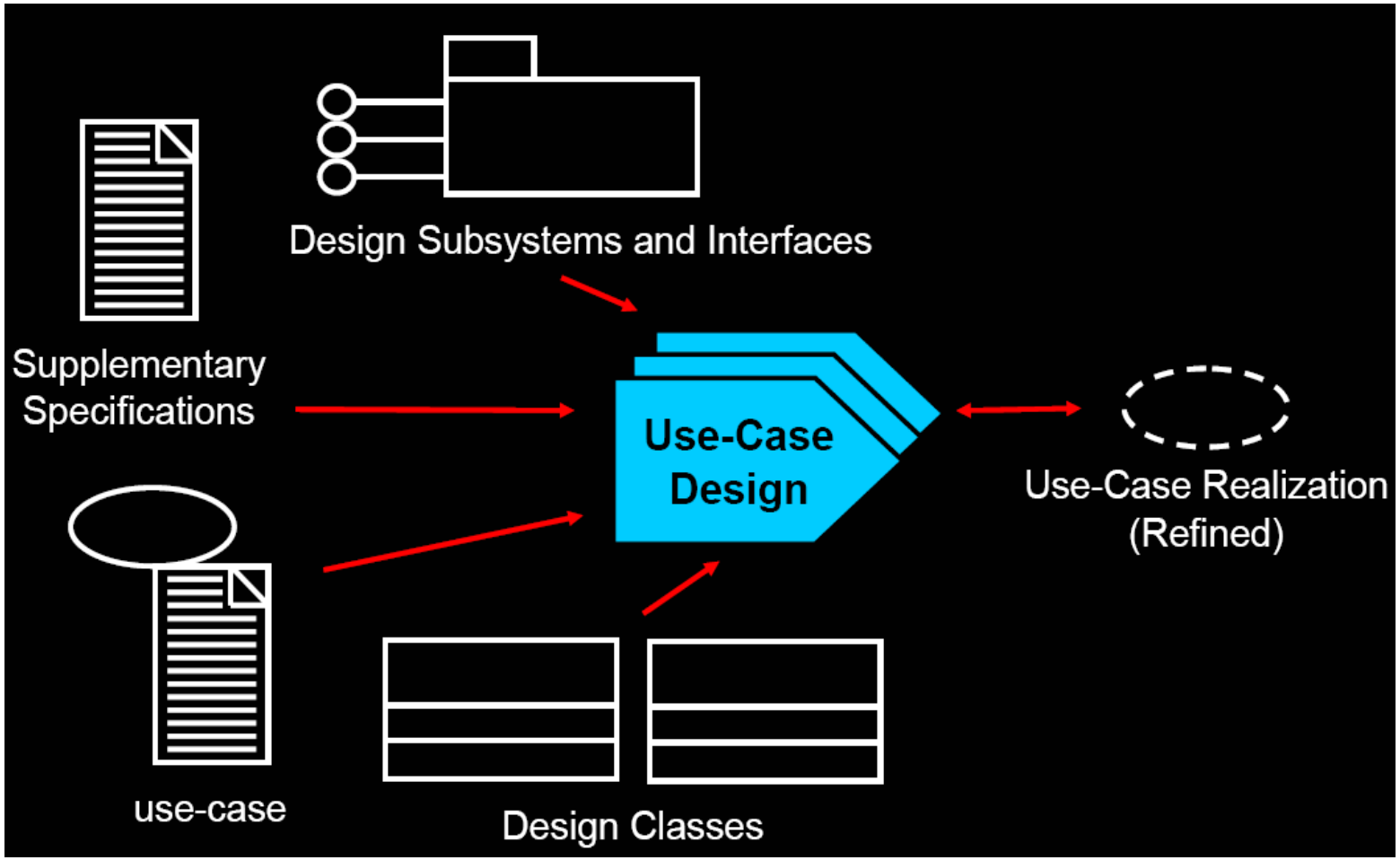
Agenda

- Objectives
- Use-Case Design in Context
- Use-Case Design Steps
- Lab

Use-Case Design in Context



Use-Case Design Overview



Chapter 12 Use-Case Design

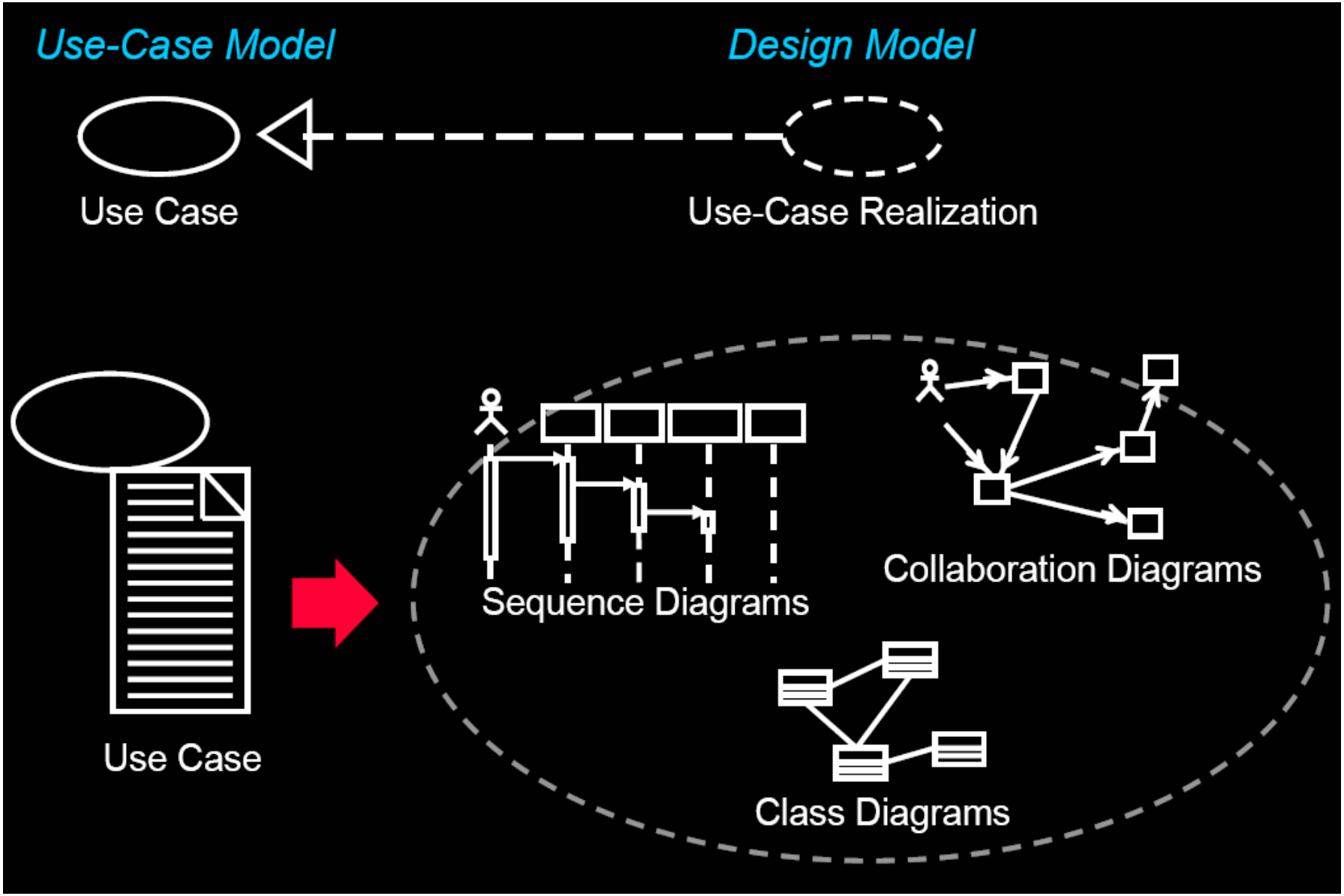
Agenda

- Objectives
- Use-Case Design in Context
- Use-Case Design Steps
- Lab

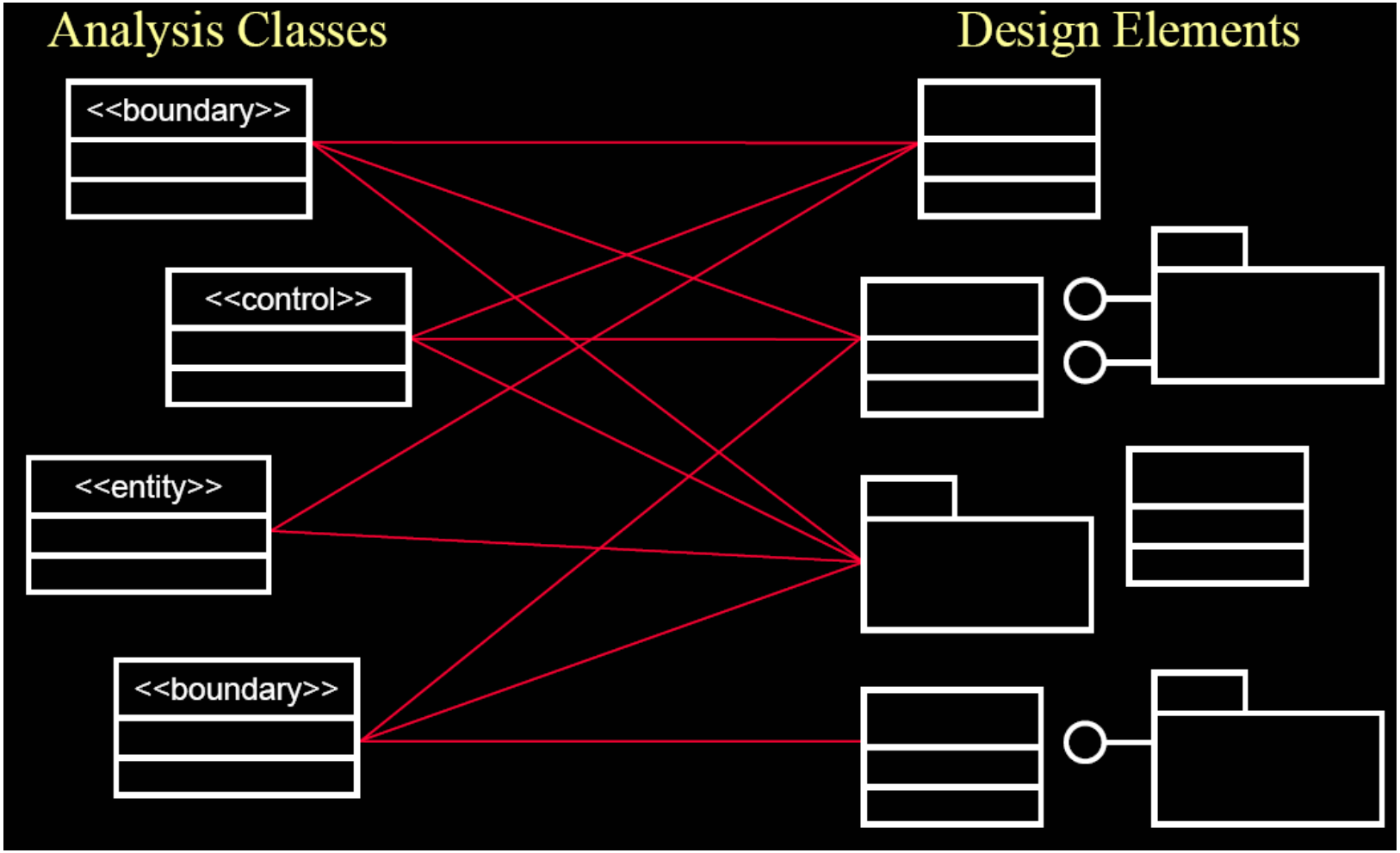
Use-Case Design Steps

- Describe interaction among design objects
- Simplify sequence diagrams using subsystems
- Describe persistence-related behavior
- Refine the flow of events description
- Unify classes and subsystems

Review: Use-Case Realization



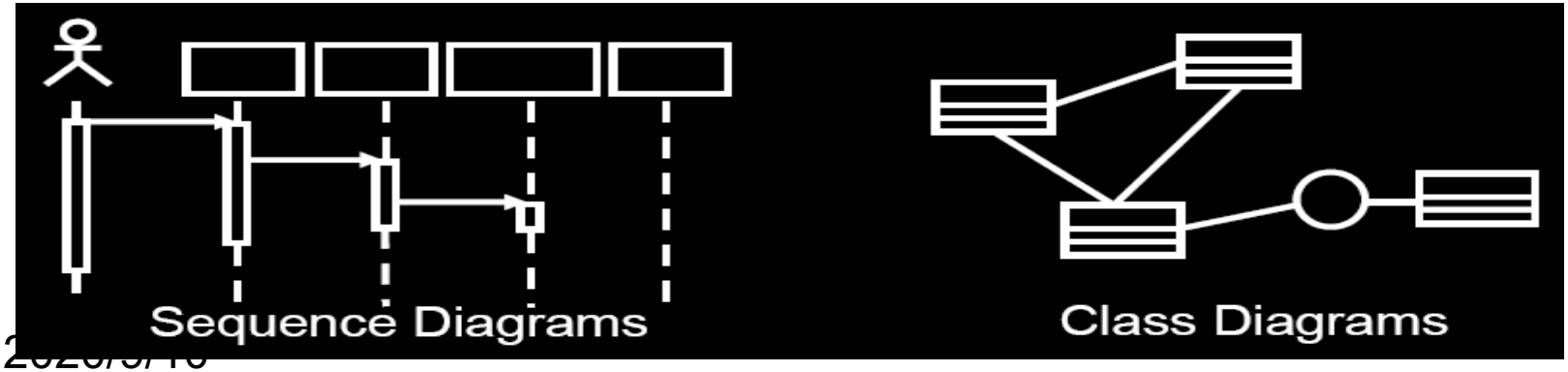
Review: From Analysis Classes to Design Elements



Many-to-Many Mapping

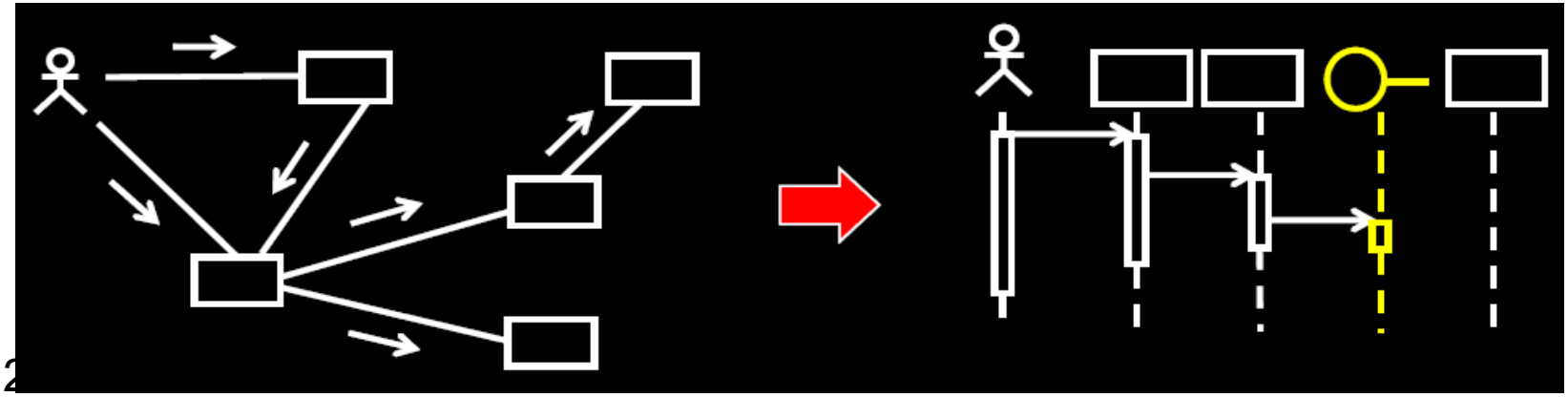
Use-Case Realization Refinement

- Identify participating objects
- Allocate responsibilities among objects
- Model messages between objects
- Describe processing resulting from messages
- Model associated class relationships



Use-Case Realization Refinement Steps

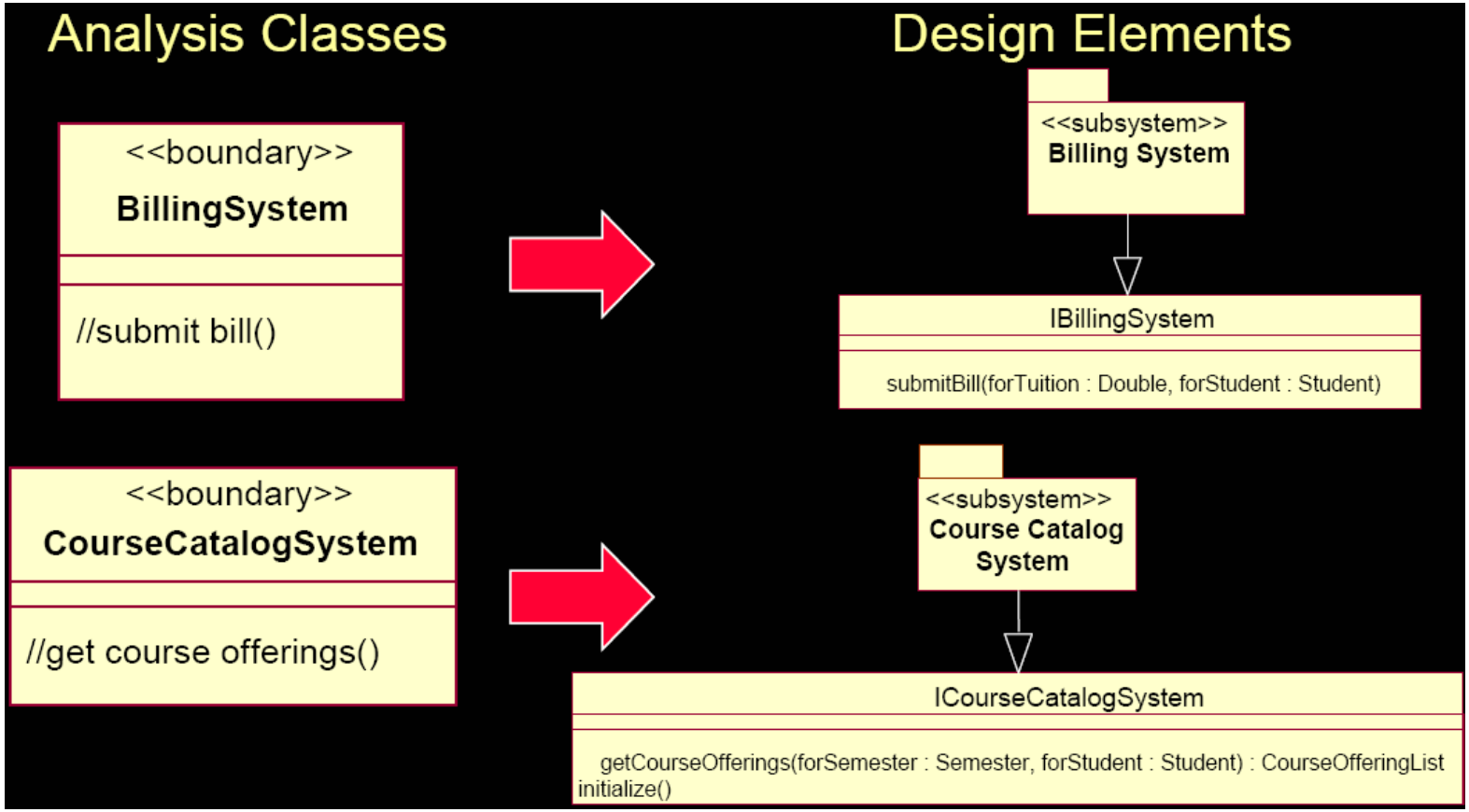
- Identify each object that participates in the flow of the use case
- Represent each participating object in a sequence diagram
- Incrementally incorporate applicable architectural mechanisms



Representing Subsystems on a Sequence Diagram

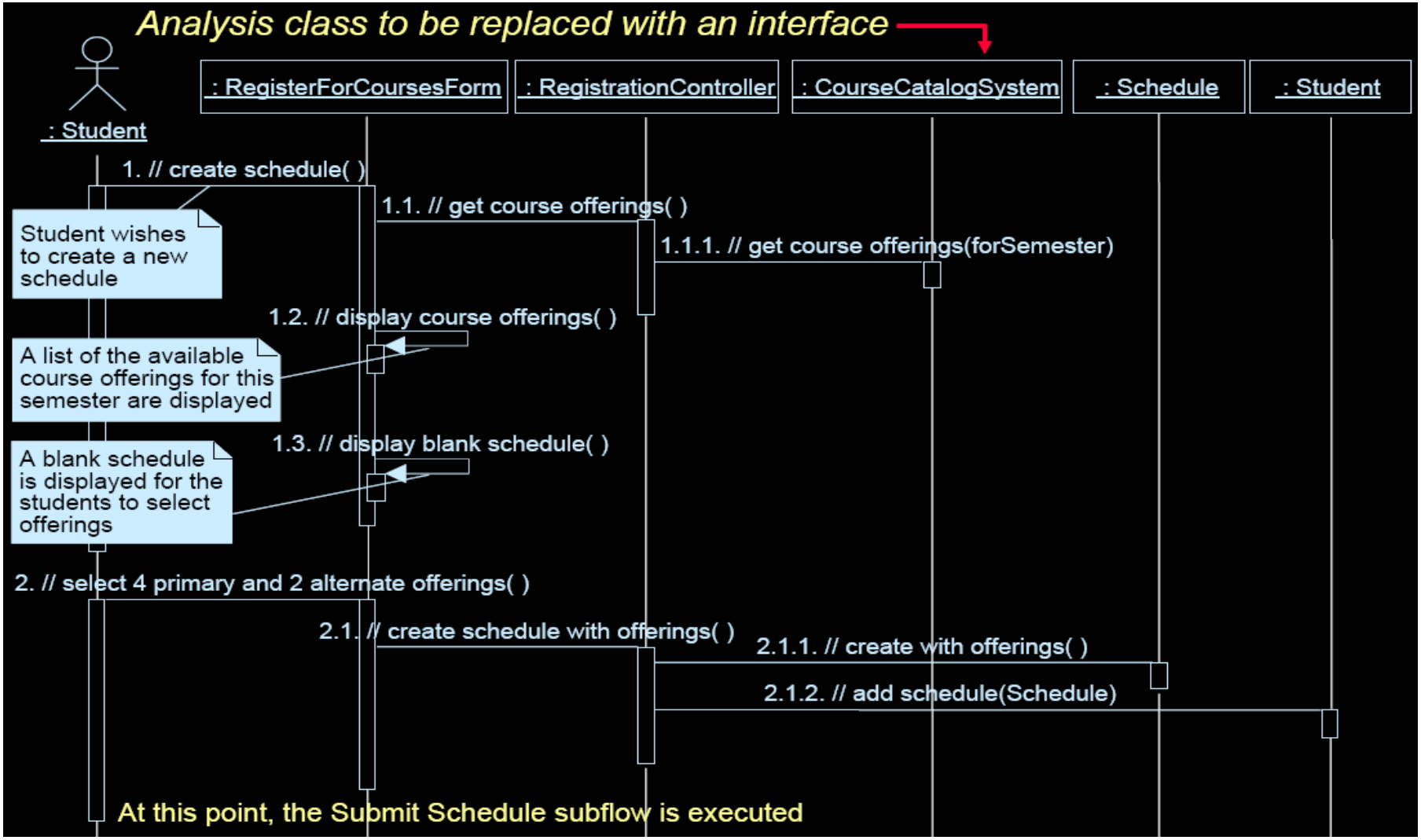
- Interfaces
- ☐ –Represent any model element that realizes the interface
- ☐ – No message should be drawn from the interface
- Proxy class
- ☐ – Represents a specific subsystem
- ☐ – Messages can be drawn from the proxy

Example: Incorporating Subsystem Interfaces

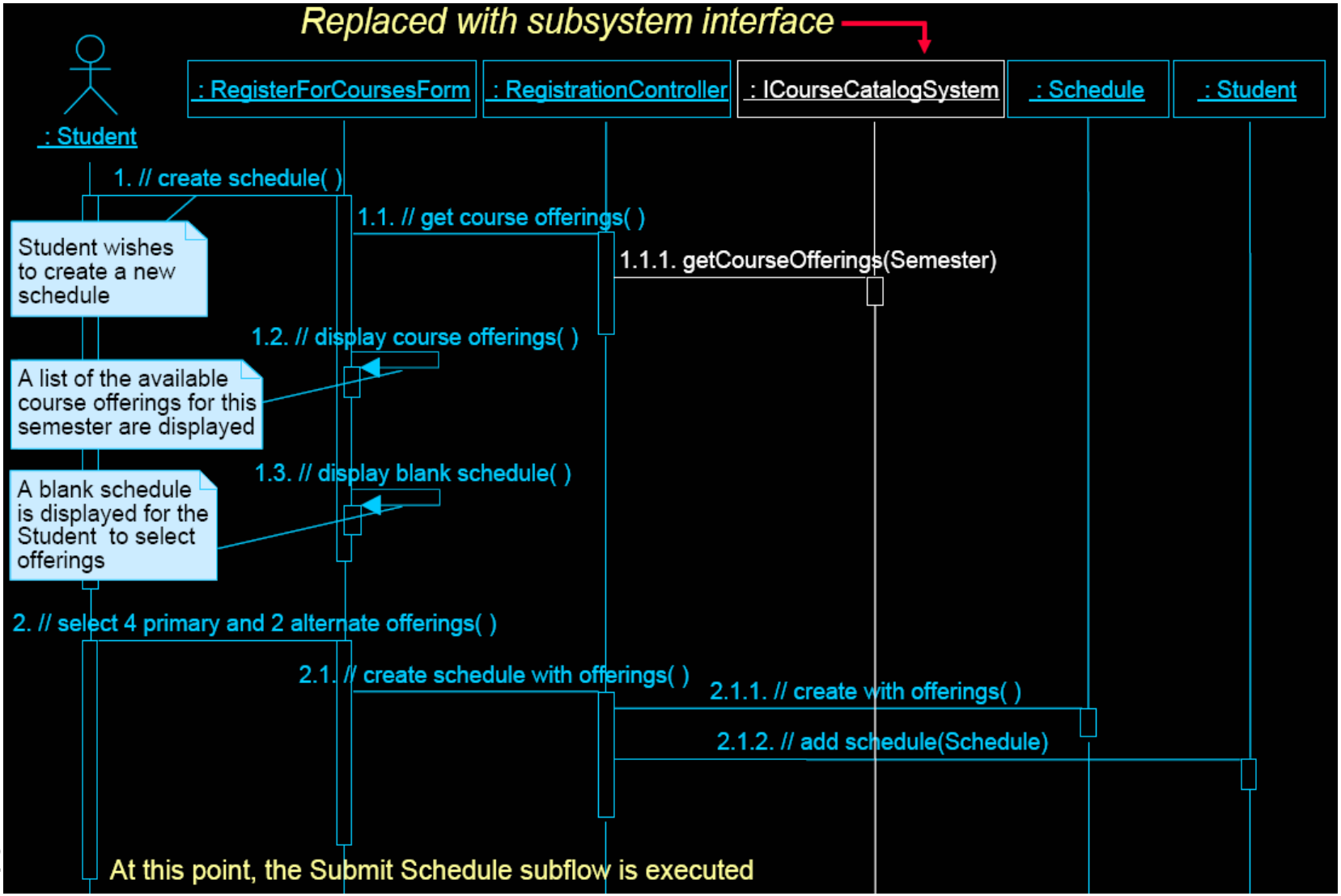


Analysis classes are mapped directly to design classes.

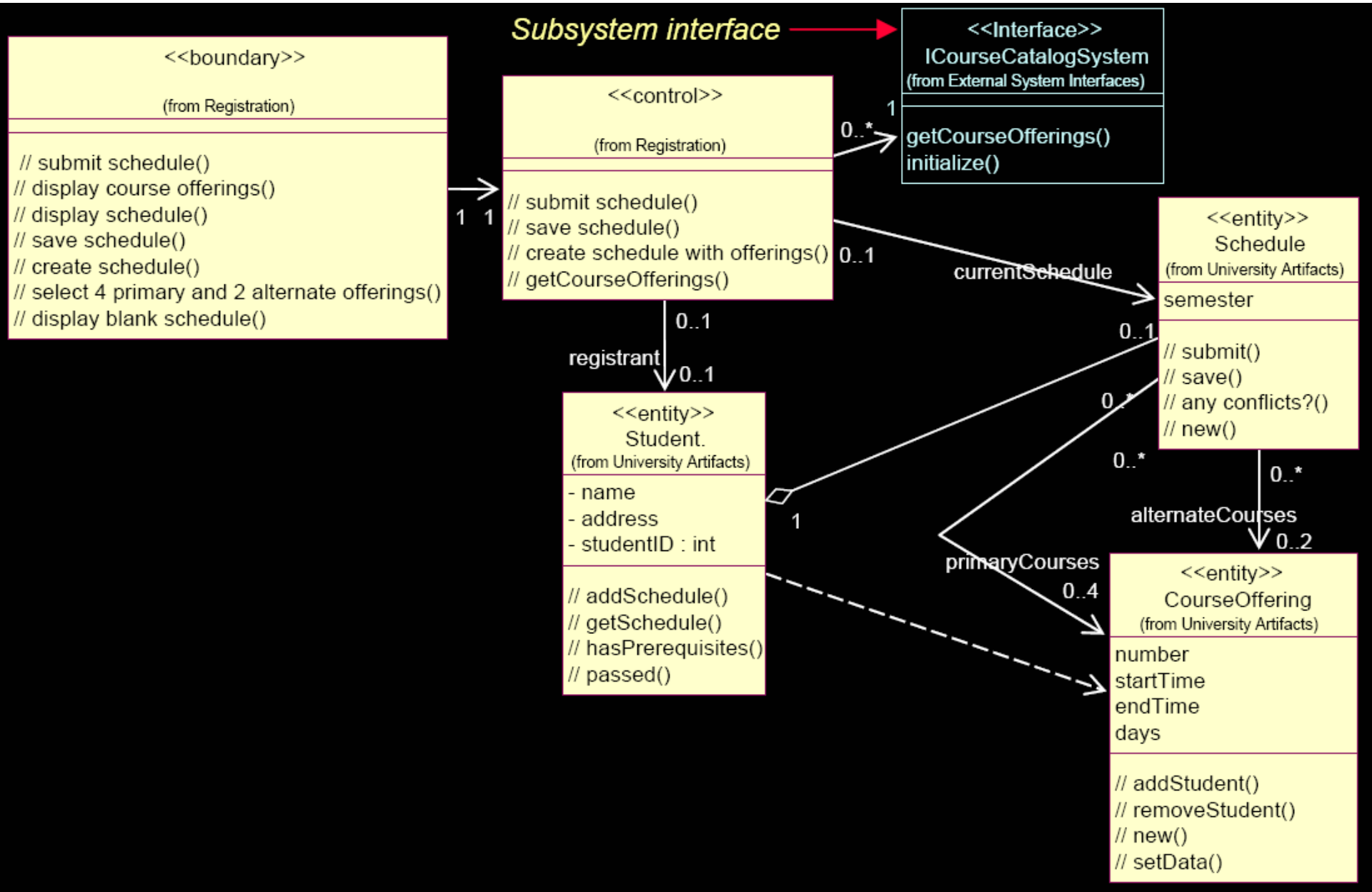
Example: Incorporating Subsystem Interfaces (Before)



Example: Incorporating Subsystem Interfaces (After)



Example: Incorporating Subsystem Interfaces (VOPC)

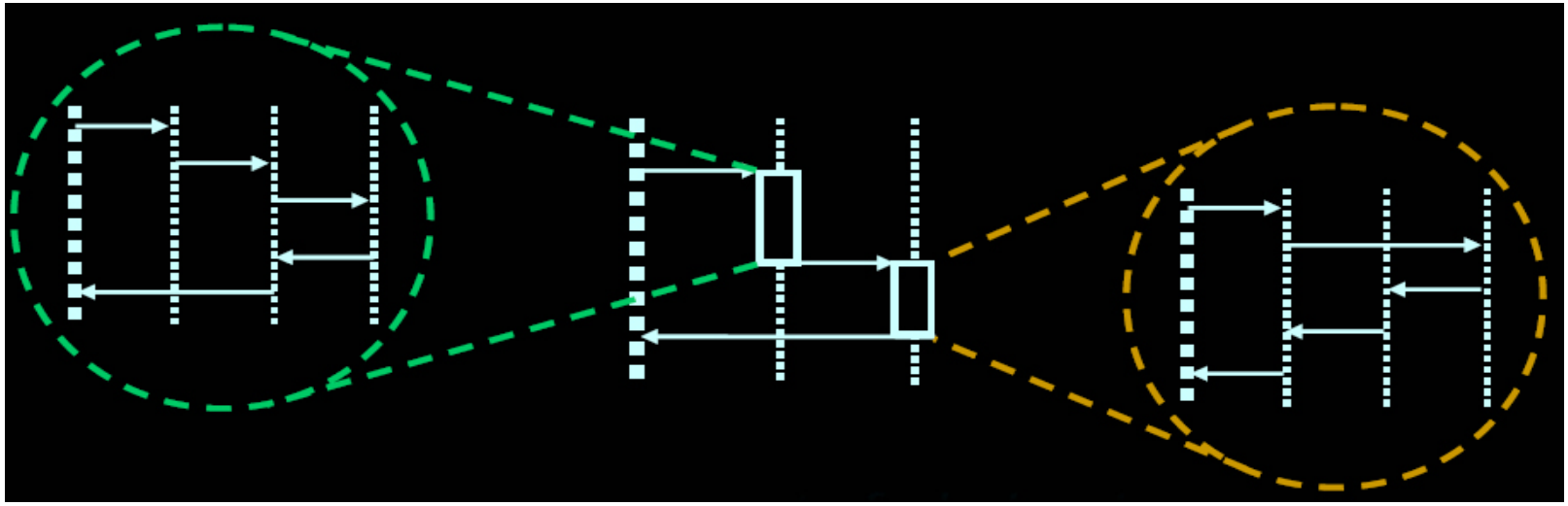


Use-Case Design Steps

- Describe interaction among design objects
- Simplify sequence diagrams using subsystems
- Describe persistence-related behavior
- Refine the flow of events description
- Unify classes and subsystems

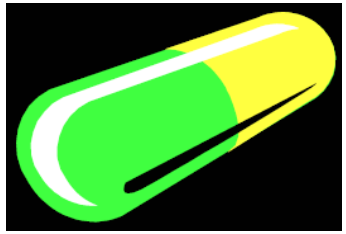
Encapsulating Subsystem Interactions

- Interactions can be described at several levels
- Subsystem interactions can be described in their own interaction diagrams



Raises the level of abstraction

When to Encapsulate Subflows in a Subsystem



- Encapsulate a Subflow when it:

☐ – Occurs in multiple use-case realizations

☐ – Has reuse potential

☐ – Is complex and easily encapsulated

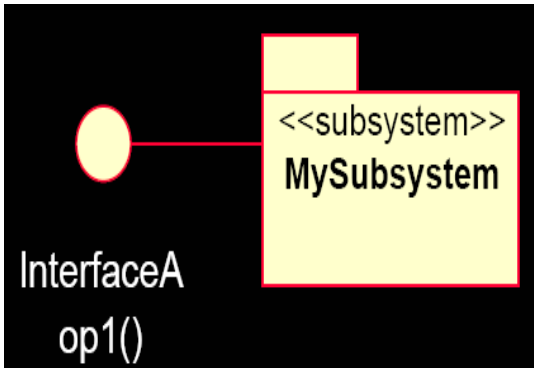
☐ – Is responsibility of one person or team

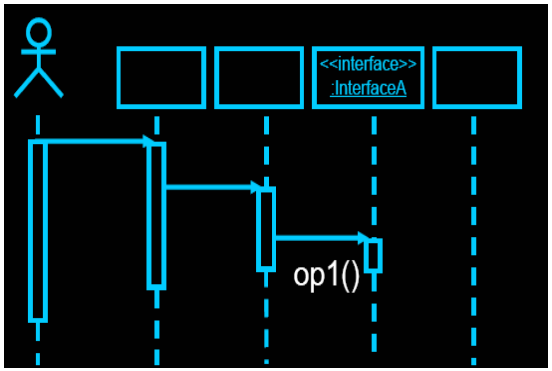
☐ – Produces a well-defined result

☐ – Is encapsulated within a single Implementation Model component

Guidelines: Encapsulating Subsystem Interactions

- Subsystems should be represented by their interfaces on interaction diagrams
- Messages to subsystems are modeled as messages to the subsystem interface
- Messages to subsystems correspond to operations of the subsystem interface
- Interactions within subsystems are modeled in Subsystem Design





Advantages of Encapsulating Subsystem Interactions

- Use-case realizations:
  - Are less cluttered
  - Can be created before the internal designs of subsystems are created (parallel development)
  - Are more generic and easier to change (Subsystems can be substituted.)

Parallel Subsystem Development

- Concentrate on requirements that affect subsystem interfaces
- Outline required interfaces
- Model messages that cross subsystem boundaries
- Draw interaction diagrams in terms of subsystem interfaces for each use case
- Refine the interfaces needed to provide messages
- Develop each subsystem in parallel

Use subsystem interfaces as synchronization points

Use-Case Design Steps

- Describe interaction among design objects
- Simplify sequence diagrams using subsystems
- Describe persistence-related behavior
- Refine the flow of events description
- Unify classes and subsystems

Use-Case Design Steps: Describe Persistence-Related Behavior

- Describe Persistence-Related Behavior

☐– Modeling Transactions

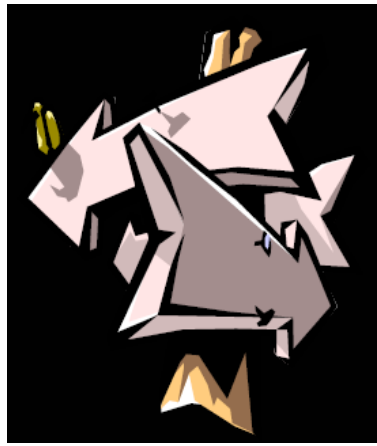
  – Writing Persistent Objects

☐– Reading Persistent Objects

☐– Deleting Persistent Objects

Modeling Transactions

- What is a transaction?
  - Atomic operation invocations
  - “All or nothing”
  - Provide consistency
- Modeling options
  - Textually (scripts)
  - Explicit messages
- Error conditions
  - Rollback
  - Failure modes
  - May require separate interaction diagrams

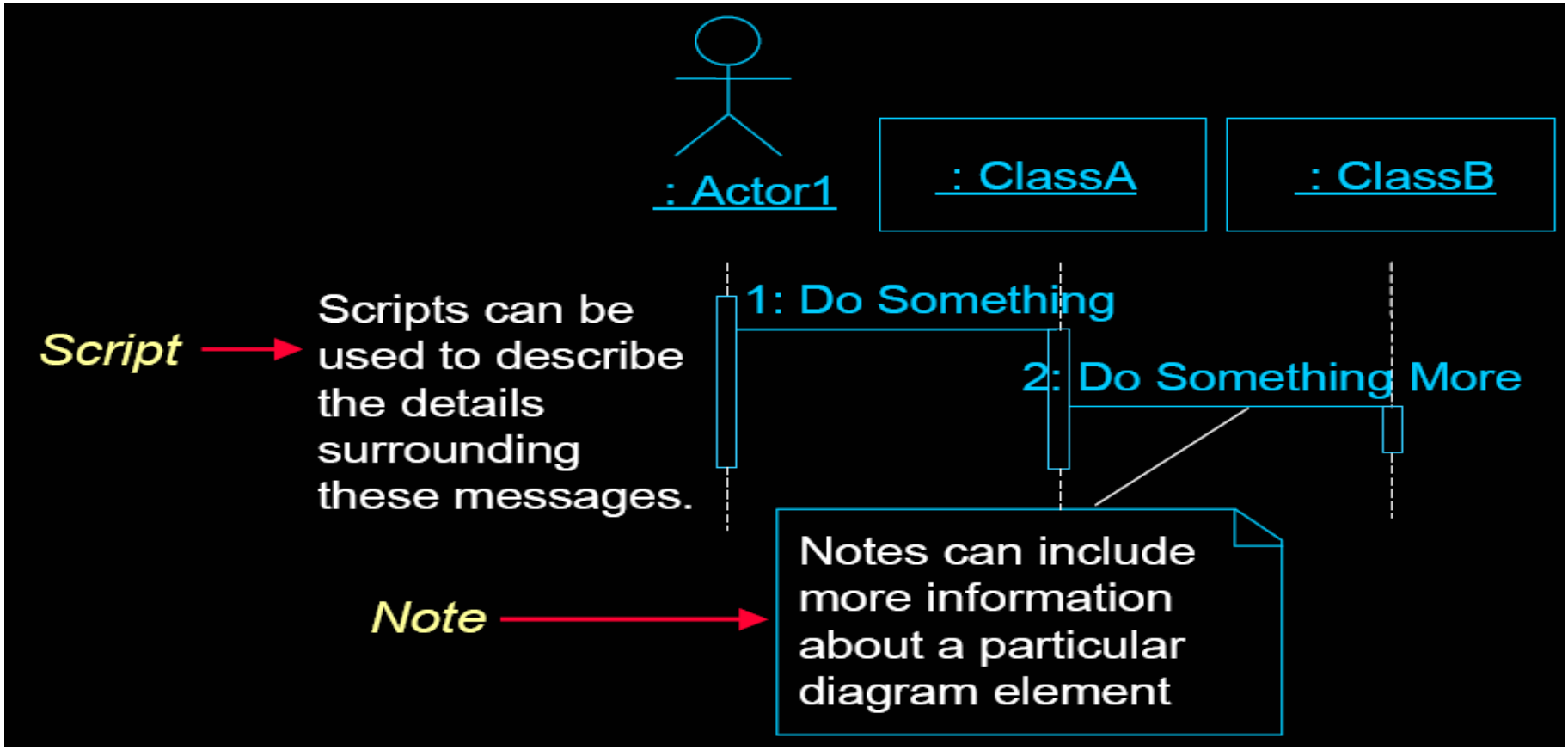


Use-Case Design Steps

- Describe interaction among design objects
- Simplify sequence diagrams using subsystems
- Describe persistence-related behavior
- Refine the flow of events description
- Unify classes and subsystems

Detailed Flow of Events Description Options

- Annotate the interaction diagrams

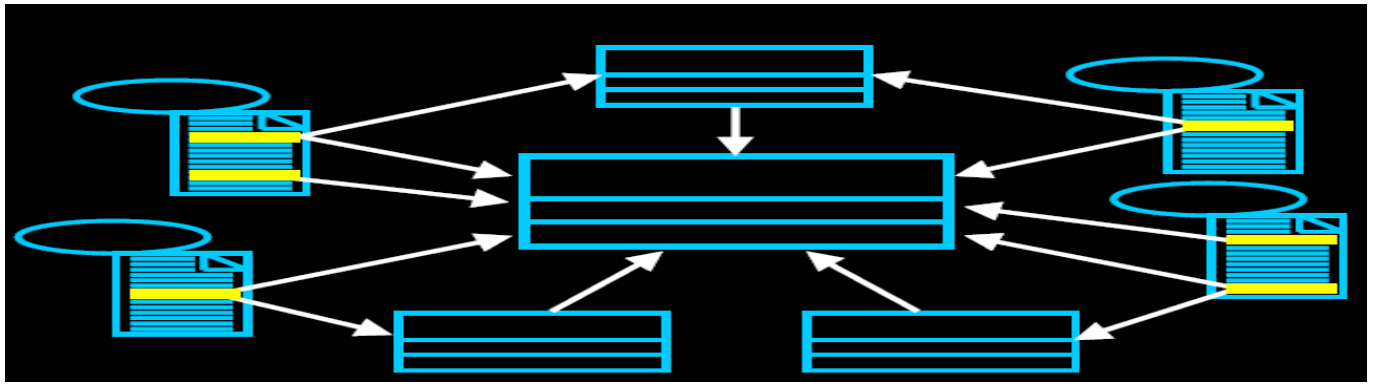


Use-Case Design Steps

- Describe interaction among design objects
- Simplify sequence diagrams using subsystems
- Describe persistence-related behavior
- Refine the flow of events description
- Unify classes and subsystems

Design Model Unification Considerations

- Model element names should describe their function
- Merge similar model elements
- Use inheritance to abstract model elements
- Keep model elements and flows of events consistent



Checkpoints: Use-Case Design

- Is package/subsystem partitioning logical and consistent?
- Are the names of the packages/subsystems descriptive?
- Do the public package classes and subsystem interfaces provide a single, logically consistent set of services?
- Do the package/subsystem dependencies correspond to the relationships between the contained classes?
- Do the classes contained in a package belong there according to the criteria for the package division?
- Are there classes or collaborations of classes that can be separated into an independent package/subsystem?

Checkpoints: Use-Case Design

- Have all the main and/or subflow for this iteration been handled?
- Has all behavior been distributed among the participating design elements?
- Has behavior been distributed to the right design elements?
- If there are several interaction diagrams for the use-case realization, is it easy to understand which collaboration diagrams relate to which flow of events?

作业

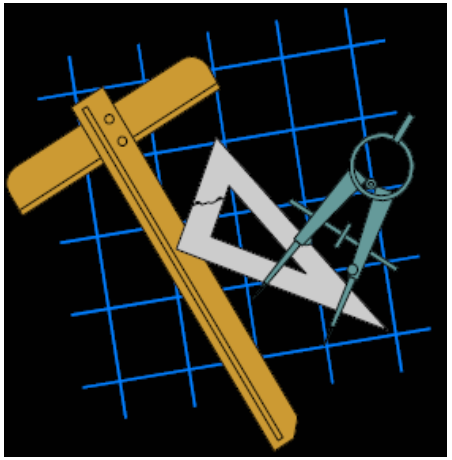
- What is the purpose of Use-Case Design?
- What is meant by encapsulating subsystem interactions? Why is it a good thing to do?

Chapter 12 Use-Case Design

Agenda

- Objectives
- Use-Case Design in Context
- Use-Case Design Steps
- Lab

Lab: Use-Case Design



- Given the following:
  - Analysis use-case realizations

    (VOPCs and interaction diagrams)
  - The analysis-class-to-design element map

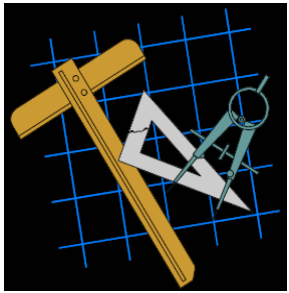
Lab: Use-Case Design (cont.)

- Identify the following:

–The design elements that replaced the analysis classes in the analysis use-case realizations

– The design element collaborations needed to implement the use case

– The relationships between the design elements needed to support the collaborations



Lab: Use-Case Design (cont.)

- Produce the following:
  - Design use-case realization
    - Interaction diagram(s) per use case flow of events that describes the design element collaborations required to implement the use case
    - Class diagram (VOPC) that includes the design elements that must collaborate to perform the use case, and their relationships

Lab: Review

- Compare your use-case realizations
  - Have all the main and subflows for this iteration 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?