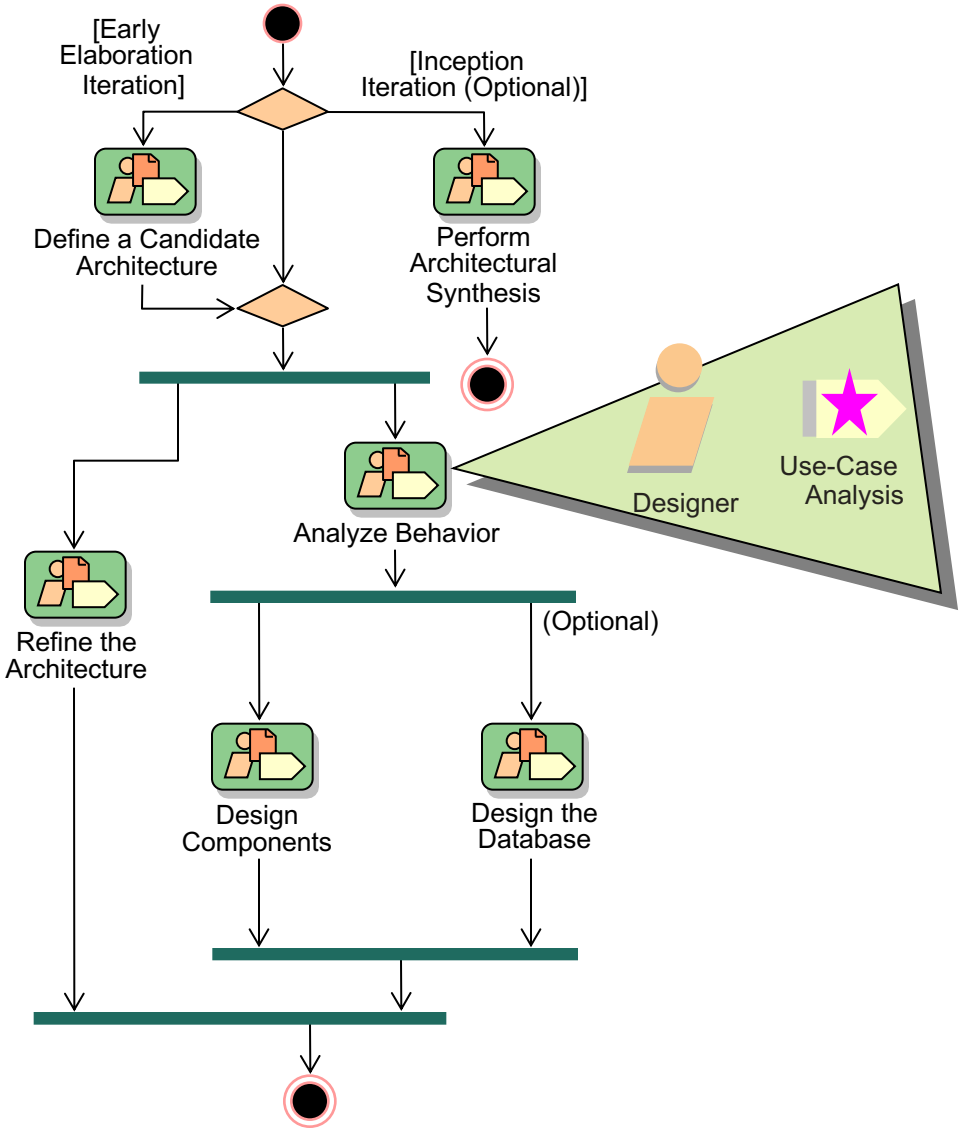
Software analysis and design

Module 10: Use Case Analysis

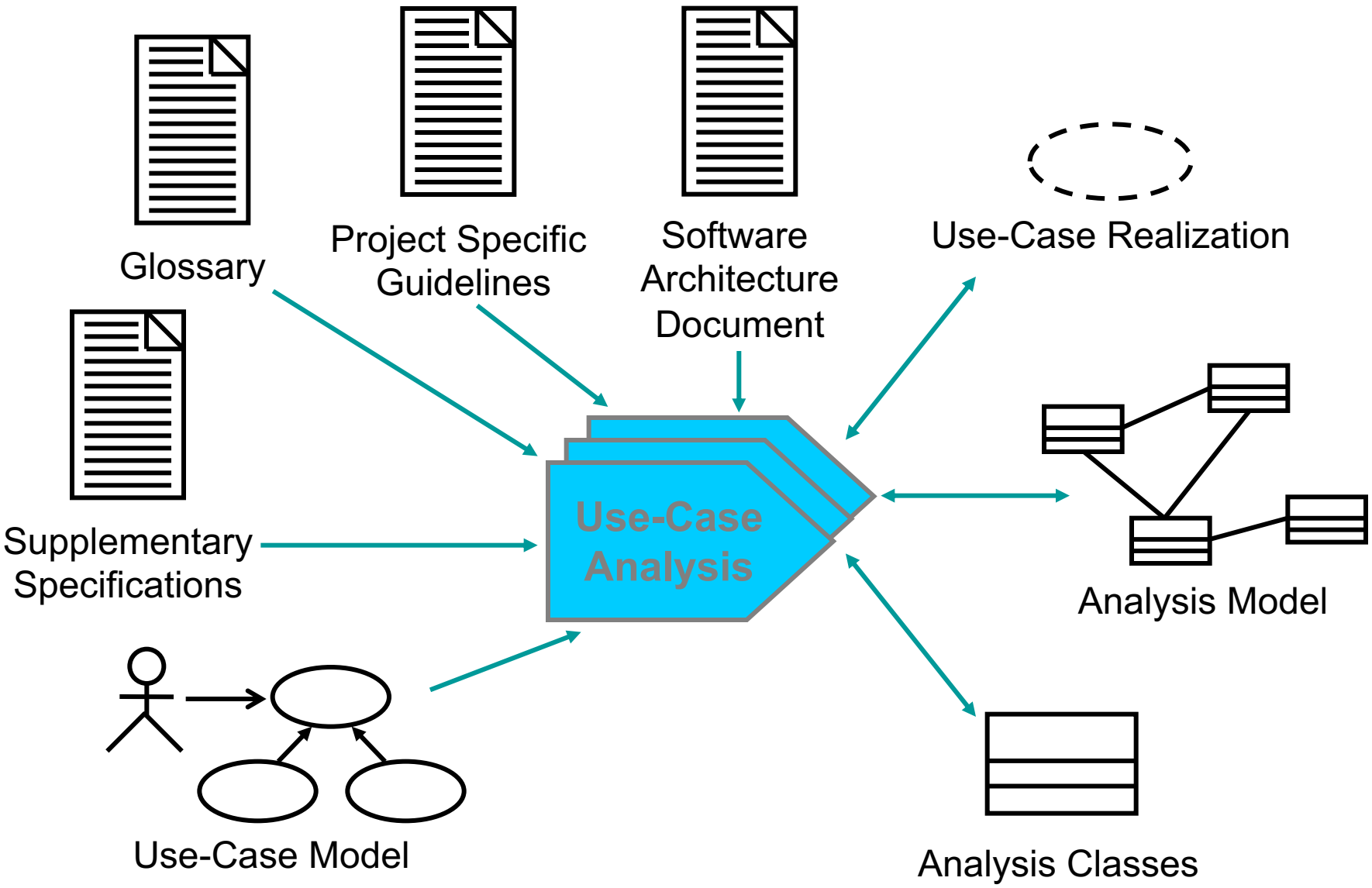
Objectives: Use-Case Analysis

- Explain the purpose of Use-Case Analysis and where in the lifecycle it is performed
- Identify the classes which perform a use-case flow of events
- Distribute the use-case behavior to those classes, identifying responsibilities of the classes
- Develop Use-Case Realizations that model the collaborations between instances of the identified classes

Use-Case Analysis in Context



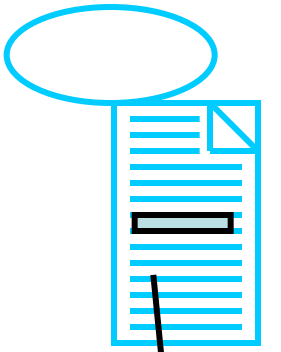
Use-Case Analysis Overview

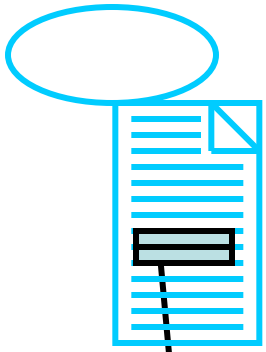


Use-Case Analysis Steps

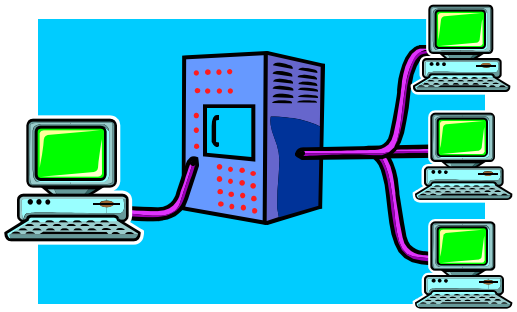
- Supplement the Use-Case Description
- For each Use-Case Realization
  - Find Classes from Use-Case Behavior
  - Distribute Use-Case Behavior to Classes
- For each resulting analysis class
  - Describe Responsibilities
  - Describe Attributes and Associations
  - Qualify Analysis Mechanisms
- Unify Analysis Classes
- Checkpoints

Supplement the Use-Case Description





- The system displays a list of course offerings.

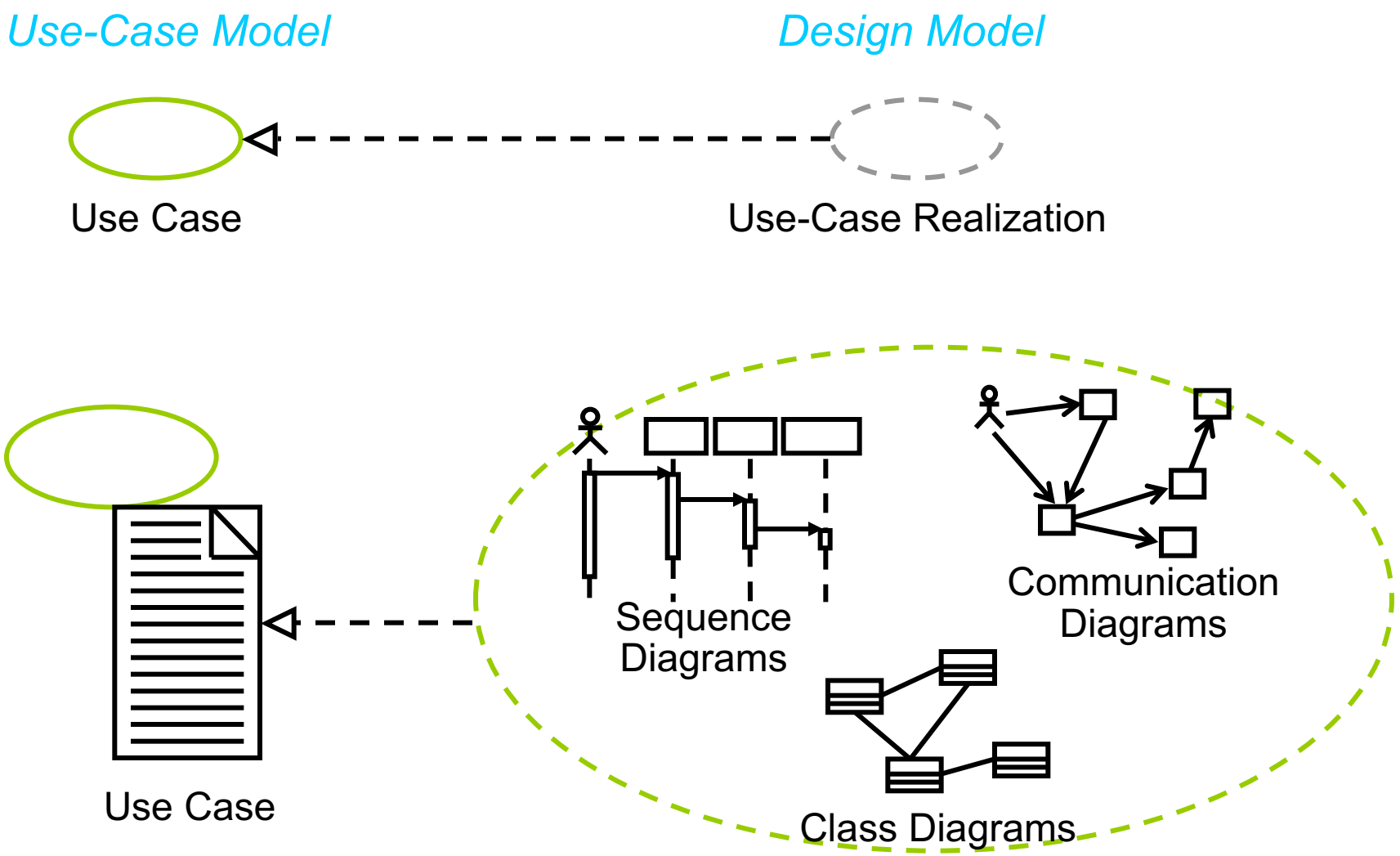


- The system retrieves and displays a list of current course offerings from the course catalog legacy database.

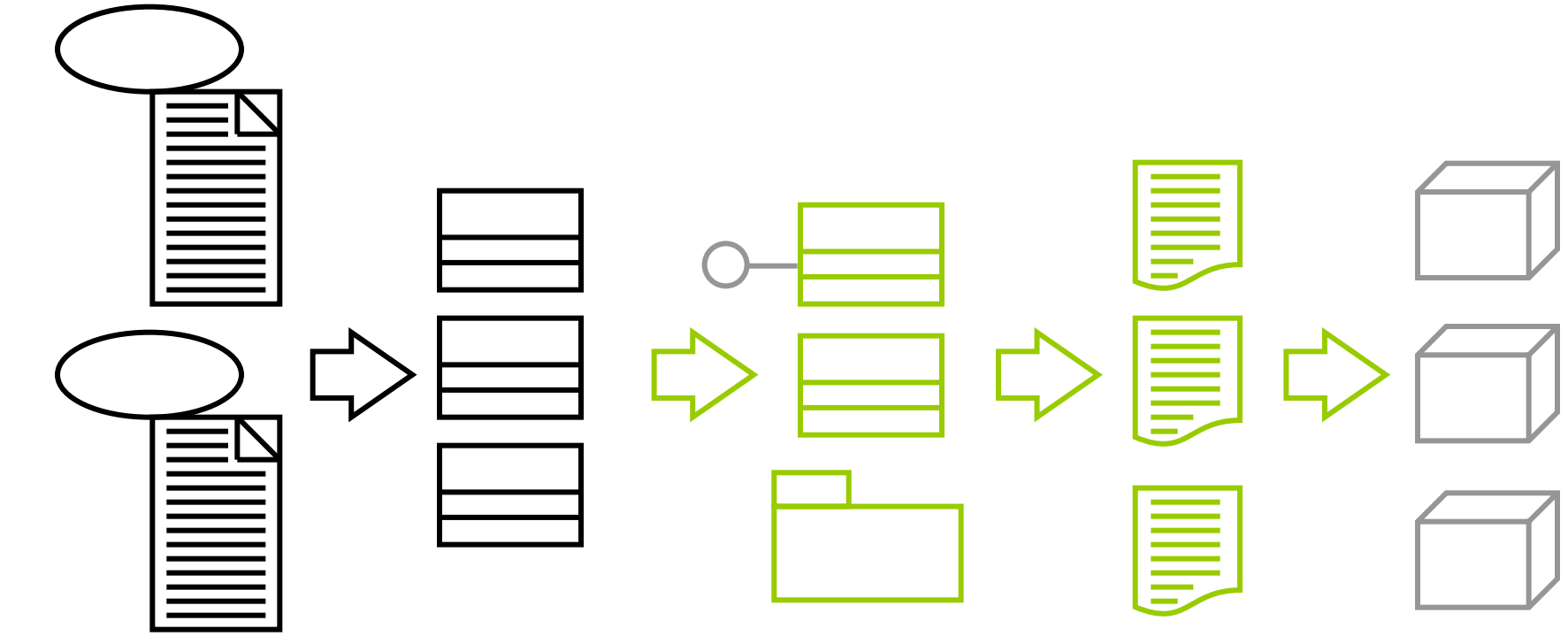
Use-Case Analysis Steps

- Supplement the Use-Case Description
- For each Use-Case Realization
  - Find Classes from Use-Case Behavior
  - Distribute Use-Case Behavior to Classes
- For each resulting analysis class
  - Describe Responsibilities
  - Describe Attributes and Associations
  - Qualify Analysis Mechanisms
- Unify Analysis Classes
- Checkpoints

Review: Use-Case Realization

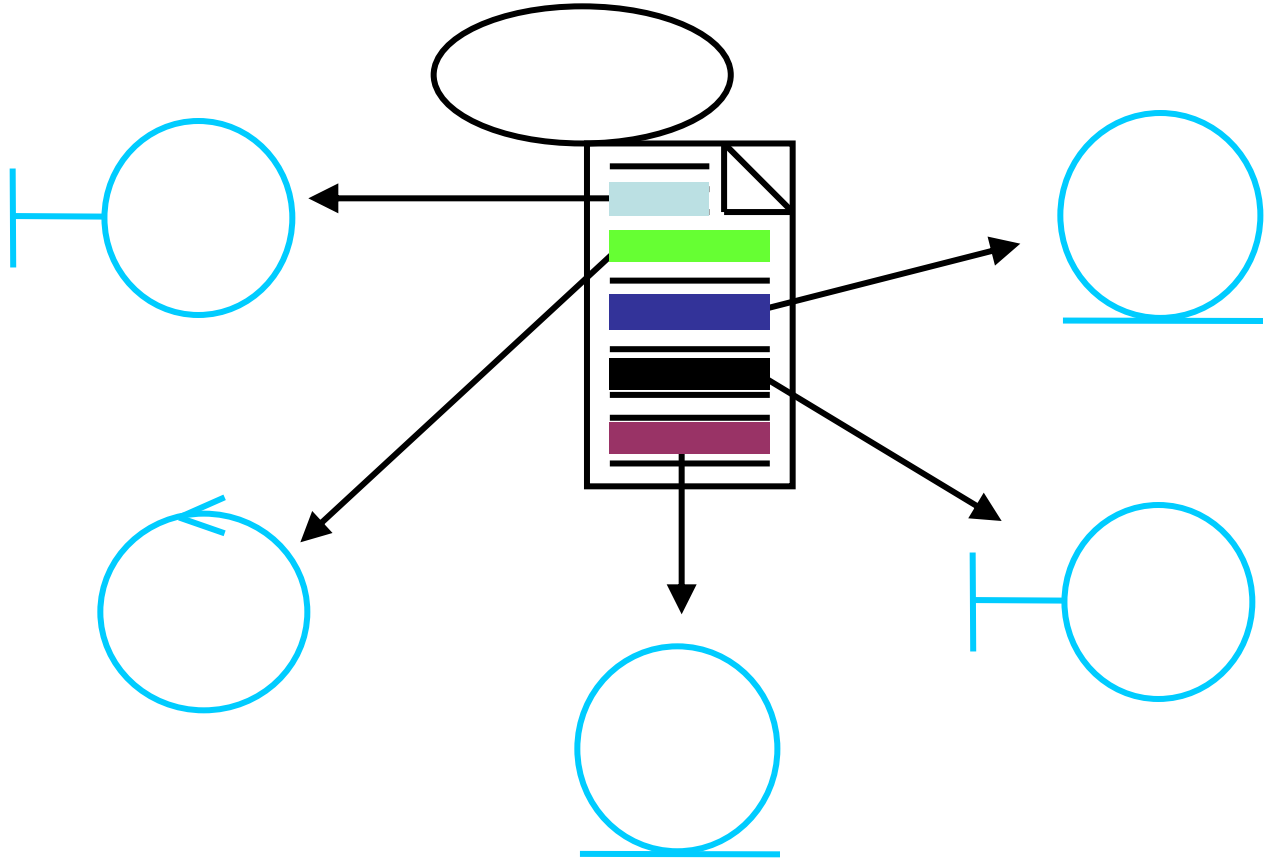


Analysis Classes: A First Step Toward Executables

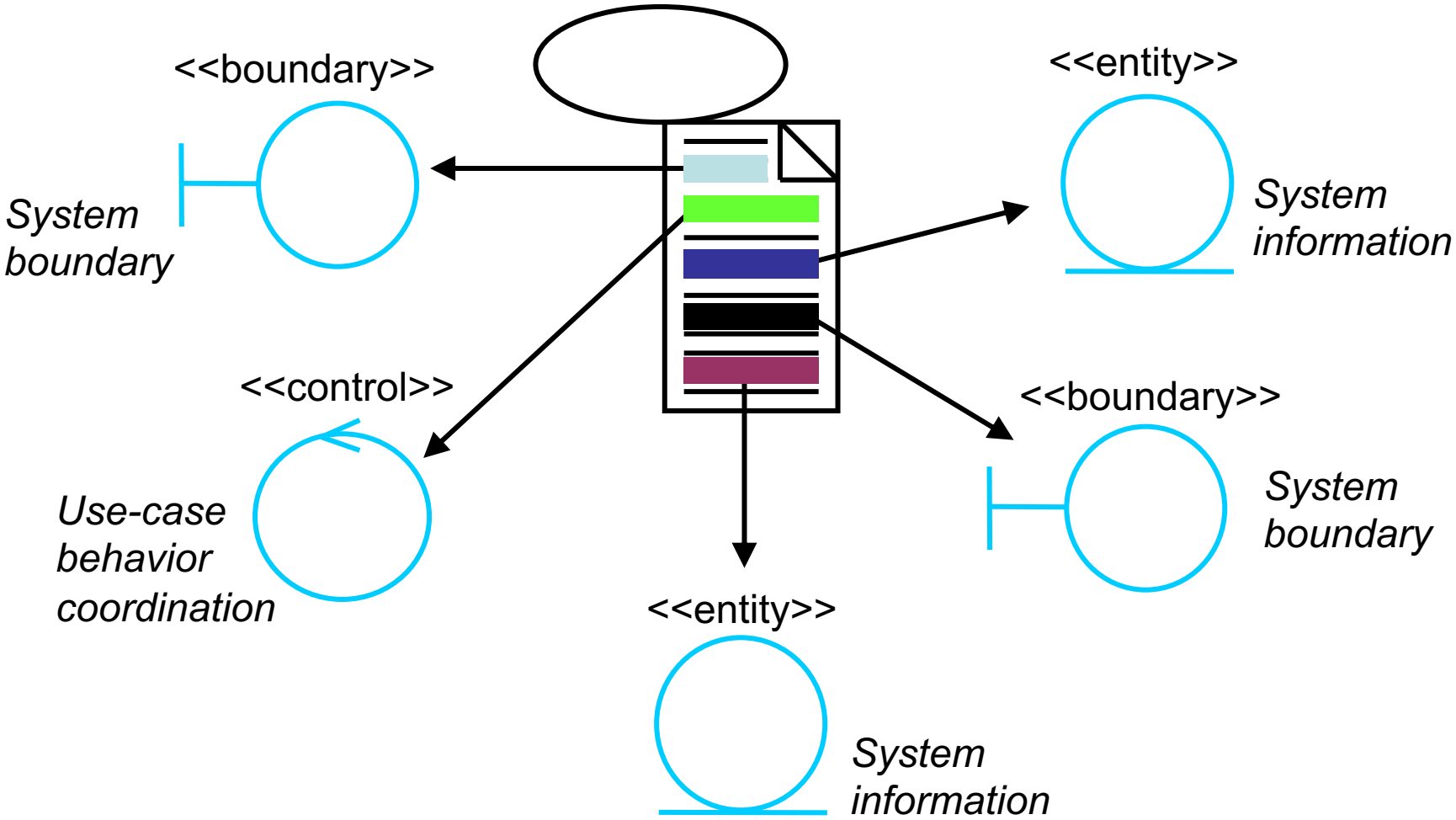


Find Classes from Use-Case Behavior

- The complete behavior of a use case has to be distributed to analysis classes

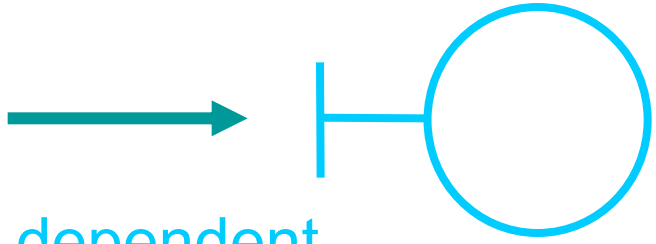


What Is an Analysis Class?



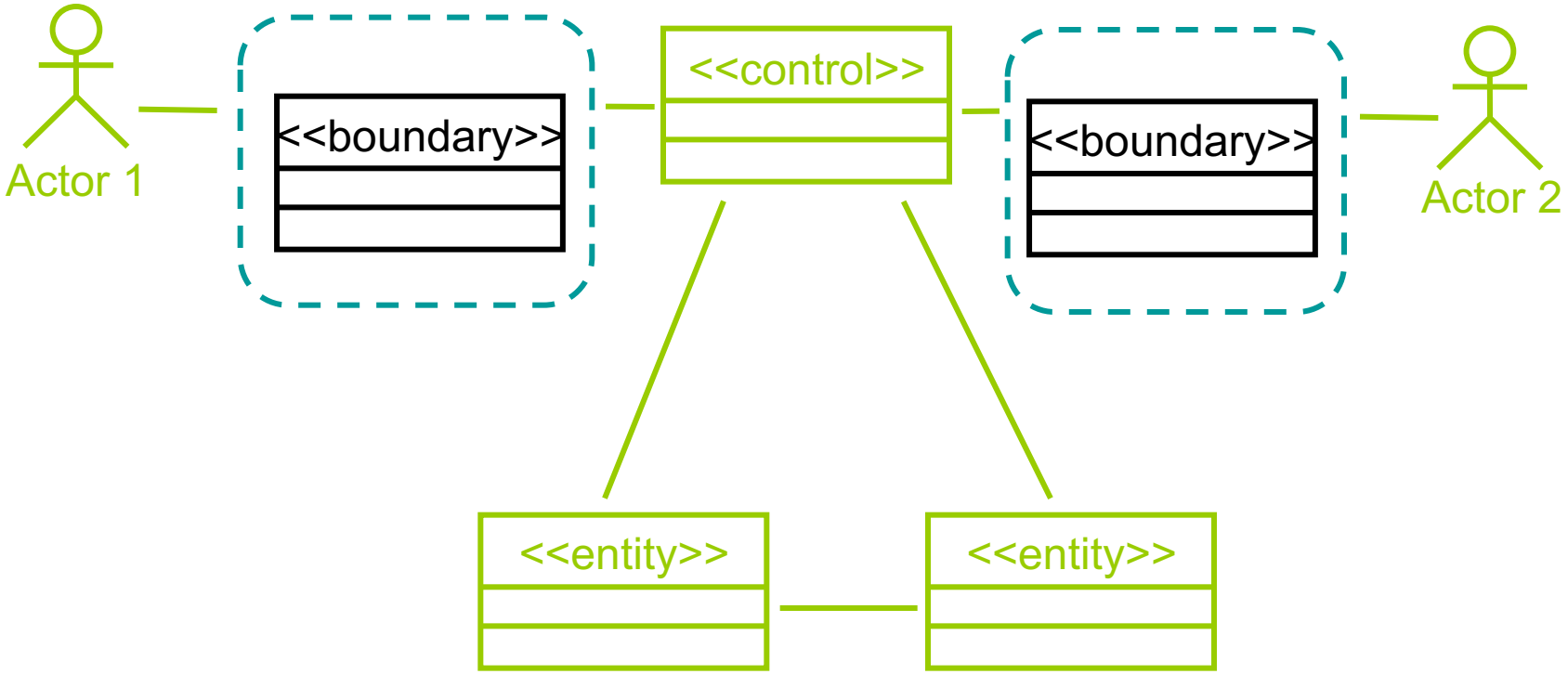
What Is a Boundary Class?

- Intermediates between the interface and something outside the system
- Several Types
  - User interface classes
  - System interface classes
  - Device interface classes
- *One boundary class per actor/use-case pair*

Analysis class stereotype 

Environment dependent.

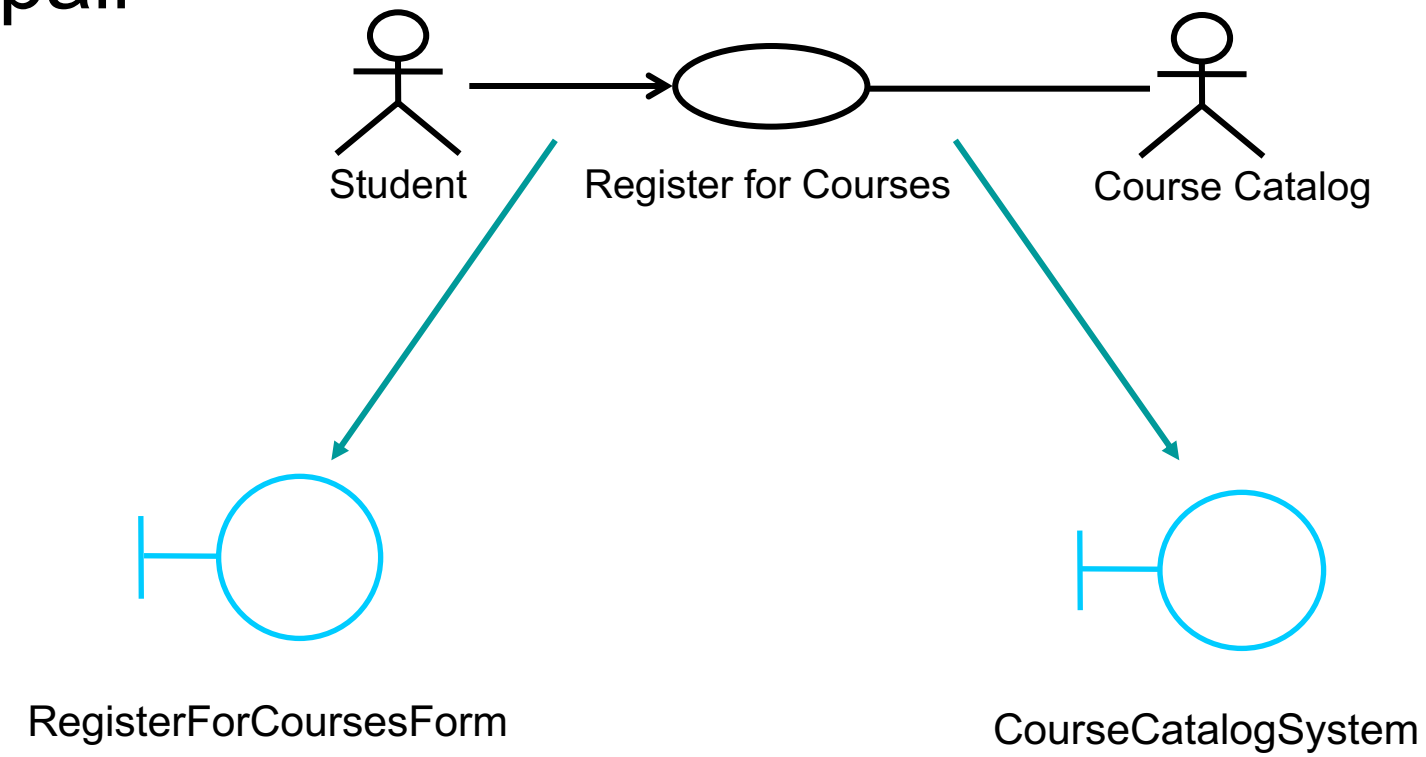
The Role of a Boundary Class



Model interaction between the system and its environment.

Example: Finding Boundary Classes

- One boundary class per actor/use case pair



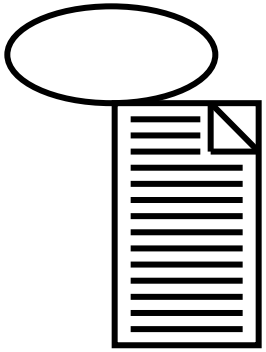
Guidelines: Boundary Class

- User Interface Classes
  - Concentrate on what information is presented to the user
  - Do NOT concentrate on the UI details
- System and Device Interface Classes
  - Concentrate on what protocols must be defined
  - Do NOT concentrate on how the protocols will be implemented

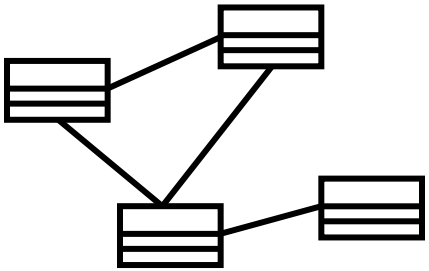
Concentrate on the responsibilities, not the details!

What Is an Entity Class?

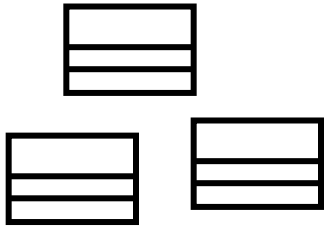
- Key abstractions of the system



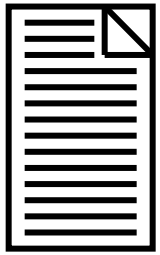
Use Case



Business-Domain Model

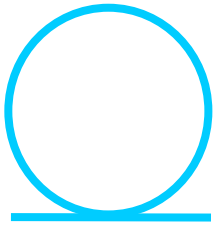


Architectural Analysis Abstractions



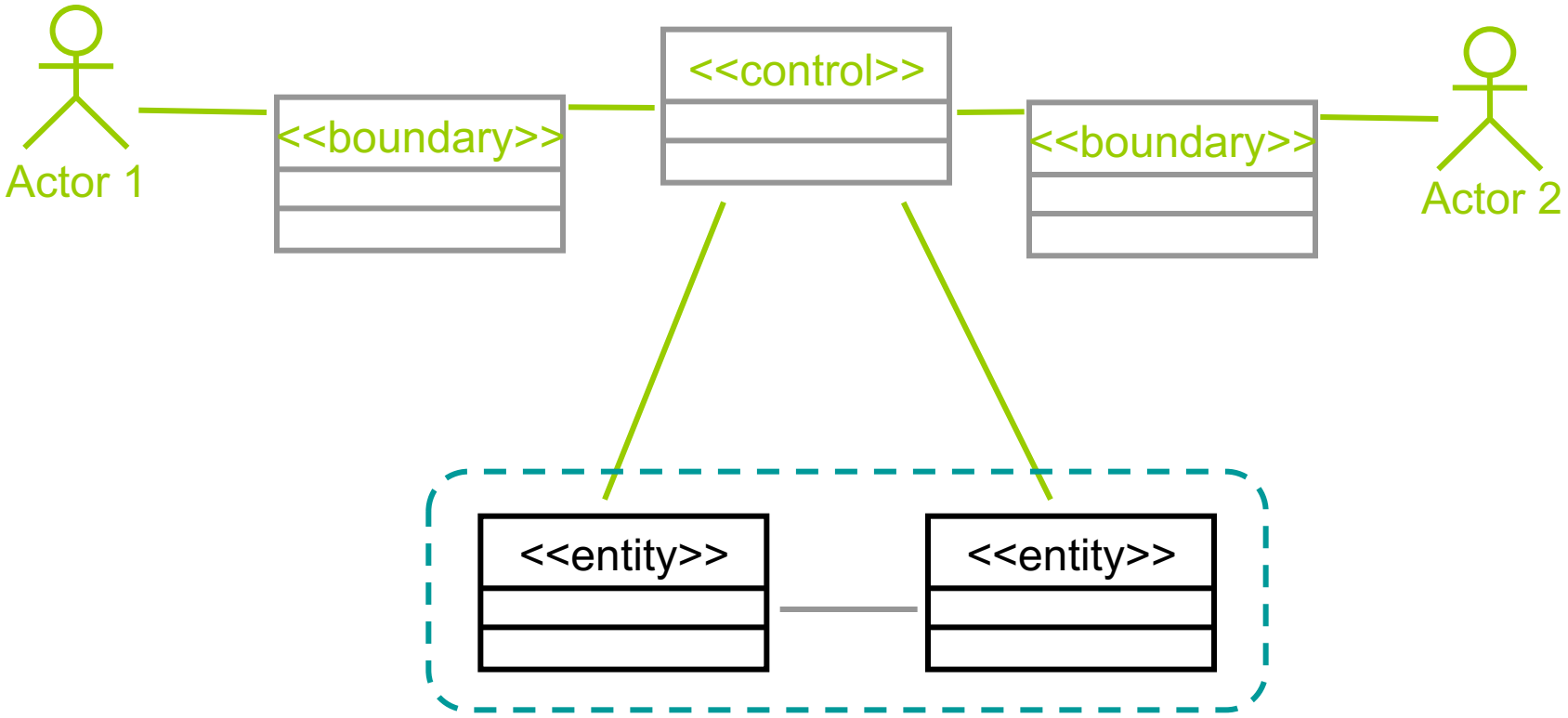
Glossary

Analysis class stereotype



Environment independent.

The Role of an Entity Class



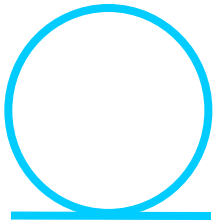
Store and manage information in the system.

Example: Finding Entity Classes

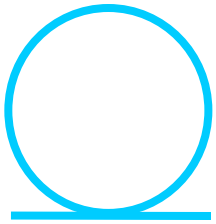
- Use use-case flow of events as input
- Key abstractions of the use case
- Traditional, filtering nouns approach
  - Underline noun clauses in the use-case flow of events
  - Remove redundant candidates
  - Remove vague candidates
  - Remove actors (out of scope)
  - Remove implementation constructs
  - Remove attributes (save for later)
  - Remove operations

Example: Candidate Entity Classes

- Register for Courses (Create Schedule)



CourseOffering

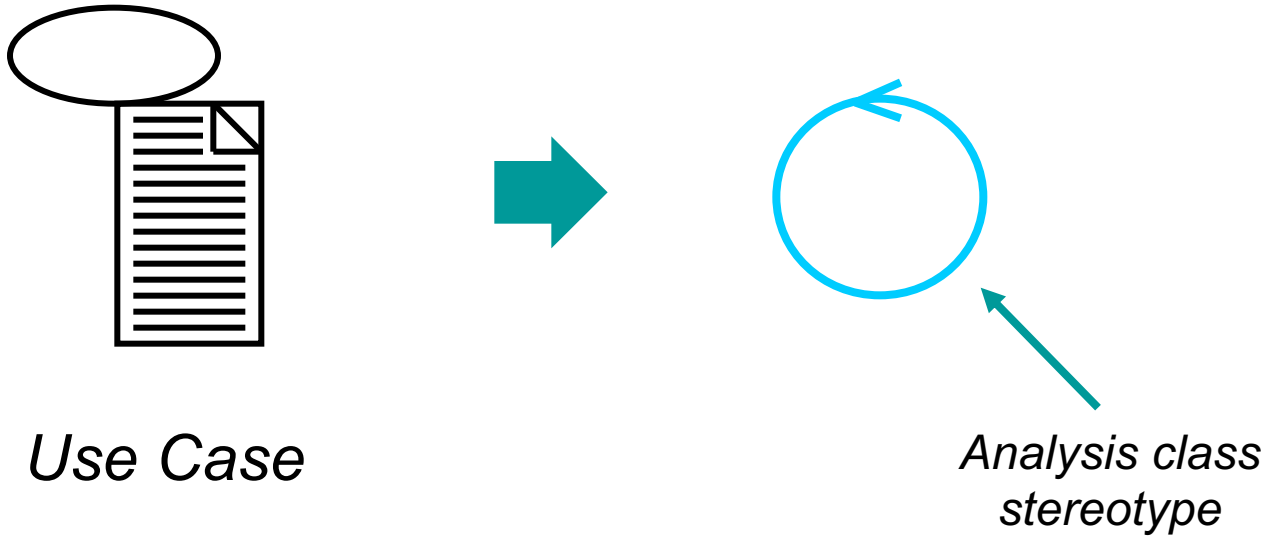


Schedule

Student

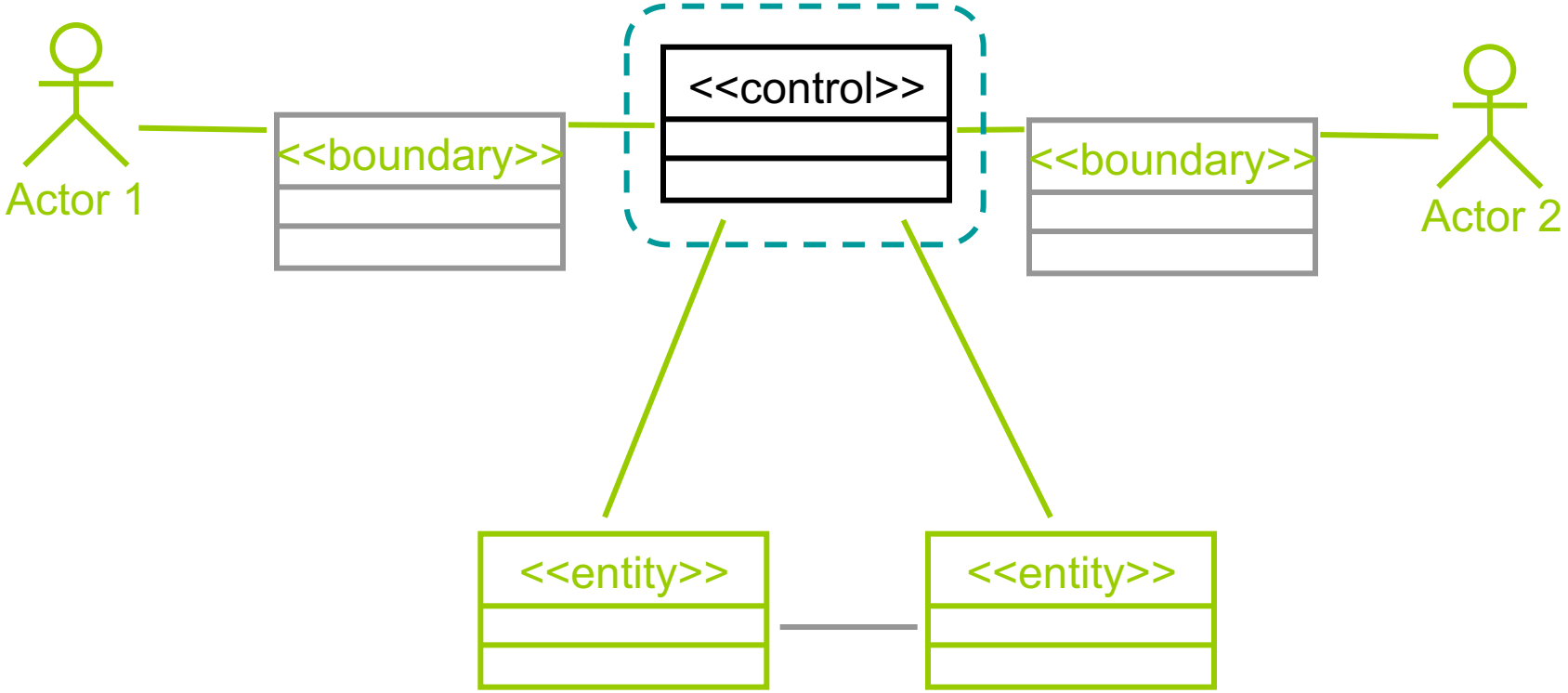
What Is a Control Class?

- Use-case behavior coordinator
  - More complex use cases generally require one or more control cases



Use-case dependent. Environment independent.

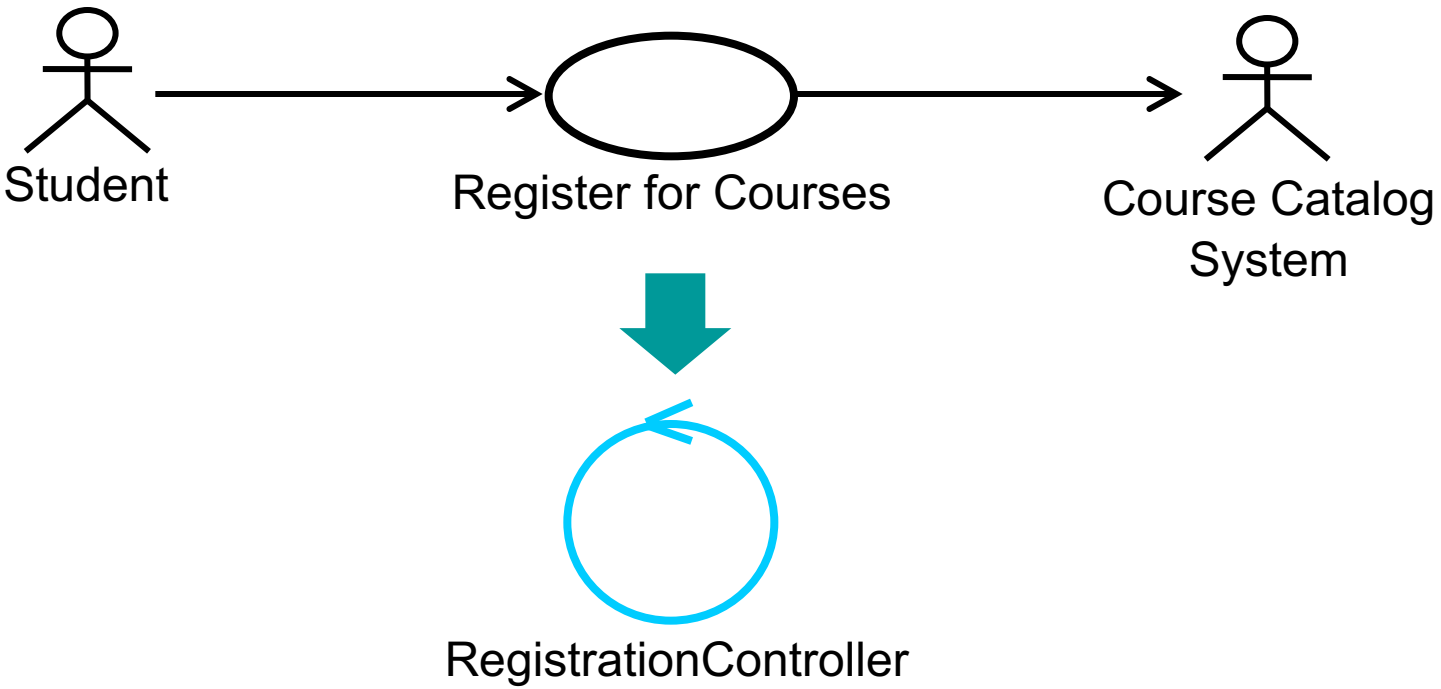
The Role of a Control Class



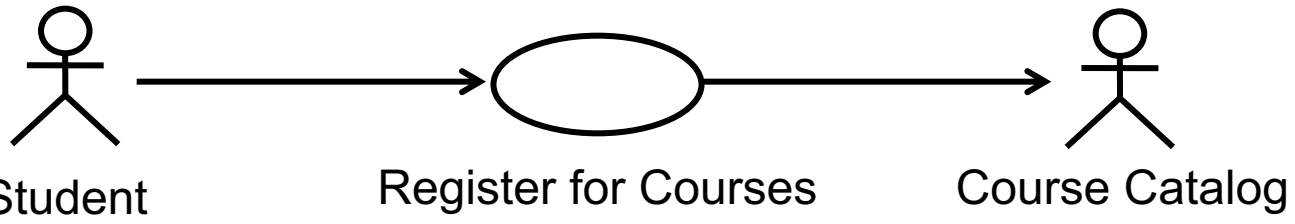
Coordinate the use-case behavior.

Example: Finding Control Classes

- In general, identify one control class per use case.
  - As analysis continues, a complex use case's control class may evolve into more than one class

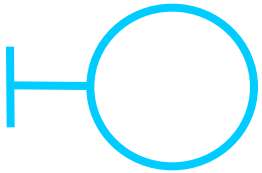


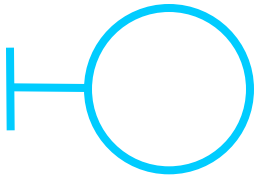
Example: Summary: Analysis Classes

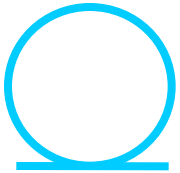


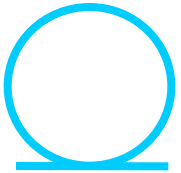
Use-Case Model

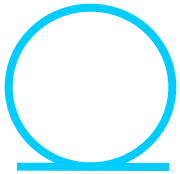
Design Model

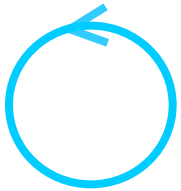

RegisterForCoursesForm


CourseCatalogSystem


Student


Schedule


CourseOffering

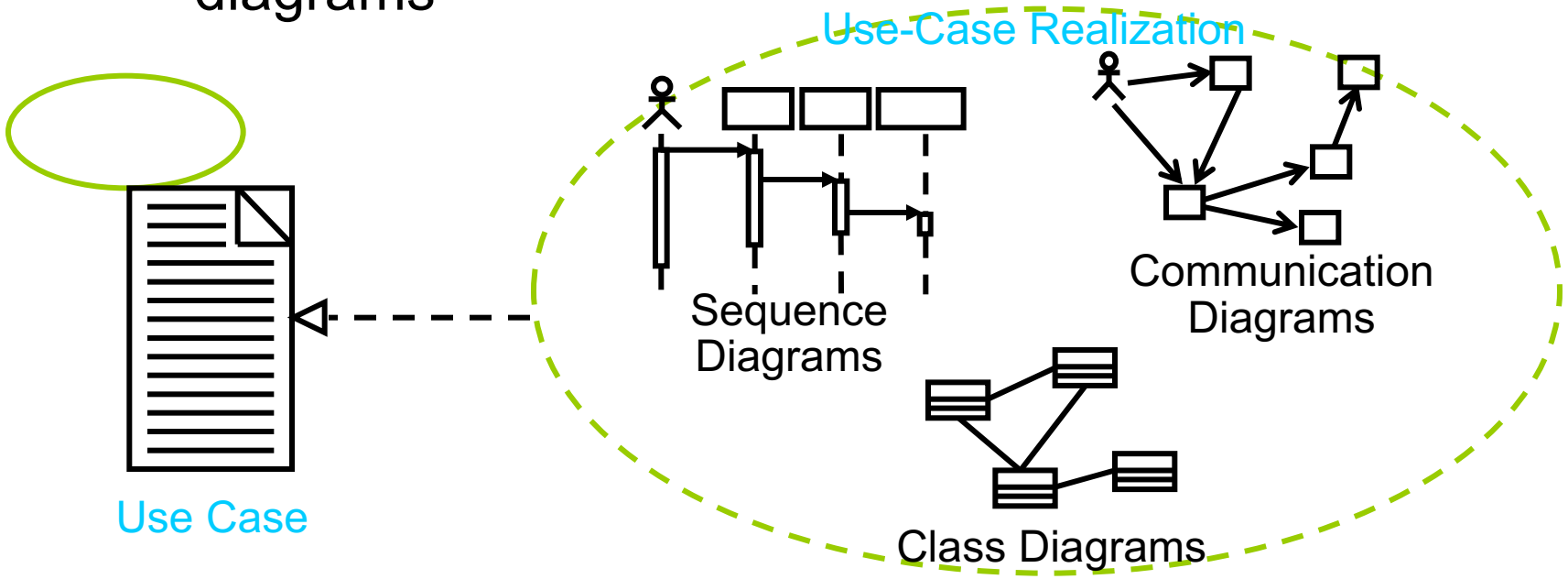

RegistrationController

Use-Case Analysis Steps

- Supplement the Use-Case Descriptions
- For each Use-Case Realization
  - Find Classes from Use-Case Behavior
  - Distribute Use-Case Behavior to Classes
- For each resulting analysis class
  - Describe Responsibilities
  - Describe Attributes and Associations
  - Qualify Analysis Mechanisms
- Unify Analysis Classes
- Checkpoints

Distribute Use-Case Behavior to Classes

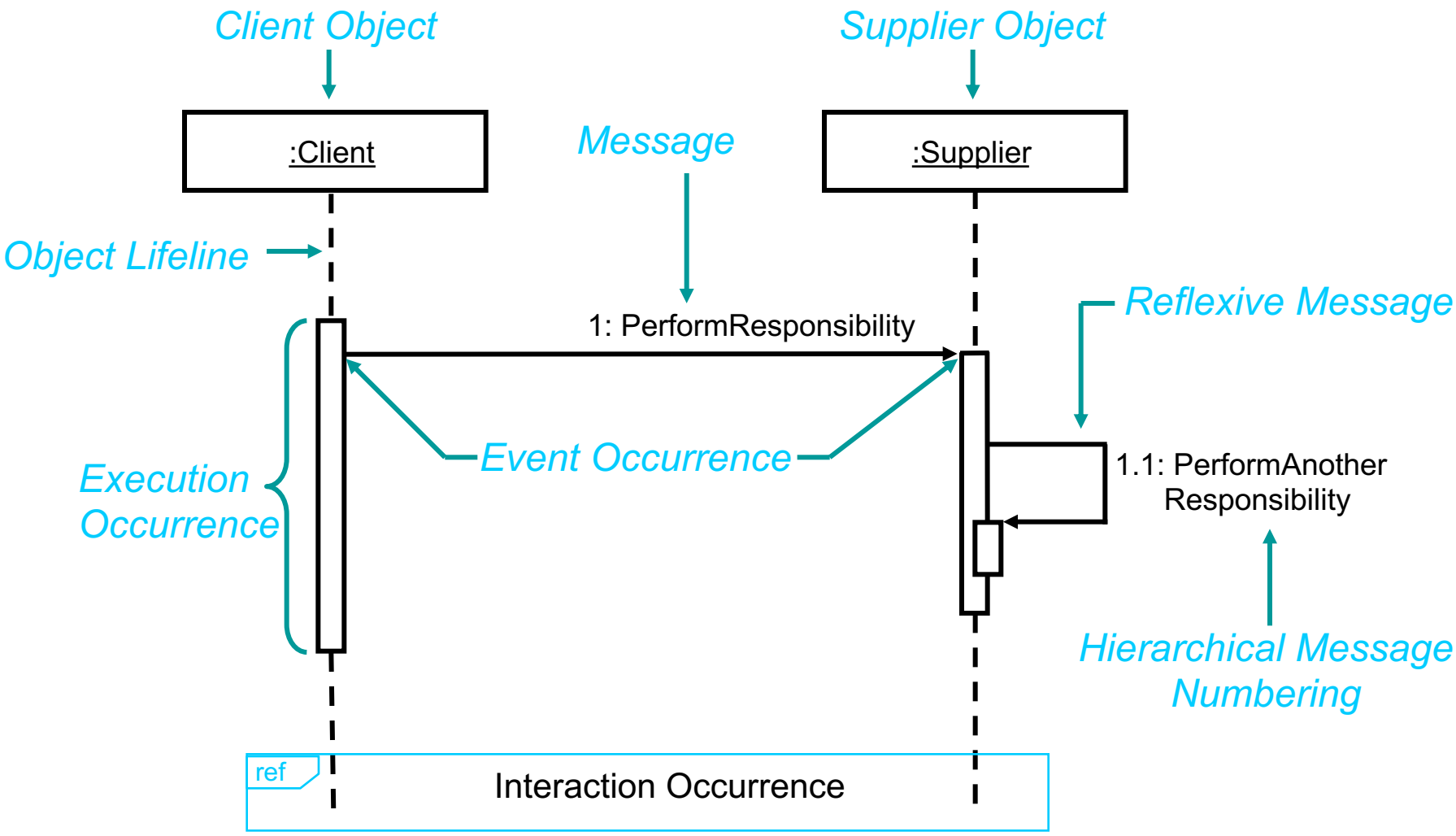
- For each use-case flow of events:
  - Identify analysis classes
  - Allocate use-case responsibilities to analysis classes
  - Model analysis class interactions in Interaction diagrams



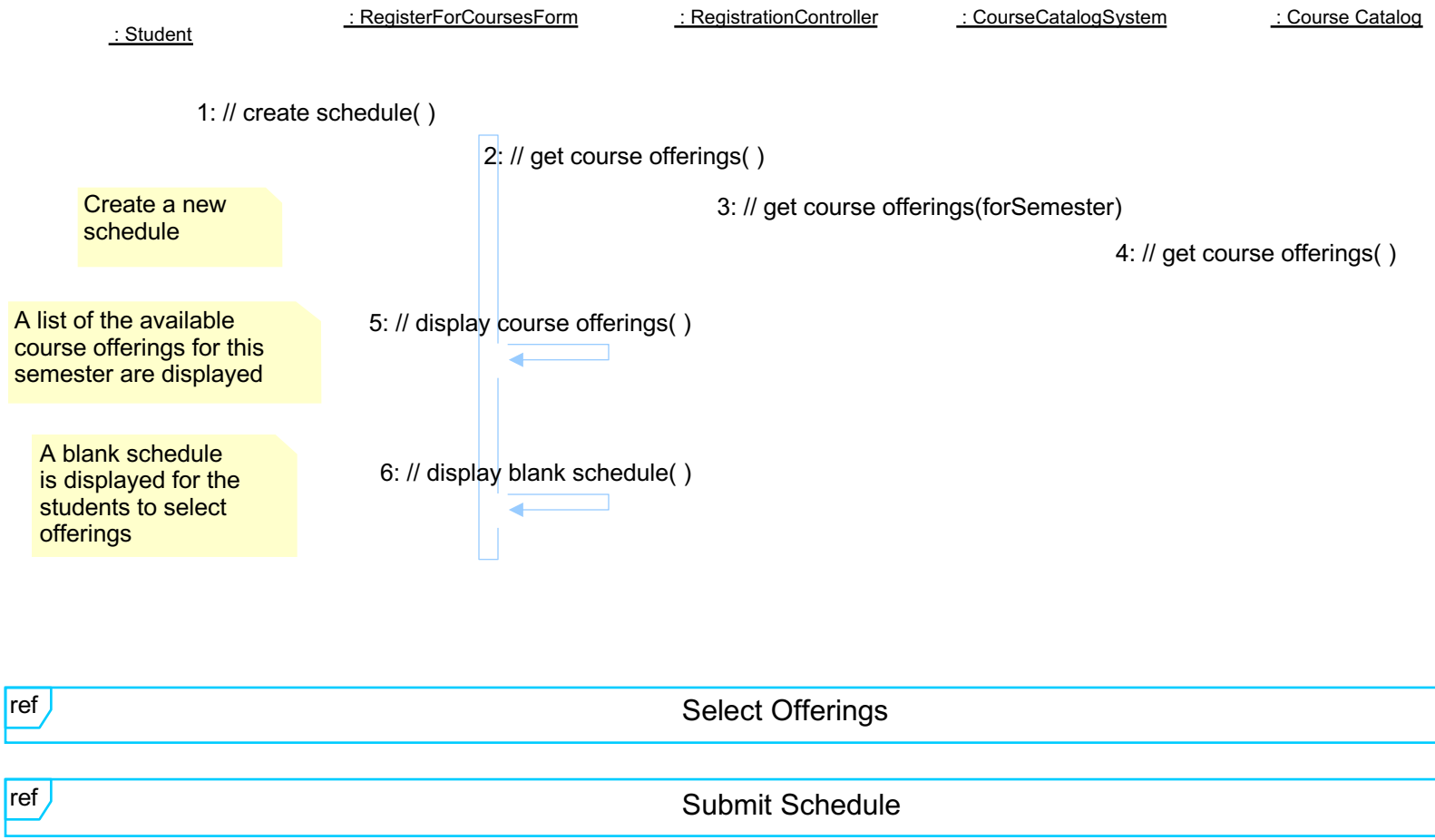
Guidelines: Allocating Responsibilities to Classes

- Use analysis class stereotypes as a guide
  - Boundary Classes
    - Behavior that involves communication with an actor
  - Entity Classes
    - Behavior that involves the data encapsulated within the abstraction
  - Control Classes
    - Behavior specific to a use case or part of a very important flow of events

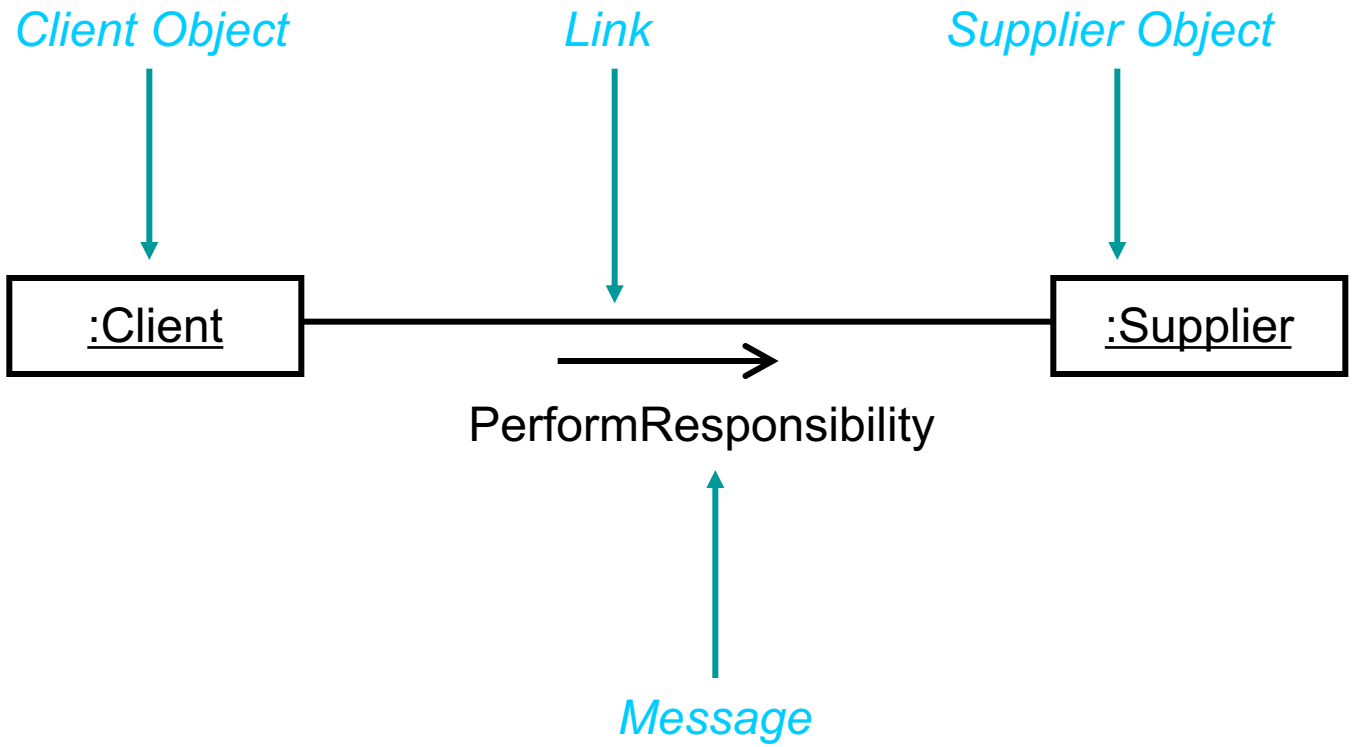
The Anatomy of Sequence Diagrams



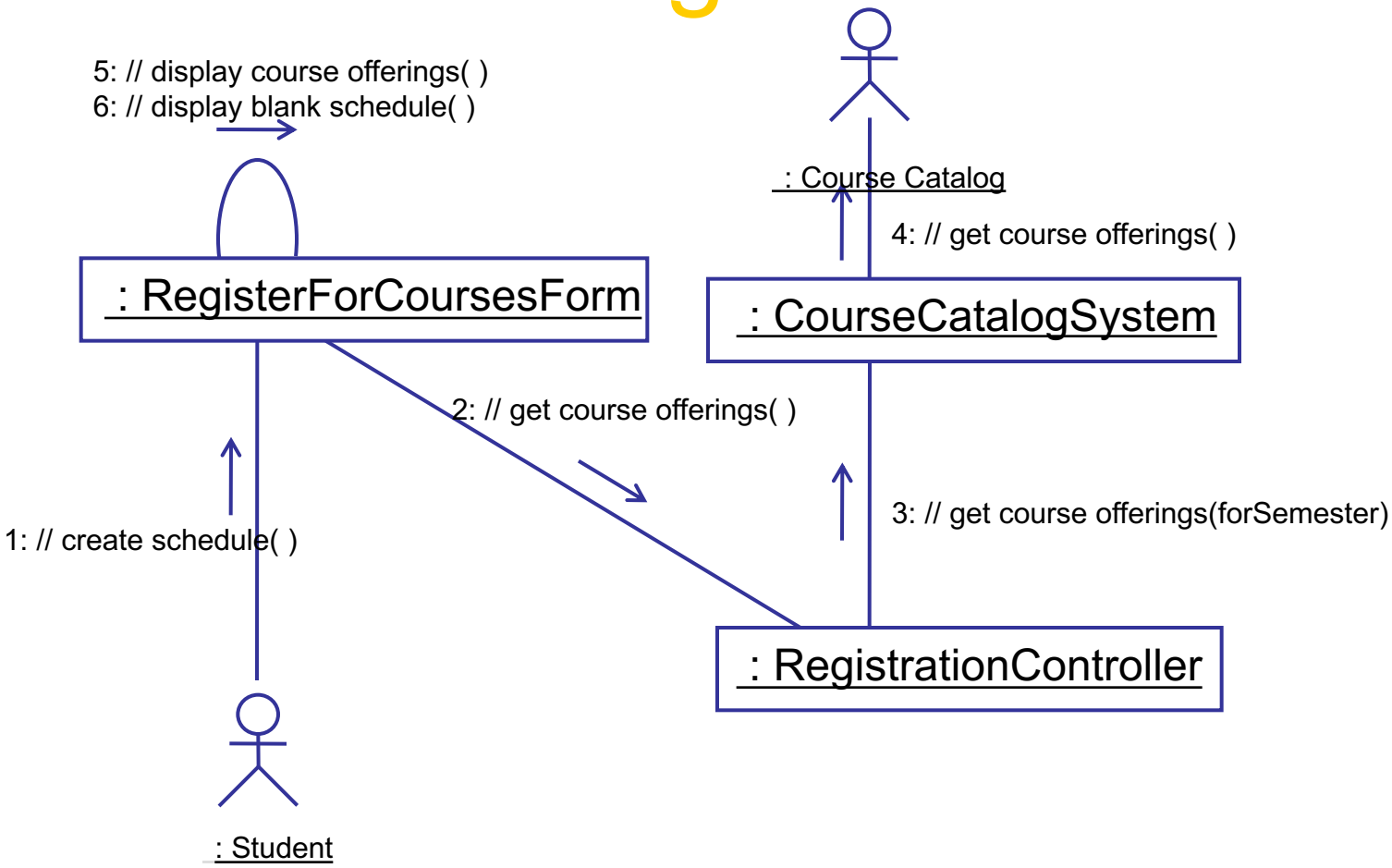
Example: Sequence Diagram



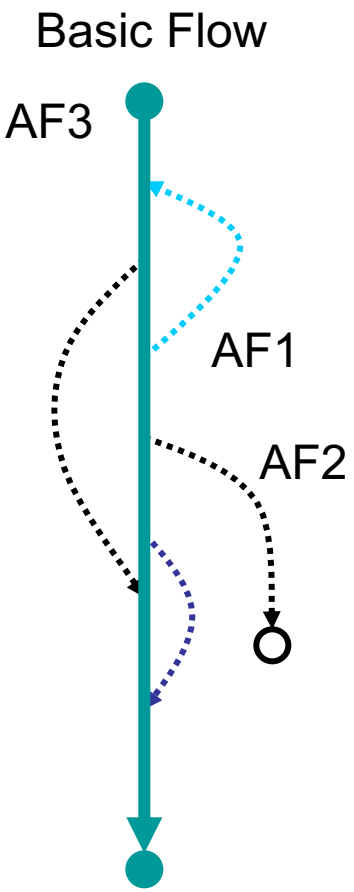
The Anatomy of Communication Diagrams



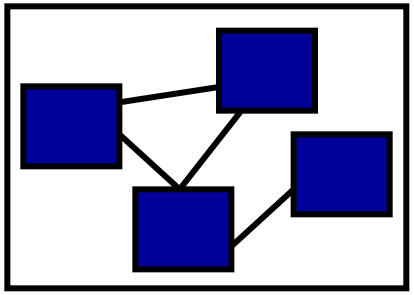
Example: Communication Diagram



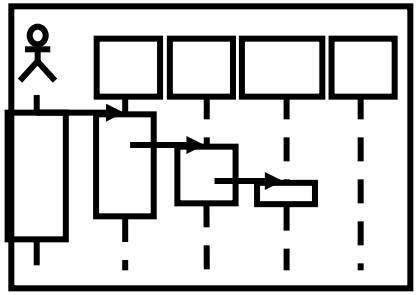
One Interaction Diagram Is Not Good Enough



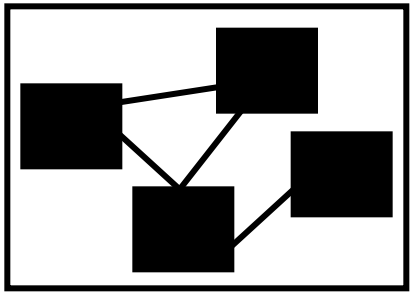
Alternate Flow 1



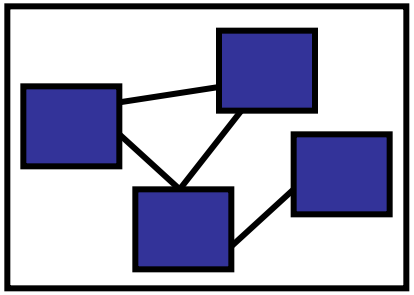
Alternate Flow 2



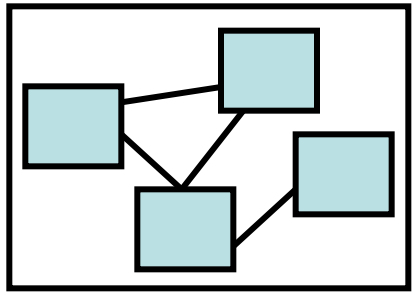
Alternate Flow 3



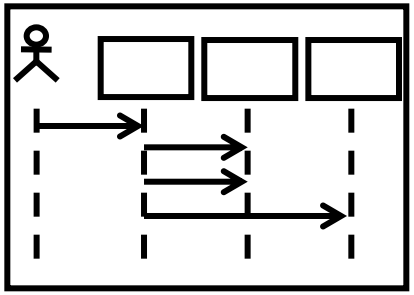
Alternate Flow 4



Alternate Flow 5



Alternate Flow n



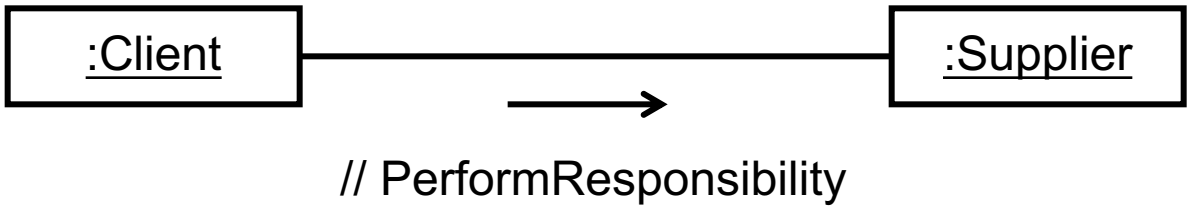
Use-Case Analysis Steps

- Supplement the Use-Case Descriptions
- For each Use-Case Realization
  - Find Classes from Use-Case Behavior
  - Distribute Use-Case Behavior to Classes
- For each resulting analysis class
  - Describe Responsibilities
  - Describe Attributes and Associations
  - Qualify Analysis Mechanisms
- Unify Analysis Classes
- Checkpoints

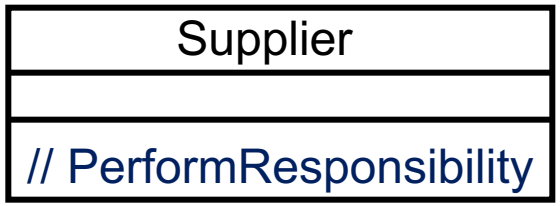
Describe Responsibilities

- What are responsibilities?
- How do I find them?

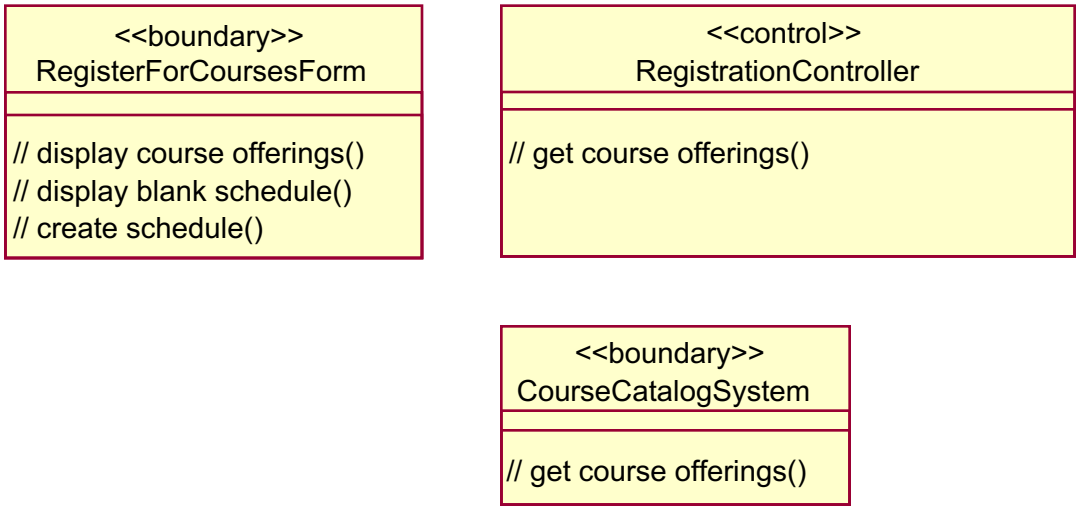
Interaction Diagram



Class Diagram



Example: View of Participating Classes (VOPC) Class Diagram



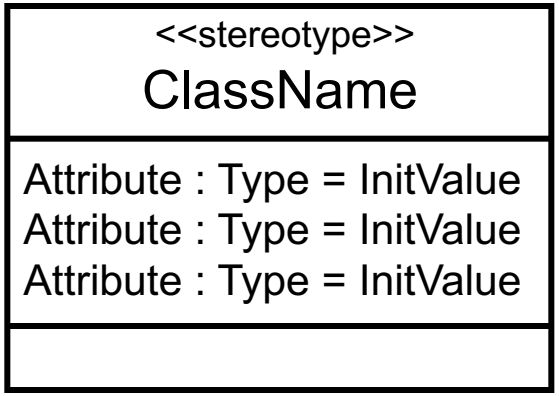
Maintaining Consistency: What to Look For

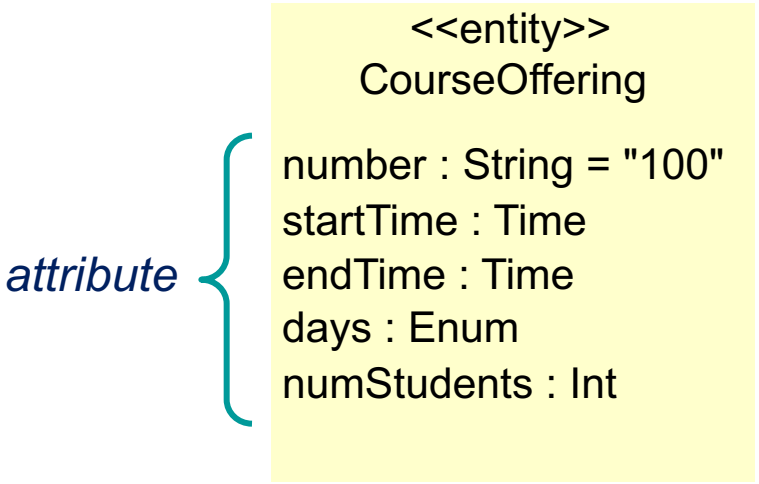
- In order of criticality
  - Redundant responsibilities across classes
  - Disjoint responsibilities within classes
  - Class with one responsibility
  - Class with no responsibilities
  - Better distribution of behavior
  - Class that interacts with many other classes

Use-Case Analysis Steps

- Supplement the Use-Case Descriptions
- For each Use-Case Realization
  - Find Classes from Use-Case Behavior
  - Distribute Use-Case Behavior to Classes
- For each resulting analysis class
  - Describe Responsibilities
  - Describe Attributes and Associations
  - Qualify Analysis Mechanisms
- Unify Analysis Classes
- Checkpoints

Review: What Is an Attribute?





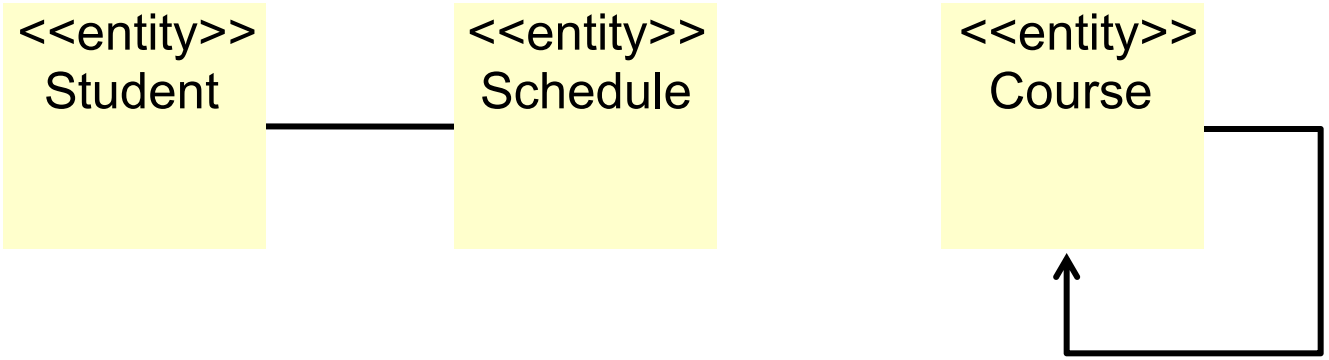
In analysis, do not spend time on attribute signatures.

Finding Attributes

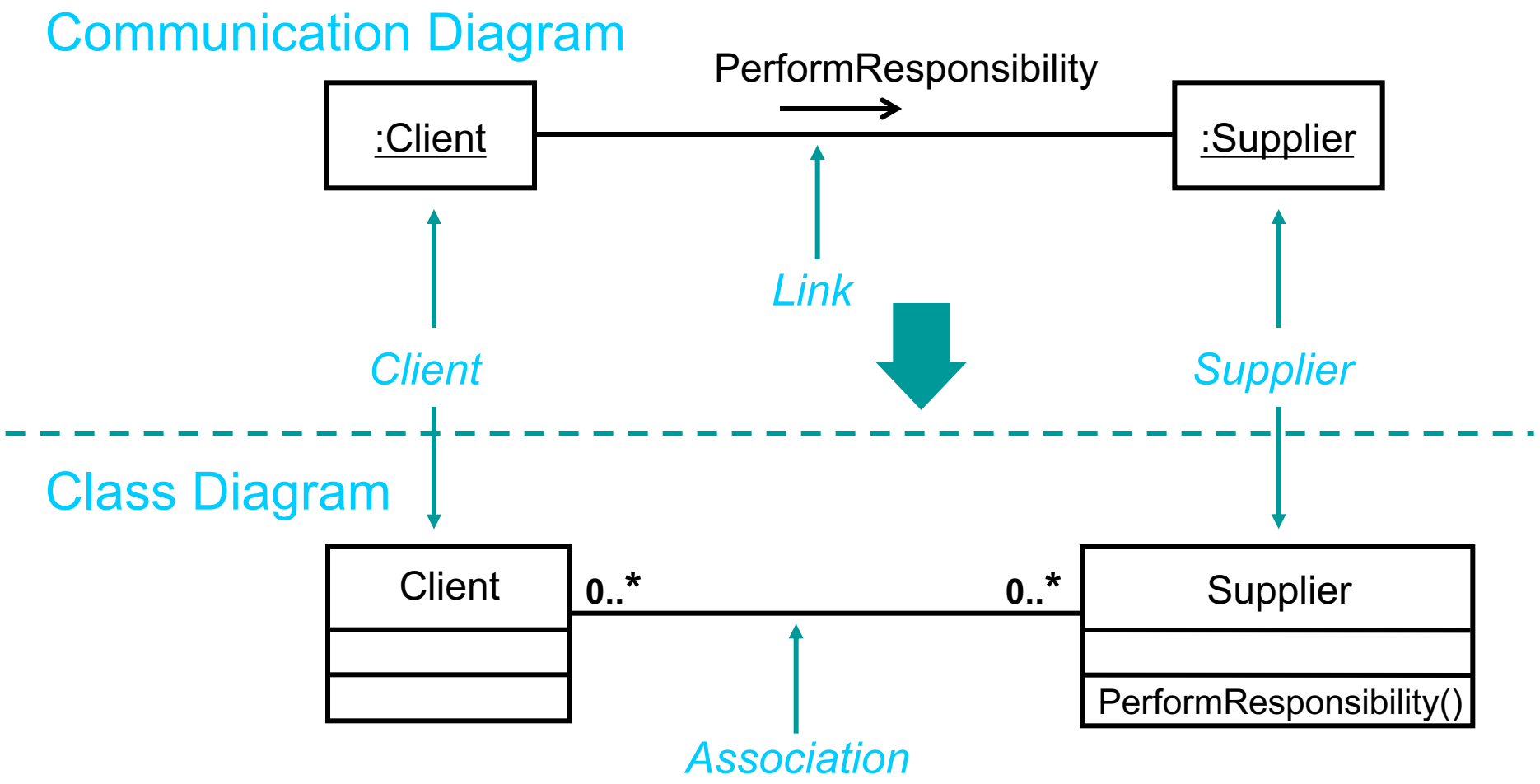
- Properties/characteristics of identified classes
- Information retained by identified classes
- “Nouns” that did not become classes
  - Information whose value is the important thing
  - Information that is uniquely "owned” by an object
  - Information that has no behavior

Review: What Is an Association?

- The semantic relationship between two or more classifiers that specifies connections among their instances
  - A structural relationship, specifying that objects of one thing are connected to objects of another



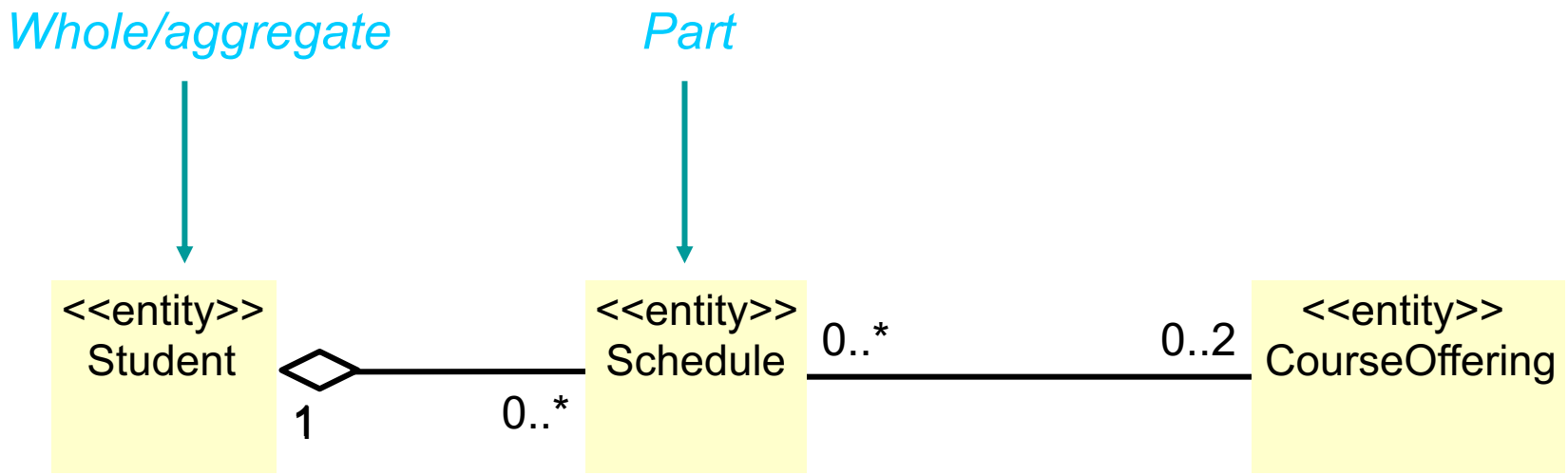
Finding Relationships



Relationship for every link!

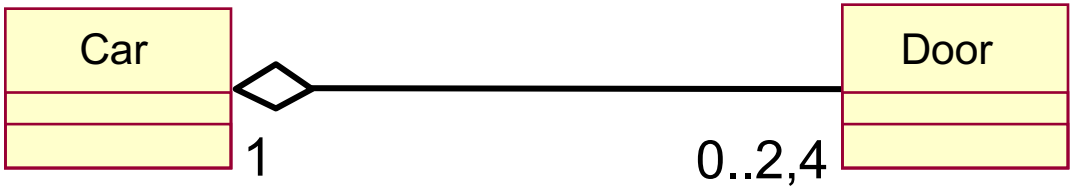
Review: What Is Aggregation?

- A special form of association that models a whole-part relationship between an aggregate (the whole) and its parts

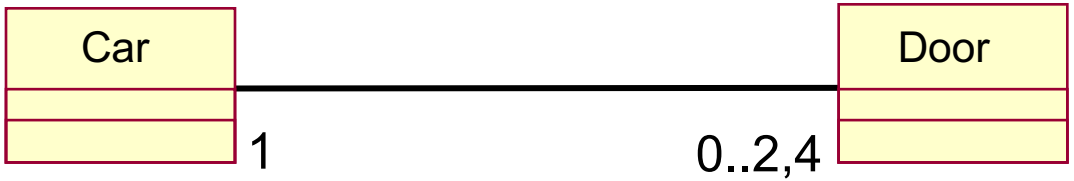


Association or Aggregation?

- If two objects are tightly bound by a whole-part relationship
  - The relationship is an aggregation.



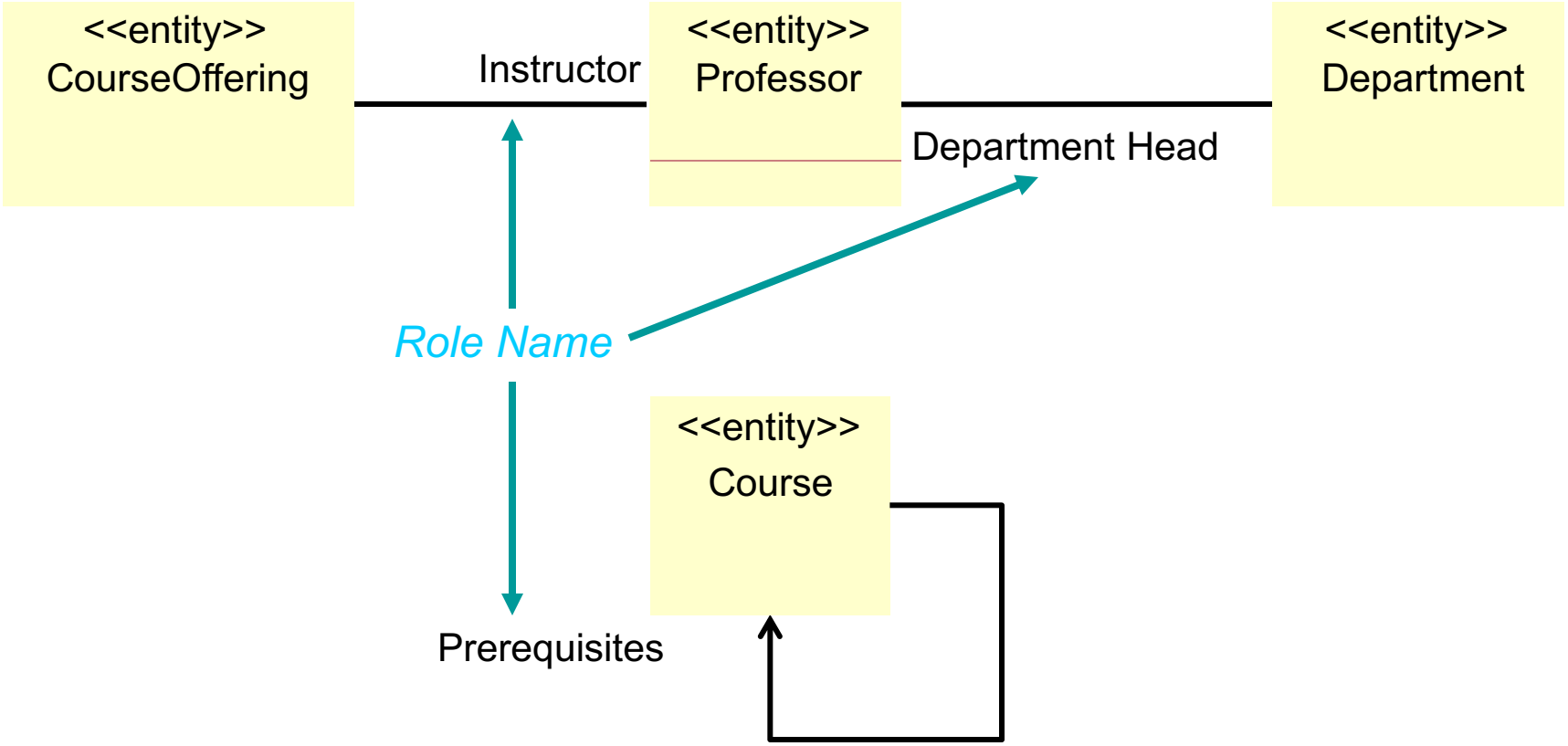
- If two objects are usually considered as independent, although they are often linked
  - The relationship is an association.



When in doubt, use association.

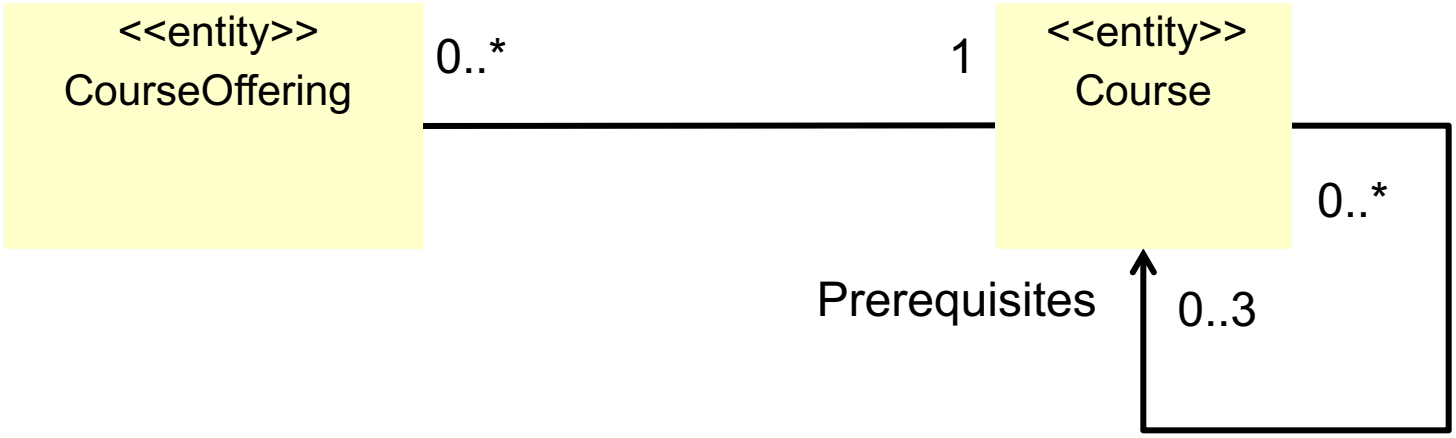
What Are Roles?

- The “face” that a class plays in the association

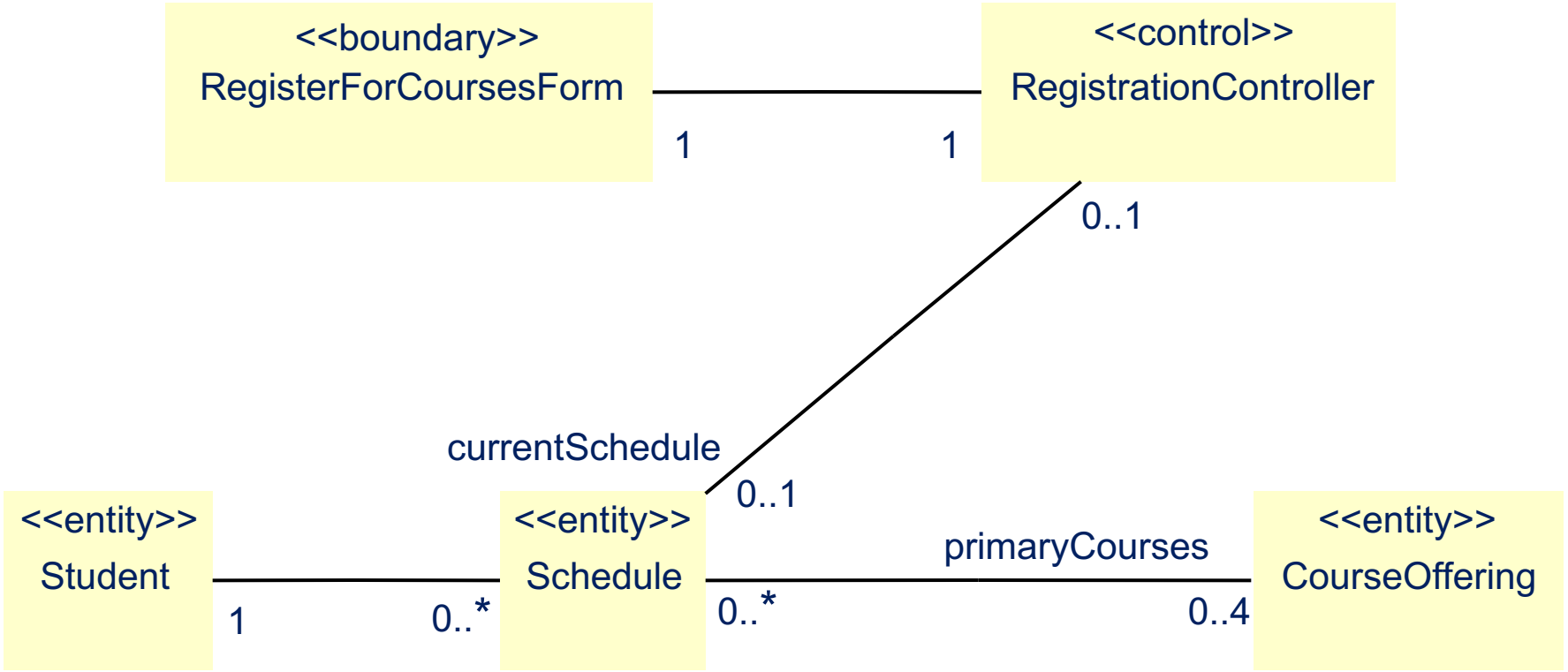


What Does Multiplicity Mean?

- Multiplicity answers two questions:
  - Is the association mandatory or optional?
  - What is the minimum and maximum number of instances that can be linked to one instance?



Example: VOPC: Finding Relationships

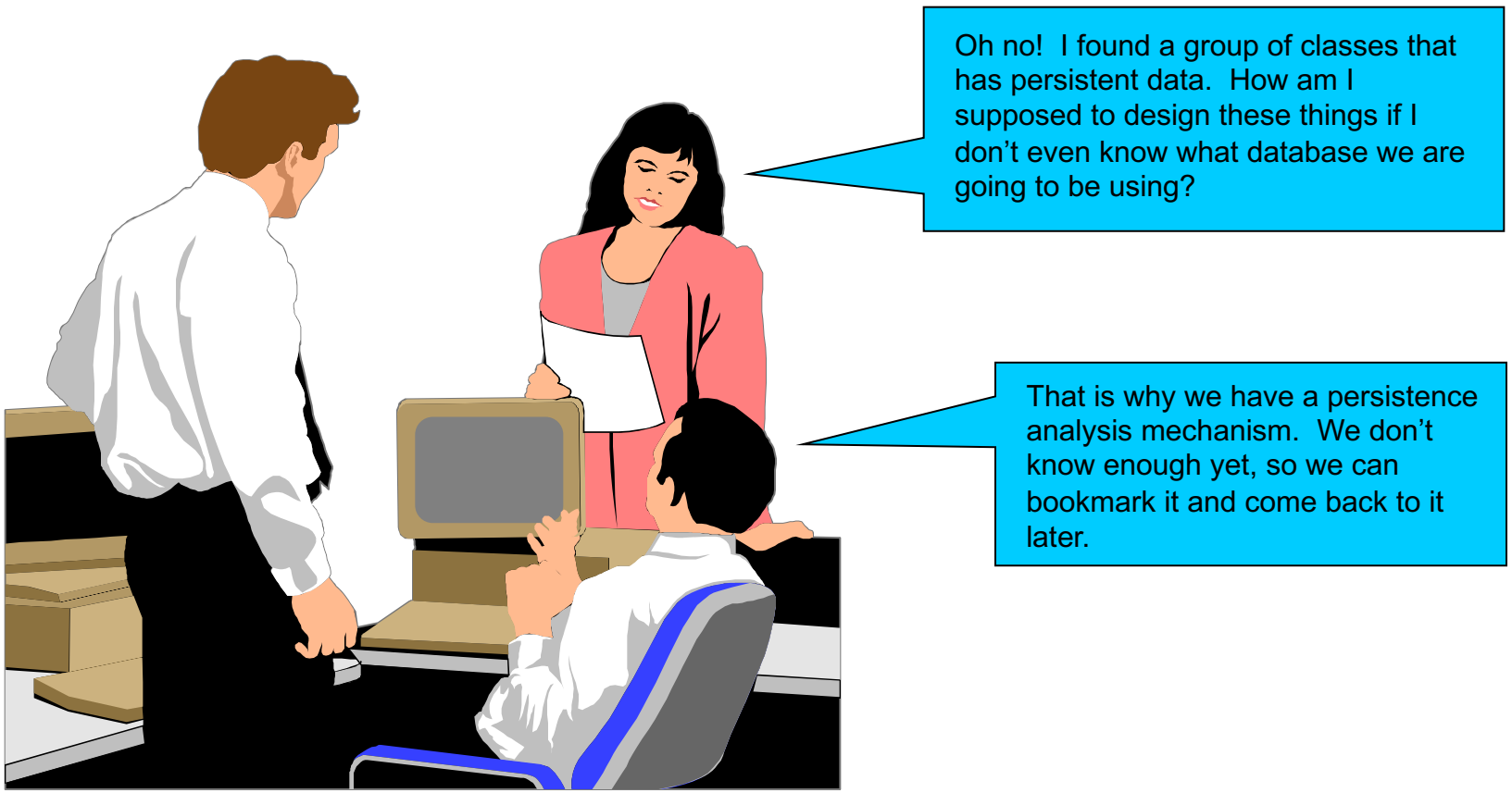


Use-Case Analysis Steps

- Supplement the Use-Case Descriptions
- For each Use-Case Realization
  - Find Classes from Use-Case Behavior
  - Distribute Use-Case Behavior to Classes
- For each resulting analysis class
  - Describe Responsibilities
  - Describe Attributes and Associations
  - Qualify Analysis Mechanisms
- Unify Analysis Classes
- Checkpoints

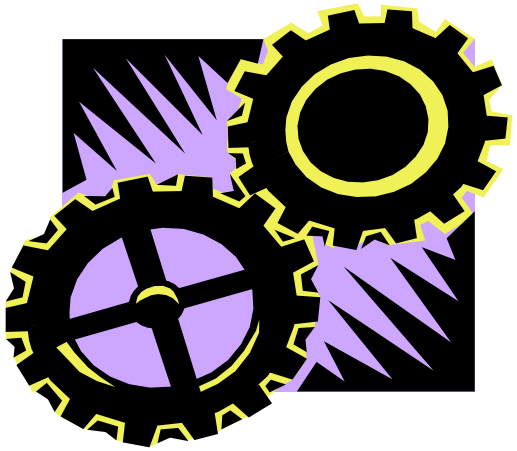
Review: Why Use Analysis Mechanisms?

Analysis mechanisms are used during analysis to reduce the complexity of analysis and to improve its consistency by providing designers with a shorthand representation for complex behavior.



Describing Analysis Mechanisms

- Collect all analysis mechanisms in a list
- Draw a map of the client classes to the analysis mechanisms
- Identify characteristics of the analysis mechanisms



Example: Describing Analysis Mechanisms

- Analysis class to analysis mechanism map

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

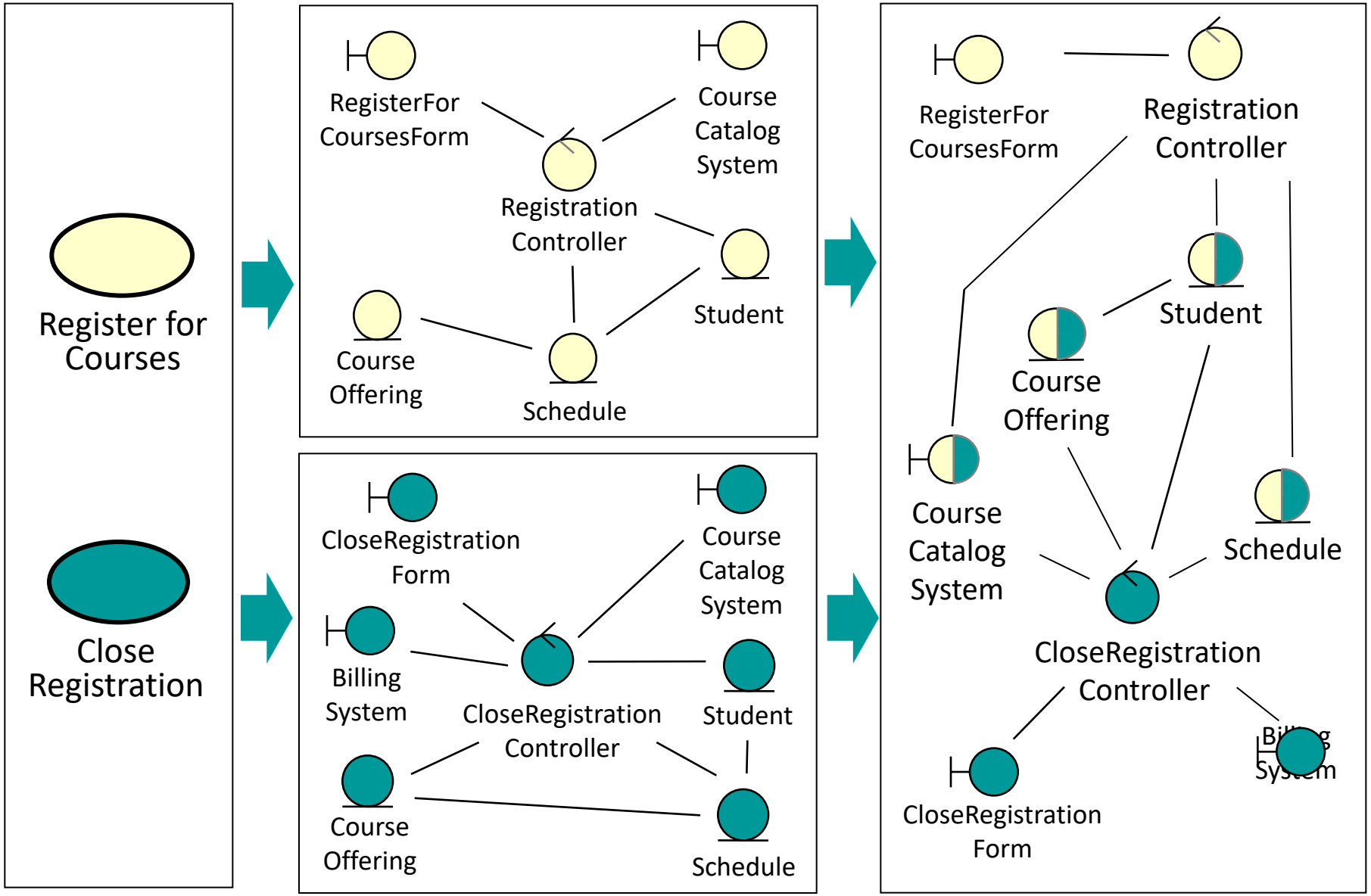
Example: Describing Analysis Mechanisms (continued)

- Analysis mechanism characteristics
- Persistency for Schedule class:
  - Granularity: 1 to 10 Kbytes per product
  - Volume: up to 2,000 schedules
  - Access frequency
    - Create: 500 per day
    - Read: 2,000 access per hour
    - Update: 1,000 per day
    - Delete: 50 per day
  - Other characteristics

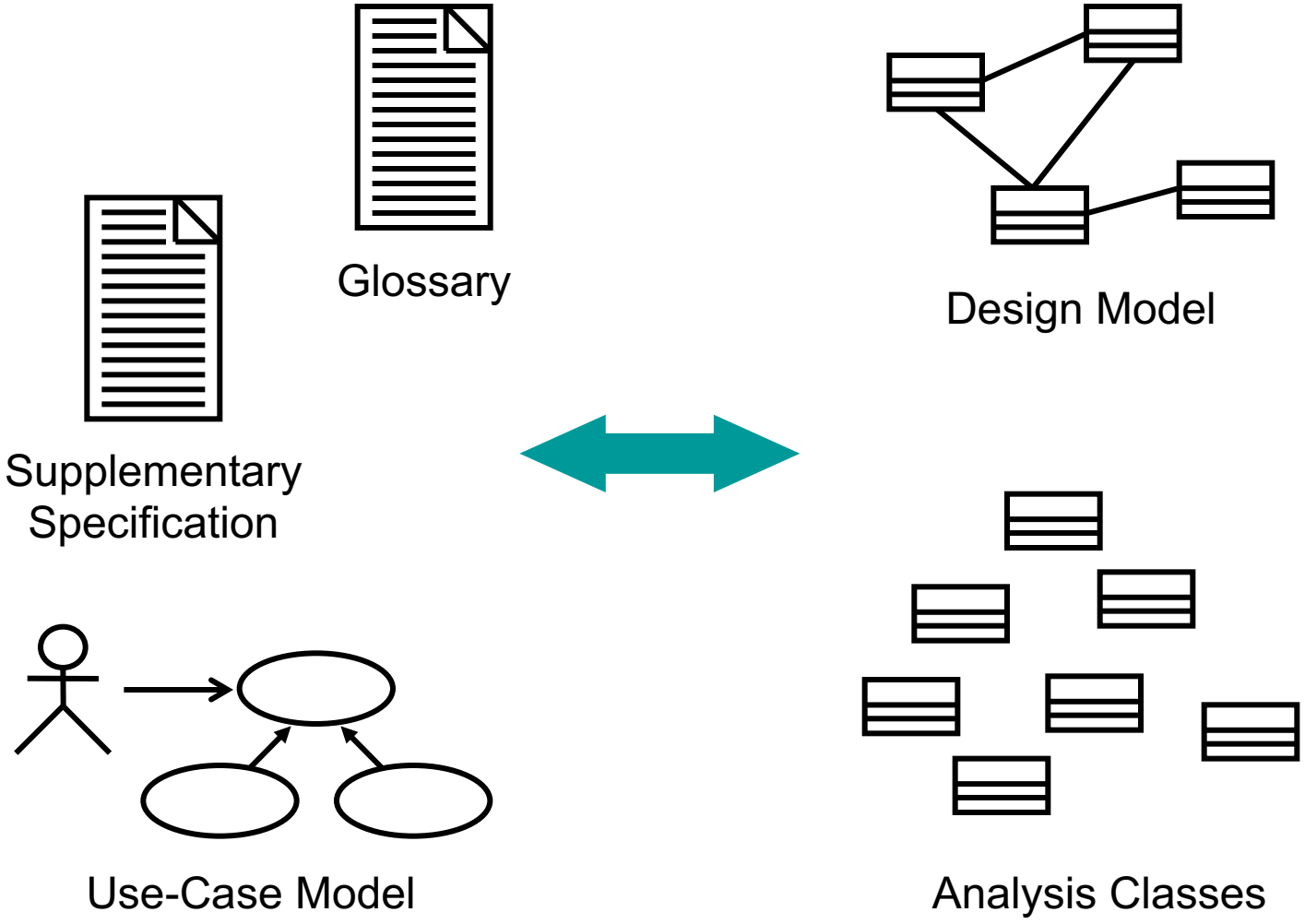
Use-Case Analysis Steps

- Supplement the Use-Case Descriptions
- For each Use-Case Realization
  - Find Classes from Use-Case Behavior
  - Distribute Use-Case Behavior to Classes
- For each resulting analysis class
  - Describe Responsibilities
  - Describe Attributes and Associations
  - Qualify Analysis Mechanisms
- Unify Analysis Classes
- Checkpoints

Unify Analysis Classes



Evaluate Your Results



Use-Case Analysis Steps

- Supplement the Use-Case Descriptions
- For each Use-Case Realization
  - Find Classes from Use-Case Behavior
  - Distribute Use-Case Behavior to Classes
- For each resulting analysis class
  - Describe Responsibilities
  - Describe Attributes and Associations
  - Qualify Analysis Mechanisms
- Unify Analysis Classes
- Checkpoints

Checkpoints: Analysis Classes

- Are the classes reasonable?
- Does the name of each class clearly reflect the role it plays?
- Does the class represent a single well-defined abstraction?
- Are all attributes and responsibilities functionally coupled?
- Does the class offer the required behavior?
- Are all specific requirements on the class addressed?

Checkpoints: Use-Case Realizations

- Have all the main and/or sub-flows been handled, including exceptional cases?
- Have all the required objects been found?
- Has all behavior been unambiguously distributed to the participating objects?
- Has behavior been distributed to the right objects?
- Where there are several Interaction diagrams, are their relationships clear and consistent?

Review: Use-Case Analysis

- What is the purpose of Use-Case Analysis?
- What is a Use-Case Realization?
- What is an analysis class? Name and describe the three analysis stereotypes.
- Describe some considerations when allocating responsibilities to analysis classes.
- What two questions does multiplicity answer?