CS3300 Introduction to Software Engineering

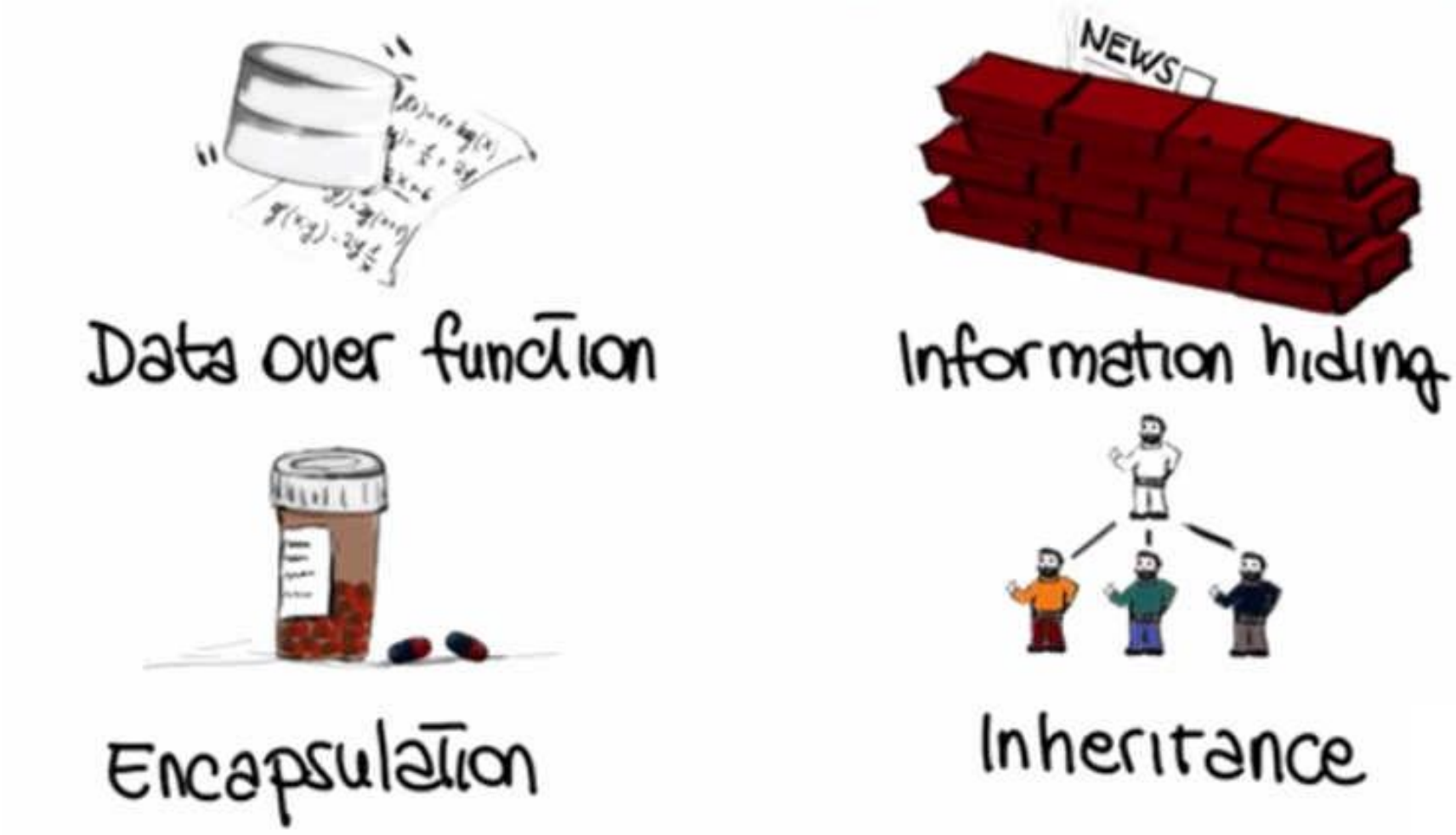
Lecture 12: Object Orientation and Unified Modeling Language

Nimisha Roy ▸ nroy9@gatech.edu

Slides adapted from Alessandro Orso

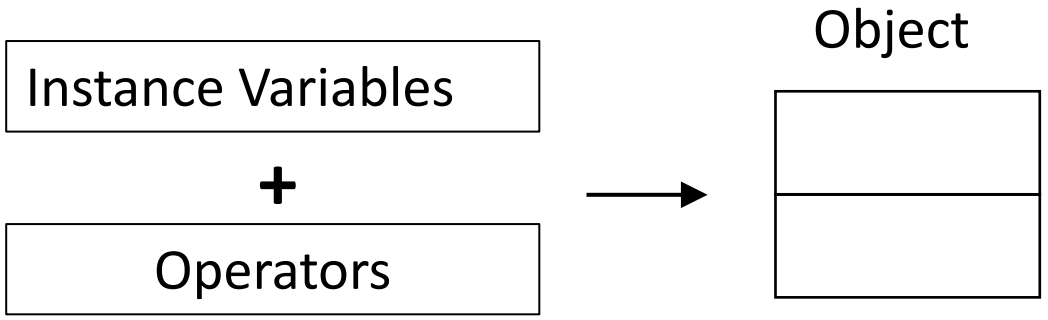
What is Object Orientation

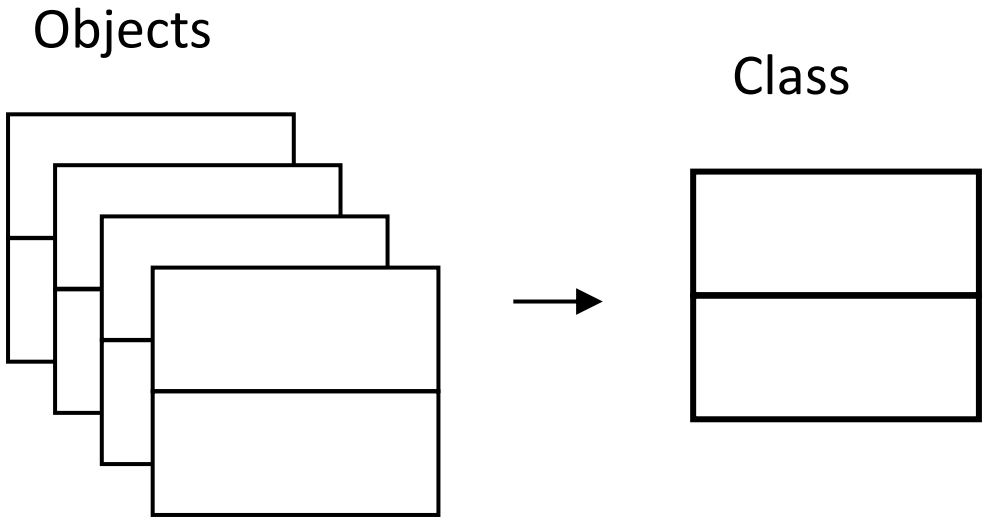
Principles of Object Orientation



Slide adapted from Alessandro Orso

Objects and Classes



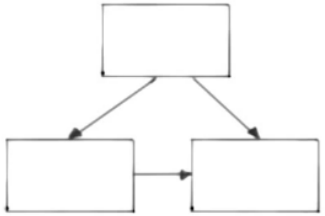


Slide adapted from Alessandro Orso

Why use OO?

- Reduce maintenance Costs – by limiting effects of changes (Encapsulation & Information Hiding)

- Improve development Process – Favoring Code & Design Reuse



- Enforce Good design Principles – encapsulation, information hiding, high cohesion, low coupling

Slide adapted from Alessandro Orso

Were you paying attention?

What are the benefits of an OO approach in software development process?

[✓] Increased reuse because of the modular coding style

[✓] Increased maintainability because the system design can accommodate changes more easily

[ ] Increase speed because OO systems tend to run faster

[✓] Increased understandability because the design models real-world entities

OO Analysis & Design

Functional Oriented => Data Oriented

Real World Objects => Requirements

Obtain/Prepare textual description of the problem

Underline nouns => Classes

Underline adjectives => Attributes

Underline active verbs => Operations

Slide adapted from Alessandro Orso

Were you paying attention?

Consider the following requirement for an online shopping website: “Users can add more than one item on sale at a time to a shopping cart”
Which of the following elements should be modeled as classes?

[✓] Item

[ ] Sale

[✓] Shopping Cart

[ ] Time

[✓] User

Running Example: Course Management System

1. The registration manager sets up the curriculum for a semester using a scheduling algorithm
2. One course may have multiple course offerings
3. Each course offering has a number, location and time
4. Students select 4 primary courses and 2 alternative courses by submitting a registration form
5. Students may use the system to add/drop courses for a period of time after registration
6. Professors use the system to receive their course offering rosters
7. Users of the registration system are assigned passwords which are used at login validation

Slide adapted from Alessandro Orso

Unified Modeling Language

Structural Diagrams: represent static characteristics of the system that we need to model

Class Diagram

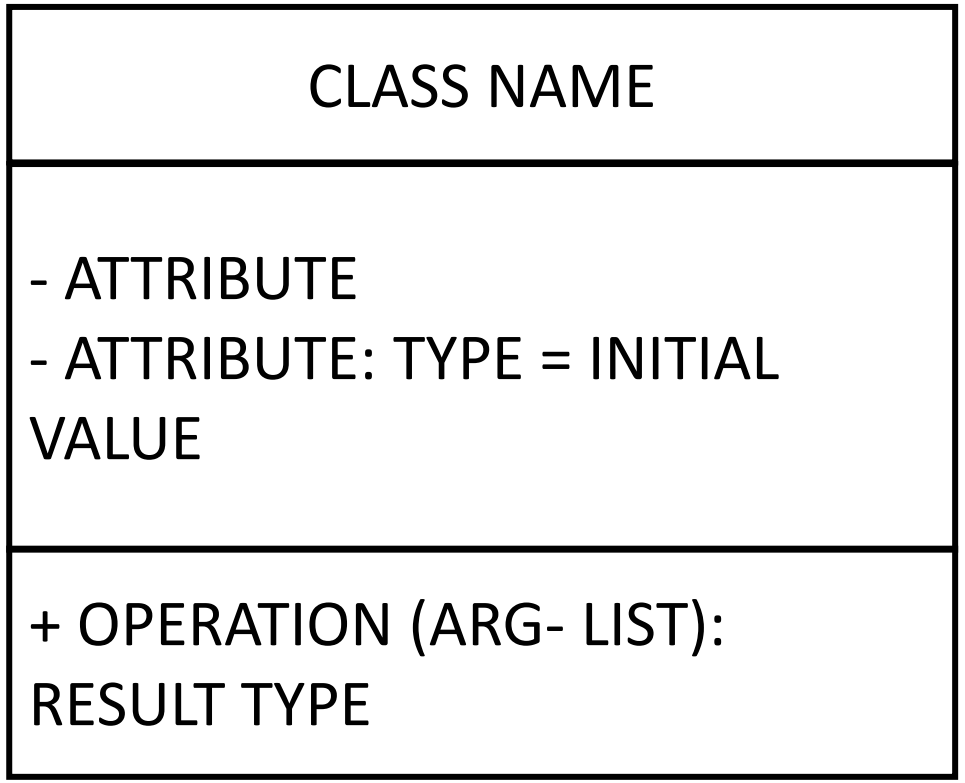
Static, Structural View of the System

Describes

- Classes and their Structure
- Relationships among classes

Slide adapted from Alessandro Orso

Class Diagram: Class



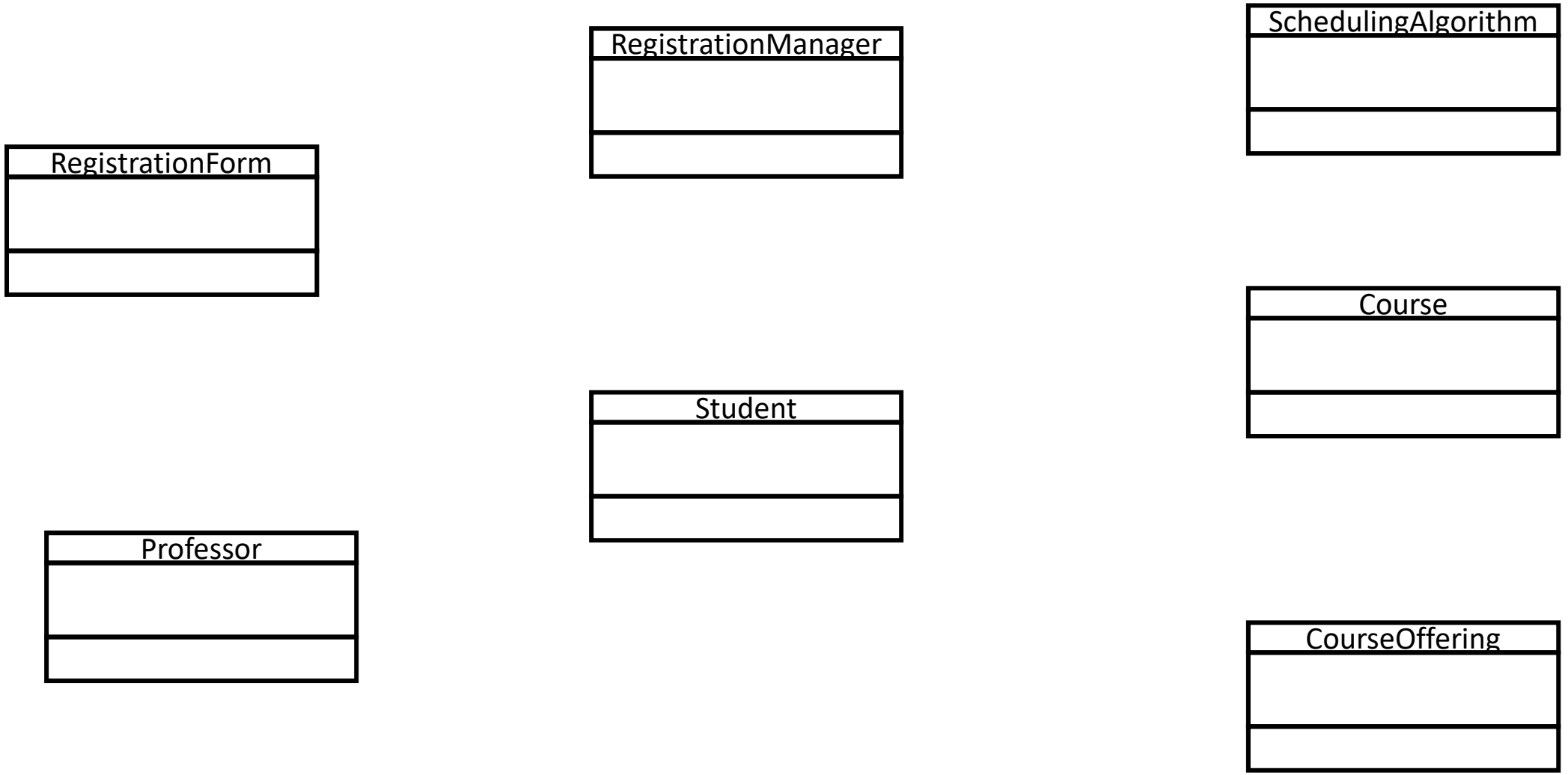
Slide adapted from Alessandro Orso

Class Diagram for our example: Step 1 identify Classes as Nouns

1. The registration manager sets up the curriculum for a semester using a scheduling algorithm
2. One course may have multiple course offerings
3. Each course offering has a number, location and time
4. Students select 4 primary courses and 2 alternative courses by submitting a registration form
5. Students may use the system to add/drop courses for a period of time after registration
6. Professors use the system to receive their course offering rosters
7. Users of the registration system are assigned passwords which are used at login validation

Slide adapted from Alessandro Orso

Class Diagram for our example



Slide adapted from Alessandro Orso

Class Diagram: Attributes

Represent the structure of a class

May be found by

- By examining class definitions
- By studying the requirements
- By applying domain knowledge.

Slide adapted from Alessandro Orso

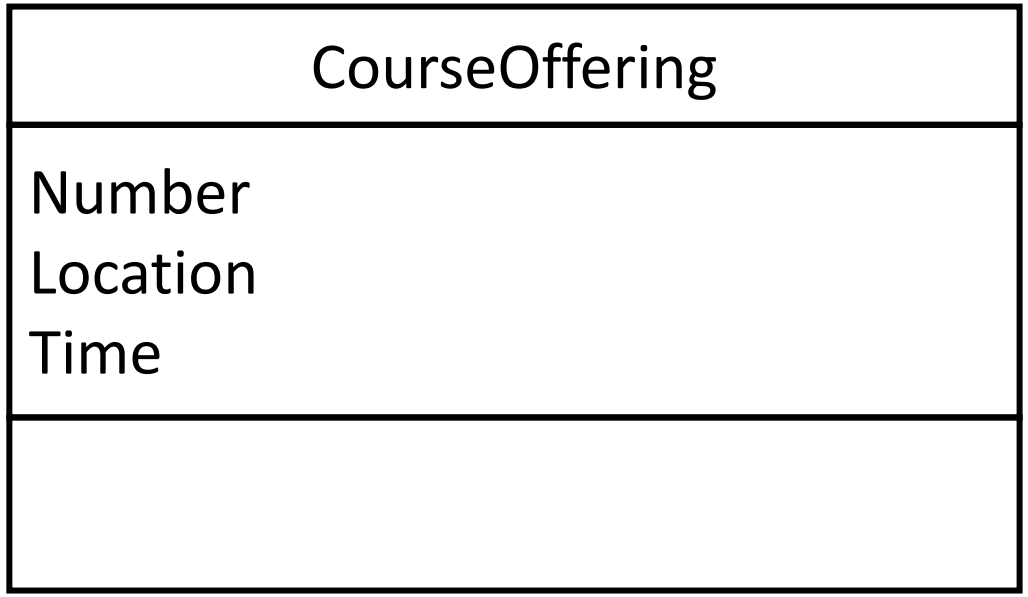
Class Diagram for our example: Step 2 identify attributes as adjectives

1. The registration manager sets up the curriculum for a semester using a scheduling algorithm
2. One course may have multiple course offerings
3. Each course offering has a number, location and time
4. Students select 4 primary courses and 2 alternative courses by submitting a registration form
5. Students may use the system to add/drop courses for a period of time after registration
6. Professors use the system to receive their course offering rosters
7. Users of the registration system are assigned passwords which are used at login validation

Slide adapted from Alessandro Orso

Class Diagram: Attributes

Each course offering has a number, location and time



Slide adapted from Alessandro Orso

Class Diagram: Operations

Represent the behavior of a class

May be found by examining interactions among entities

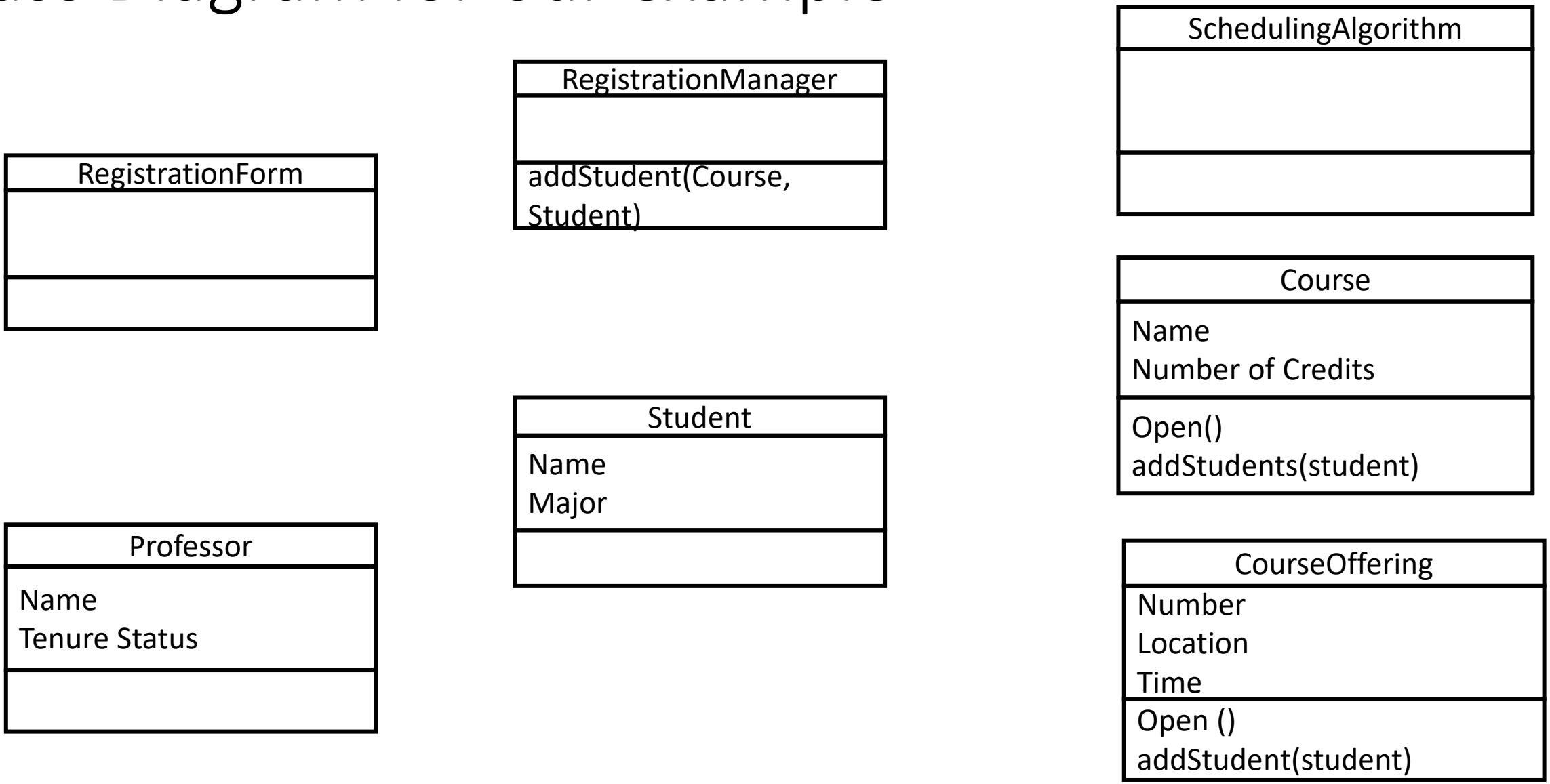
Slide adapted from Alessandro Orso

Class Diagram for our example: Step 3 identify operations as active verbs

1. The registration manager sets up the curriculum for a semester using a scheduling algorithm
2. One course may have multiple course offerings
3. Each course offering has a number, location and time
4. Students select 4 primary courses and 2 alternative courses by submitting a registration form
5. Students may use the system to add/drop courses for a period of time after registration
6. Professors use the system to receive their course offering rosters
7. Users of the registration system are assigned passwords which are used at login validation

Slide adapted from Alessandro Orso

Class Diagram for our example



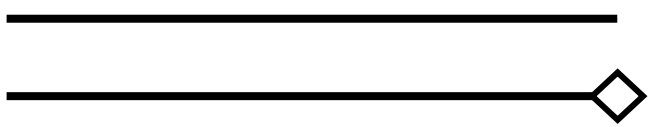
Slide adapted from Alessandro Orso

Class Diagram: Relationships

Describe interactions between objects

Dependencies: X uses Y

Associations/Aggregations: X has a Y



Generalization: X is a Y

Slide adapted from Alessandro Orso

Were you paying attention?

Which of the following relationships is an actual relationship for the system we are modelling?

[✓] RegistrationManager uses SchedulingAlgorithm (dependency)

[✓] RegistrationManager uses Student (dependency)

[ ] Student uses RegistrationManager (dependency)

[✓] Student registers for CourseOffering (association)

[ ] Student consists of CourseOffering (aggregation)

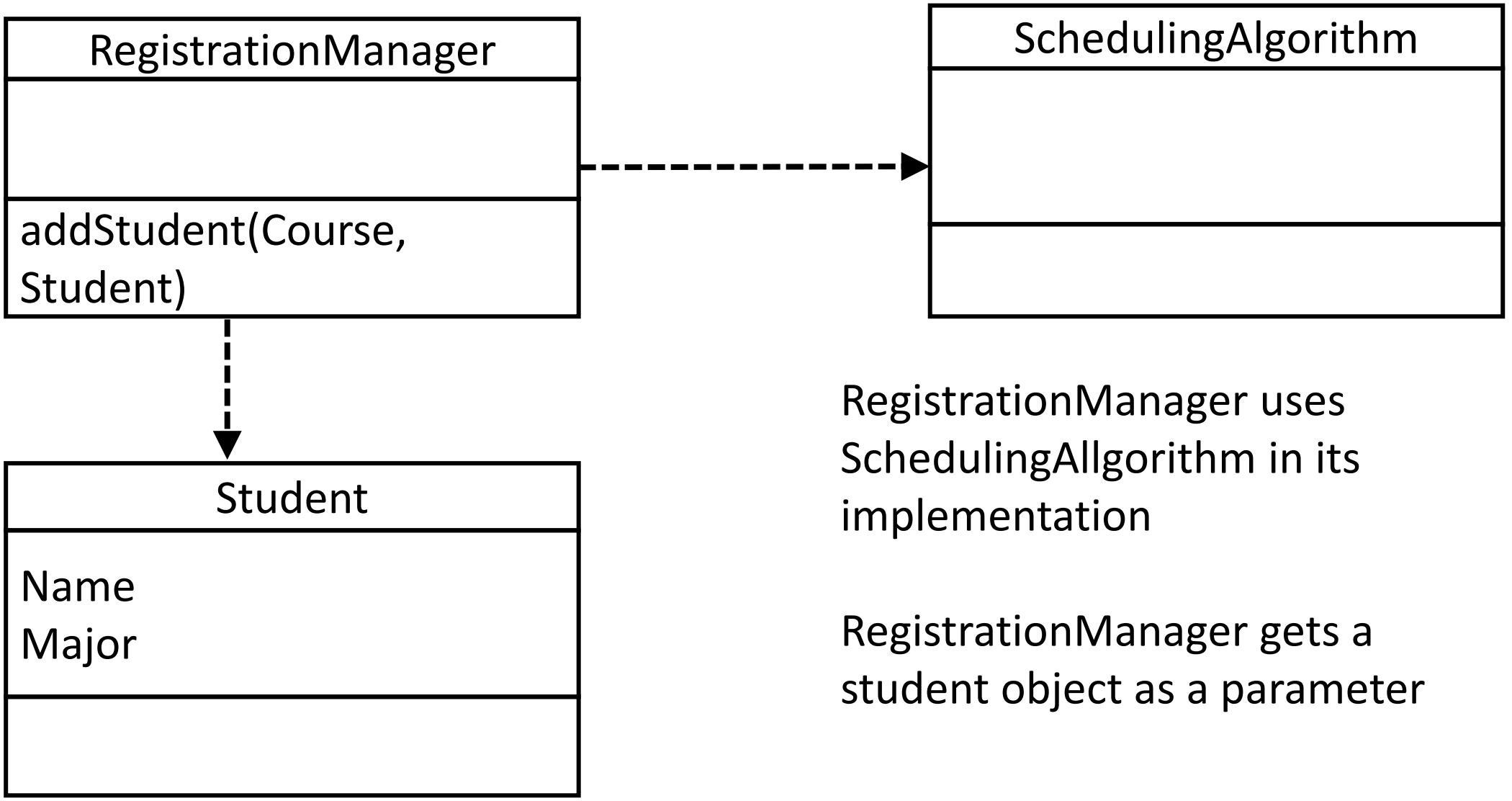
[✓] Course consists of CourseOffering (aggregation)

[ ] CourseOffering is a Course (generalization)

[✓] Student is a RegistrationUser (generalization)

[✓] Professor is a RegistrationUser (generalization)

Dependency Example

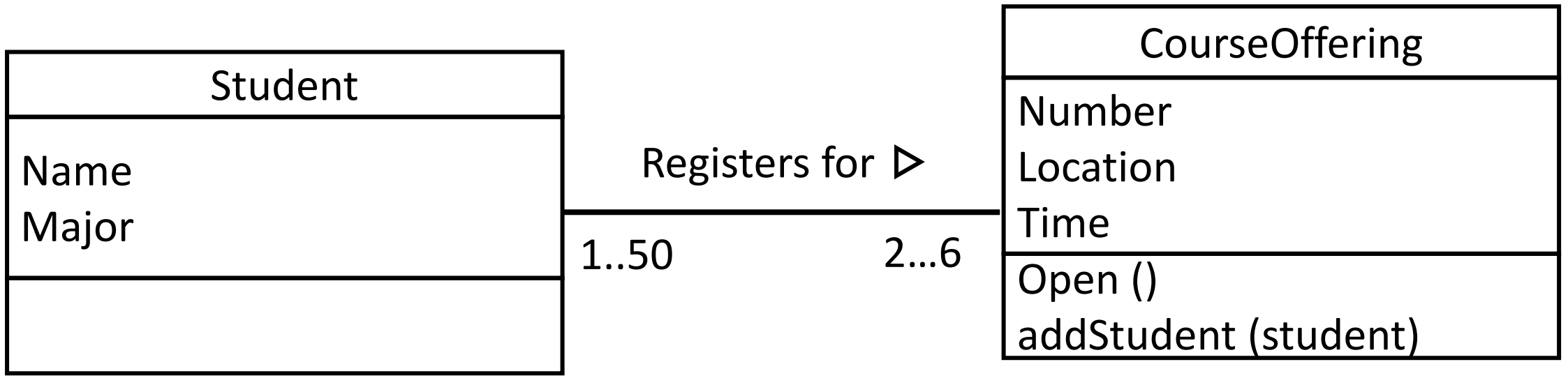


RegistrationManager uses SchedulingAllgorithm in its implementation

RegistrationManager gets a student object as a parameter

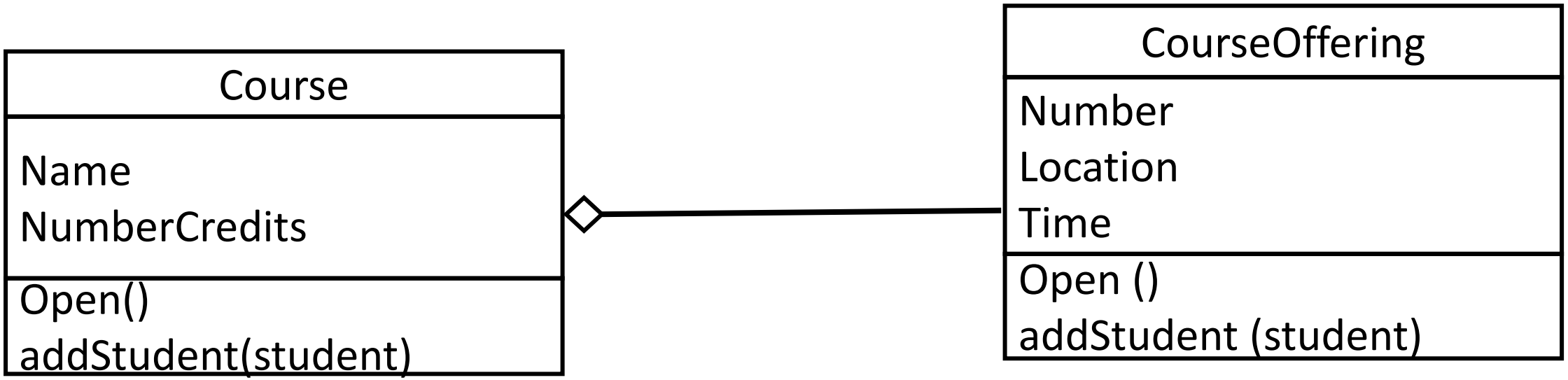
Slide adapted from Alessandro Orso

Association Example



Slide adapted from Alessandro Orso

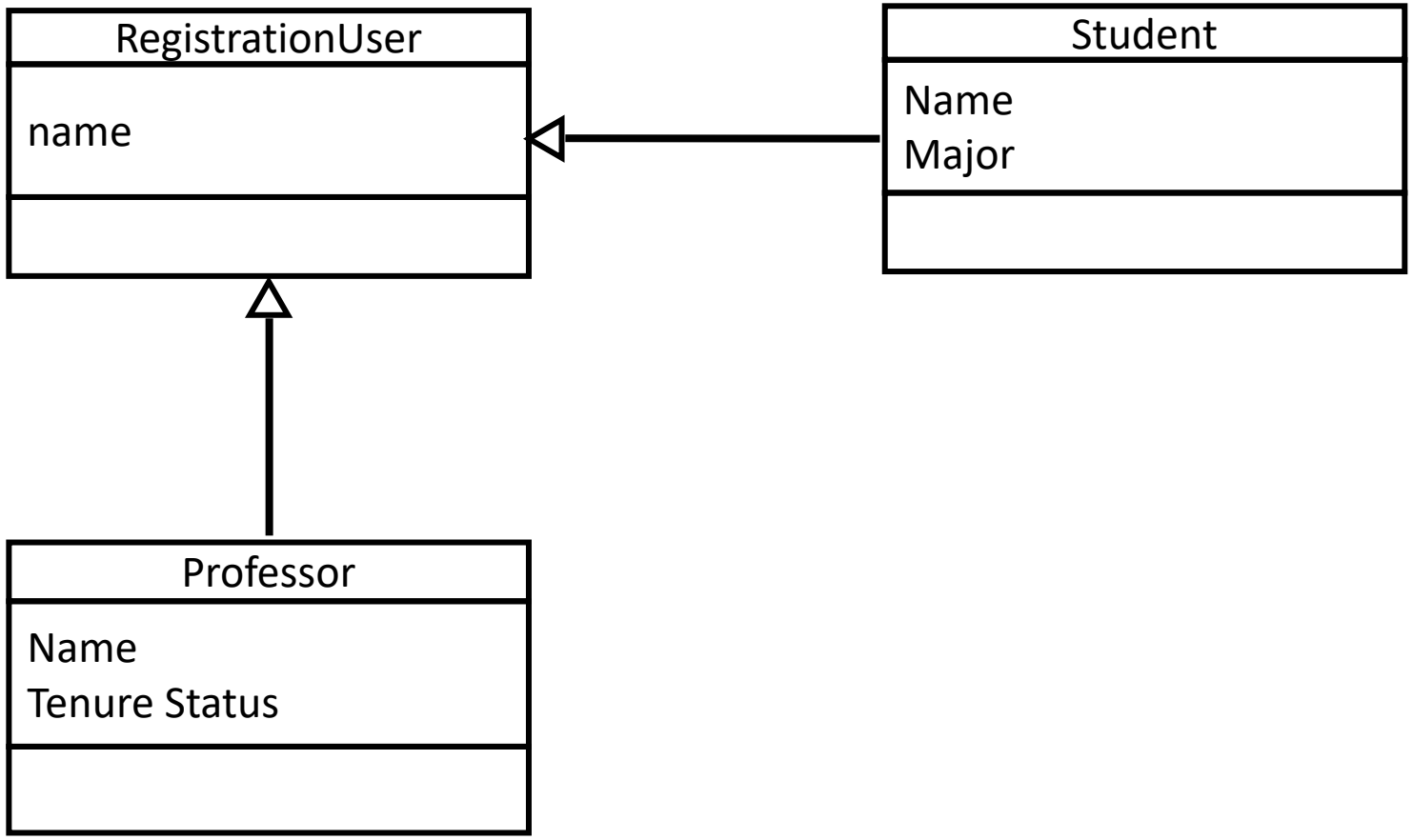
Aggregation Example



A course consists of multiple CourseOfferings

Slide adapted from Alessandro Orso

Generalization Example



Slide adapted from Alessandro Orso

Class Diagram Creation Tips

Understand the problem

Choose Good Class Names

Concentrate on the WHAT

Start with a simple diagram

Refine until you feel it is complete

Slide adapted from Alessandro Orso

Component Diagram

Static view of components and their relationships

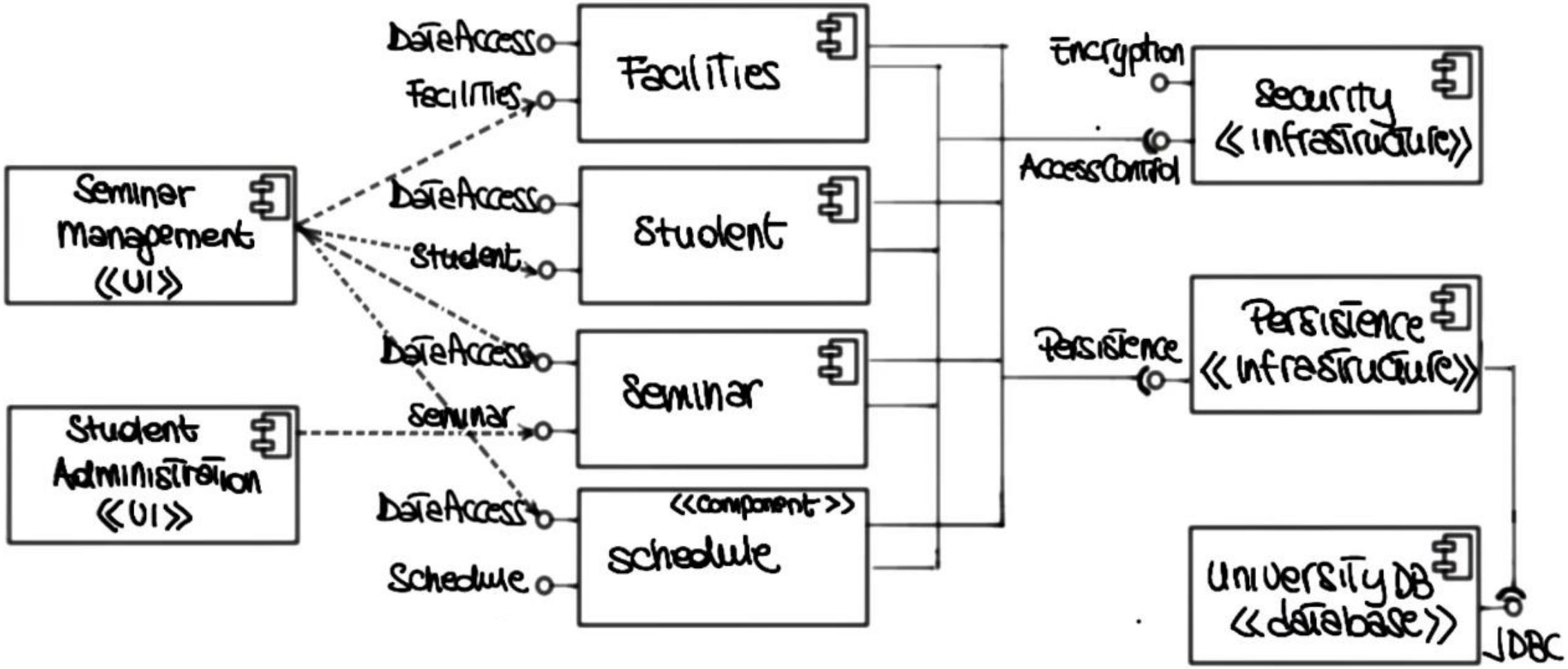
Node = Component
Set of Classes with a well-defined interface

Edge = Relationship
“Uses services of”

Can be used to represent a software architecture

Slide adapted from Alessandro Orso

Component Diagram Example



Slide adapted from Alessandro Orso

Deployment Diagram

Static deployment view of a system

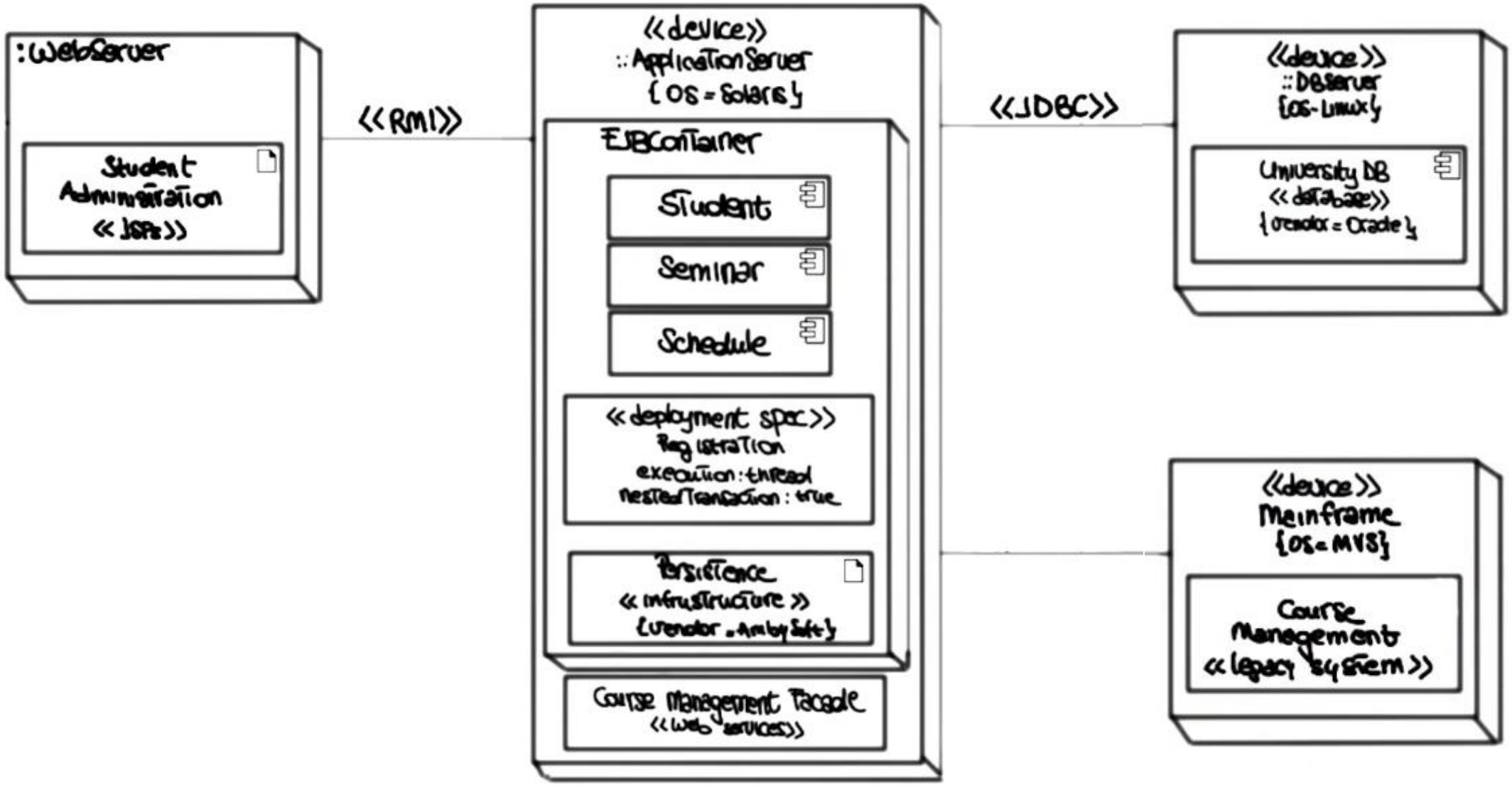
Physical Allocation of components to computational units (e.g., which component will go to the client and which will go to the server)

Node = computational unit

Edge = communication between units

Slide adapted from Alessandro Orso

Deployment Diagram Example



Slide adapted from Alessandro Orso

Unified Modeling Language

Behavioral Diagrams: represent behavioral dynamic aspects of the system

Use Case

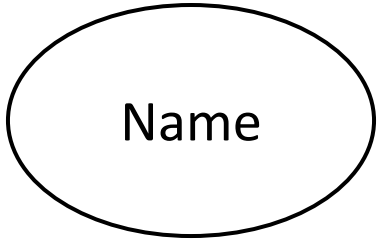
Describes the outside view of the system

- Sequence of interactions of outside entities (actors) with the system
- System actions that yields an observable result of value to the actors

AKA scenarios, scripts or user stories

Slide adapted from Alessandro Orso

Use Case: Basic Notation



Use Case

Role Name

Actor

Is the actor of

Slide adapted from Alessandro Orso

Use Case: Actor

Entity : Human or Device

Plays a role

- An entity can play more than one role
- More than one entity can play the same role

May appear in more than one use case

Slide adapted from Alessandro Orso

Actors for our example

1. The registration manager sets up the curriculum for a semester using a scheduling algorithm
2. One course may have multiple course offerings
3. Each course offering has a number, location and time
4. Students select 4 primary courses and 2 alternative courses by submitting a registration form
5. Students may use the system to add/drop courses for a period of time after registration
6. Professors use the system to receive their course offering rosters
7. Users of the registration system are assigned passwords which are used at login validation

Slide adapted from Alessandro Orso

Actors for our example

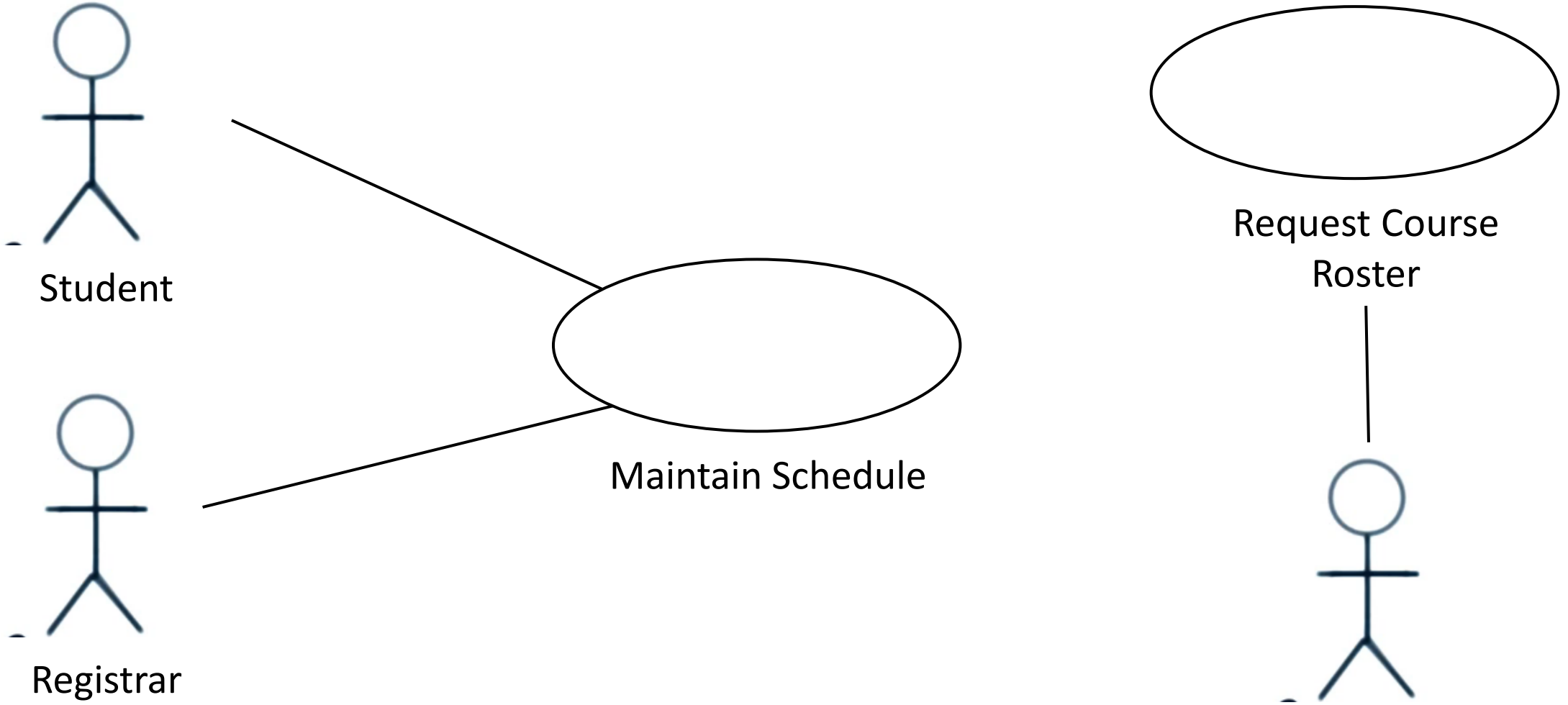
Registrar

Professor

Student

Slide adapted from Alessandro Orso

Use case diagram for our example



Slide adapted from Alessandro Orso

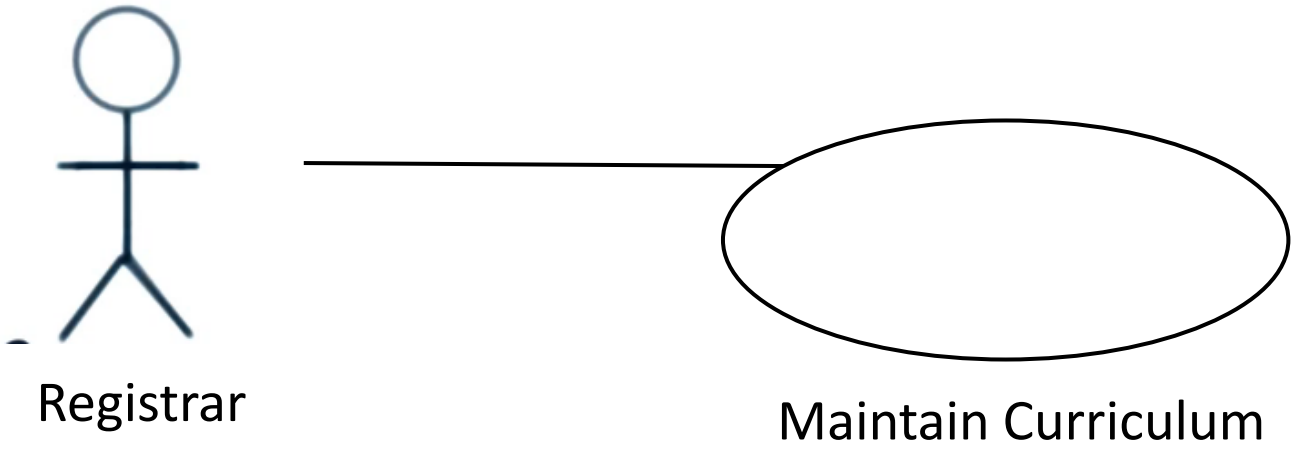
Documenting Use Cases

The behavior of a use case can be specified by describing its flow of events (formal or informal)

- How the use case starts and ends
- Normal flow of events
- Alternative flow of events
- Exceptional flow of events

Slide adapted from Alessandro Orso

Maintain Curriculum Use case: Informal Paragraph



Slide adapted from Alessandro Orso

Maintain Curriculum Use case: Informal Paragraph

1. The registration manager sets up the curriculum for a semester using a scheduling algorithm
2. One course may have multiple course offerings
3. Each course offering has a number, location and time
4. Students select 4 primary courses and 2 alternative courses by submitting a registration form
5. Students may use the system to add/drop courses for a period of time after registration
6. Professors use the system to receive their course offering rosters
7. Users of the registration system are assigned passwords which are used at login validation

Slide adapted from Alessandro Orso

Maintain Curriculum Use case: Informal Paragraph

Registrar logs onto the system and enters password

If password is valid, system asks to specify a semester

Registrar enters the desired semester

System prompts the registrar to select the desired activity: ADD, DELETE, REVIEW, or QUIT

If registrar selects ADD, system allows registrar to add a course to CourseList for a selected semester

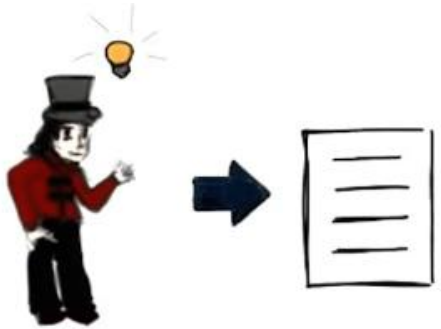
If registrar selects DELETE, system allows registrar to delete a course from CourseList for selected semester

If registrar selects REVIEW, system deploys course information in CourseList for selected semester

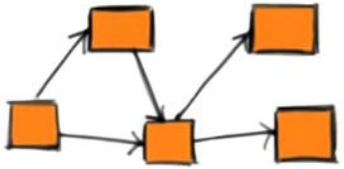
If registrar selects QUIT, system exits (use case ends)

Slide adapted from Alessandro Orso

Role of use cases



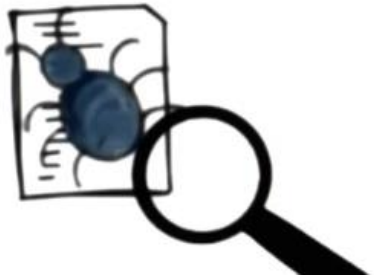
Requirements Elicitation



Architectural Analysis

User Prioritization

Planning



Testing

Slide adapted from Alessandro Orso

Use Case Diagram: Creation Tips

Use name that communicates purpose

Define one atomic behavior per use case

Define flow of events clearly (perspective of an outsider)

Provide only essential details

Factor common behaviors

Factor variants

Slide adapted from Alessandro Orso

Other Behavioral Diagrams

Sequence Diagrams: Diagrams that emphasize time ordering of messages

State Transition Diagrams: For each class, there are multiple possible states. Events cause a transition from one state to another. Actions that result from a state change

Slide adapted from Alessandro Orso