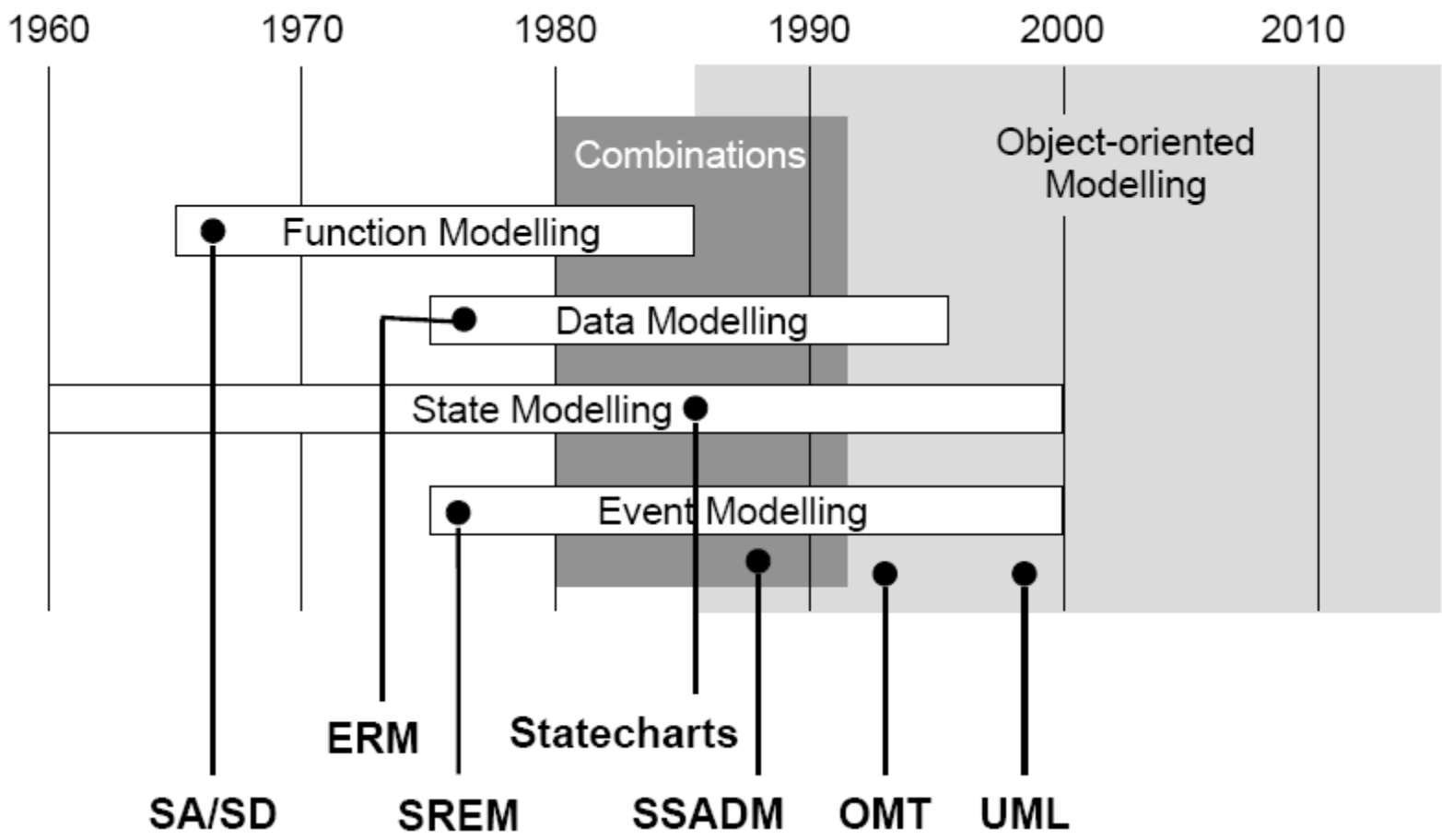
UML

Interaction Diagram and Class Diagram

DUM 05

Historical perspective



What is UML?

- UML - Unified Modelling Language
- UML combines best of
  - Data modelling concept (entity-relationship diagram, ERD)
  - Manufacturing processes modelling (workflow diagrams)
  - Object modelling
  - Component modelling
- UML is a standardized language for description, specification, construction and documentation of SW oriented system artifacts
- It can be used for modelling processes during all phases of development and for various implementation technologies

UML Concepts

UML allows to (among others):

- Display the borders of the system & its main functions through Use cases and Actors
- Illustrate realisation of use cases with interaction diagrams
- Represent static structure of the system by the Class diagram
- Model behavior of objects with State-Transition diagram
- Reveal physical implementation architecture using the Component diagram
- Extend the functionality using Stereotypes

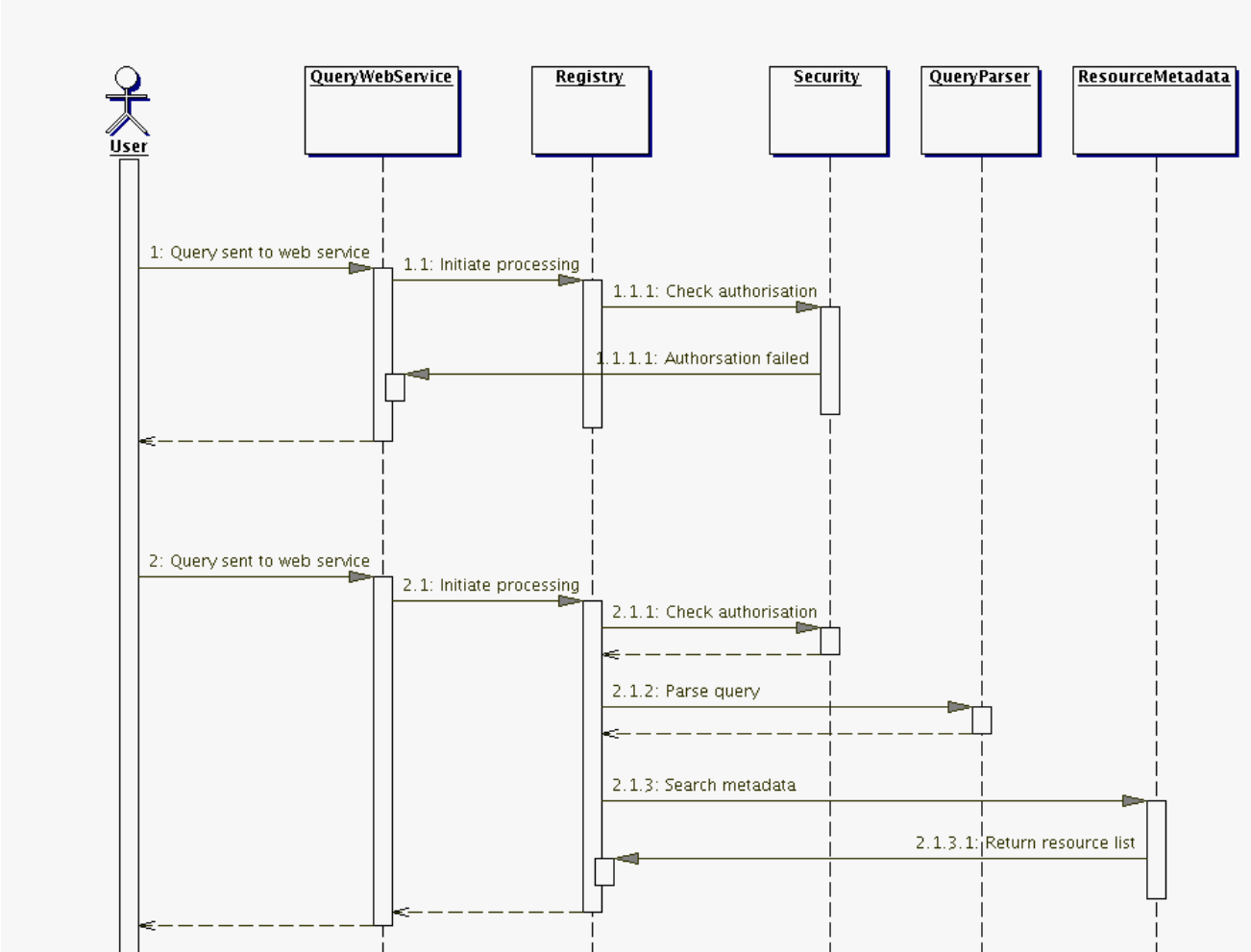
Use Case realisation

- Use Case diagram shows external view on the system
- Interaction diagrams describe how are use cases implemented as an interaction between groups of objects

- Two types of interaction diagrams:
  - Sequence diagrams
  - Communication diagrams

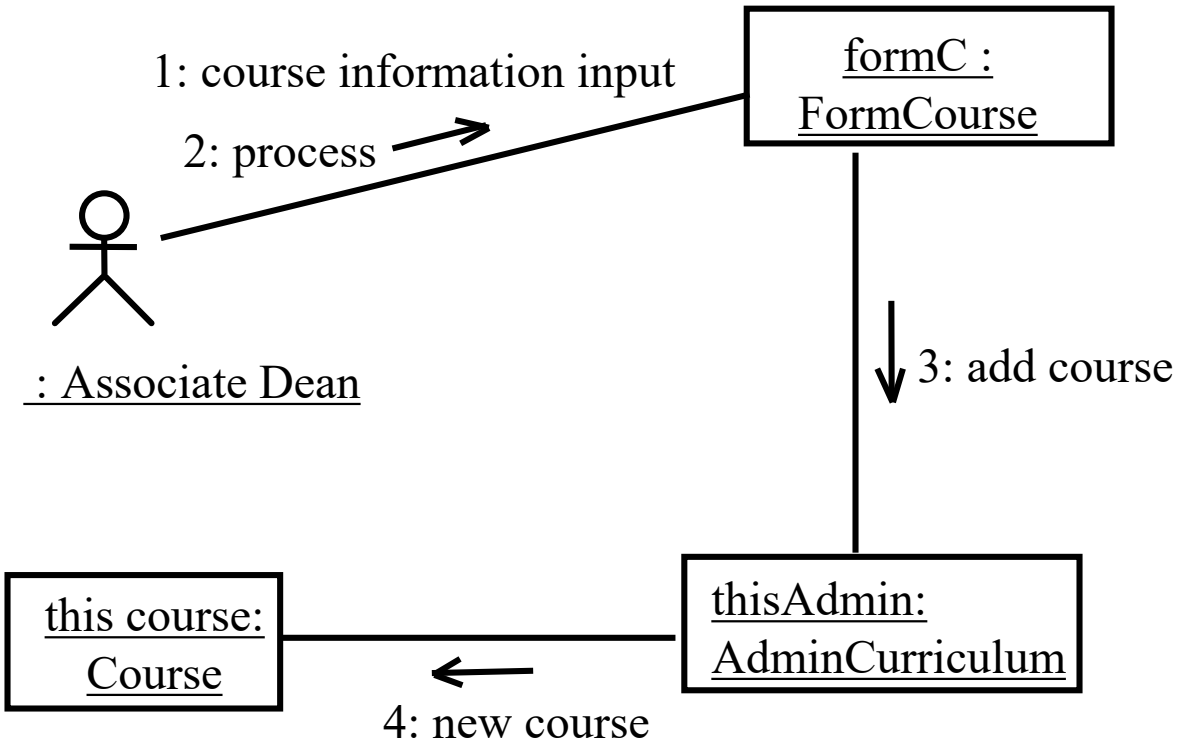
Sequence diagrams

Sequence diagrams displays interaction between objects. This interaction is ordered into a time sequence



Communication diagram

Communication diagram illustrates the interaction between objects and their mutual interconnection



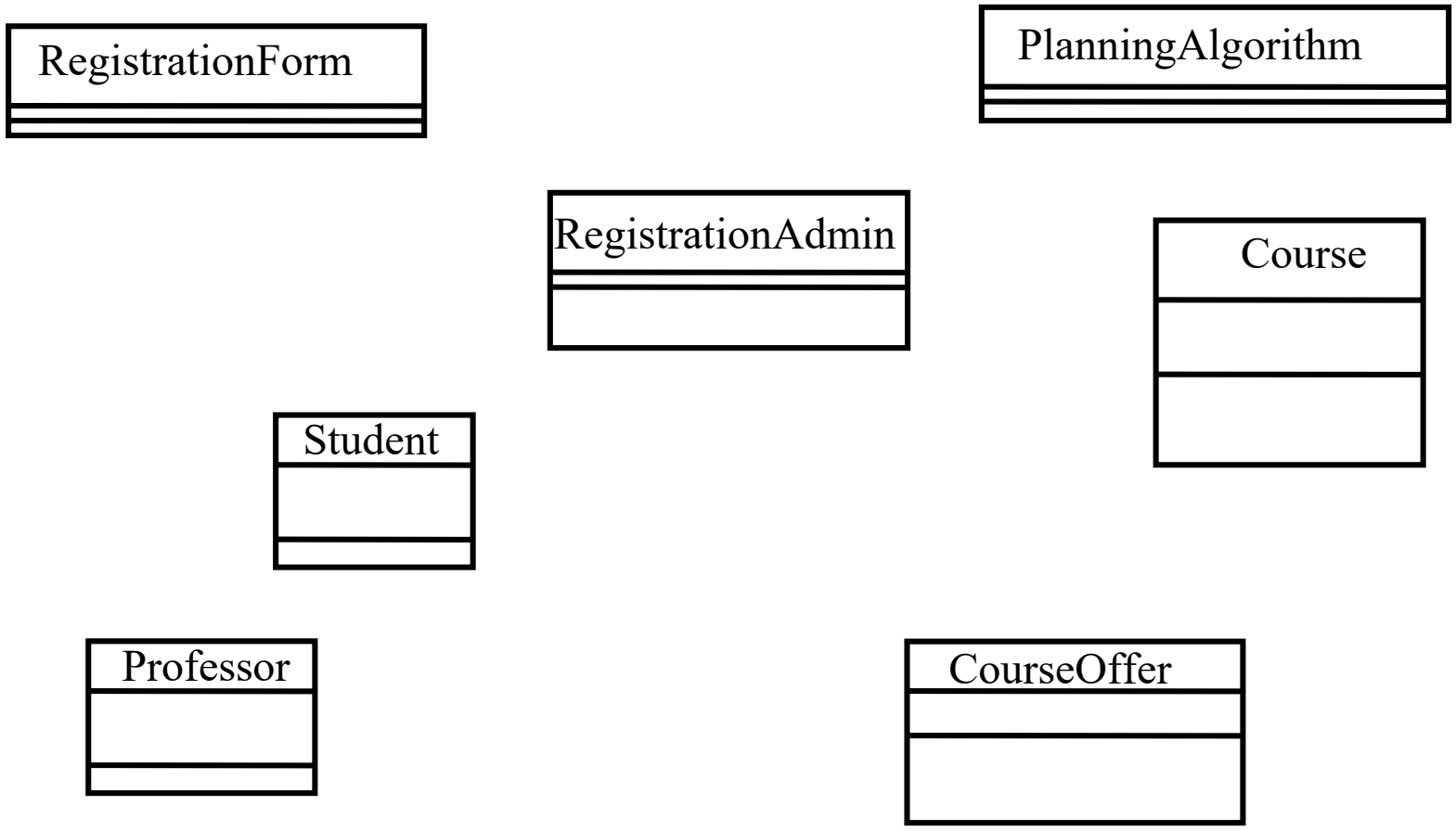
Class diagram

- Class diagram displays the existence of classes and their relationships through the logic perspective of a system
- UML modelling elements used in class diagrams:
  - Classes, their structure and behavior
  - association, agregation, dependency and inheritance relationships
  - Indicators of multiplicity and navigations
  - Name of roles

Classes

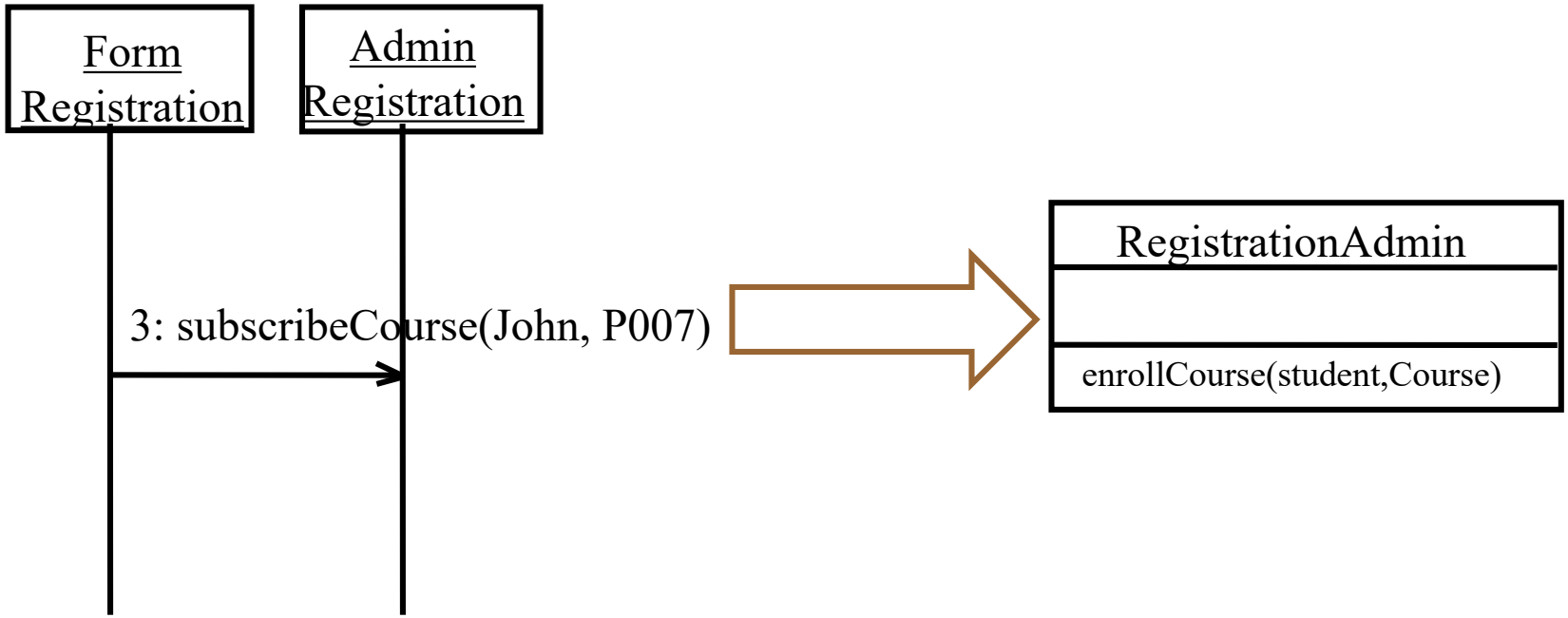
- Class is a collection of objest with identical structure, behavior and semantics
- Classes are found by re-examination of object in sequence diagrams and collaboration diagrams. Class is depicted as a rectangle with three sections
- Classes should be named using dictionary of the subjective domain
- There should be a standard for creating classes names
  - E.g. All classes will be in form of a singular noun with first capital letter

Classes


RegistrationForm
PlanningAlgorithm
RegistrationAdmin
Course
Student
Professor
CourseOffer

Operations

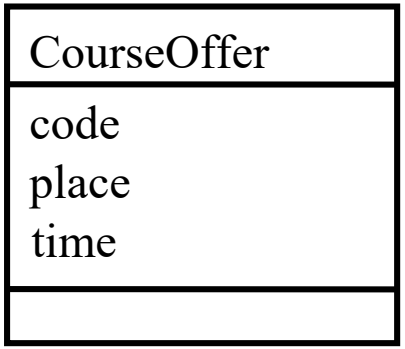
- Behavior of classes is represented by their operations
- Operations can be found after re-examination of interaction diagram



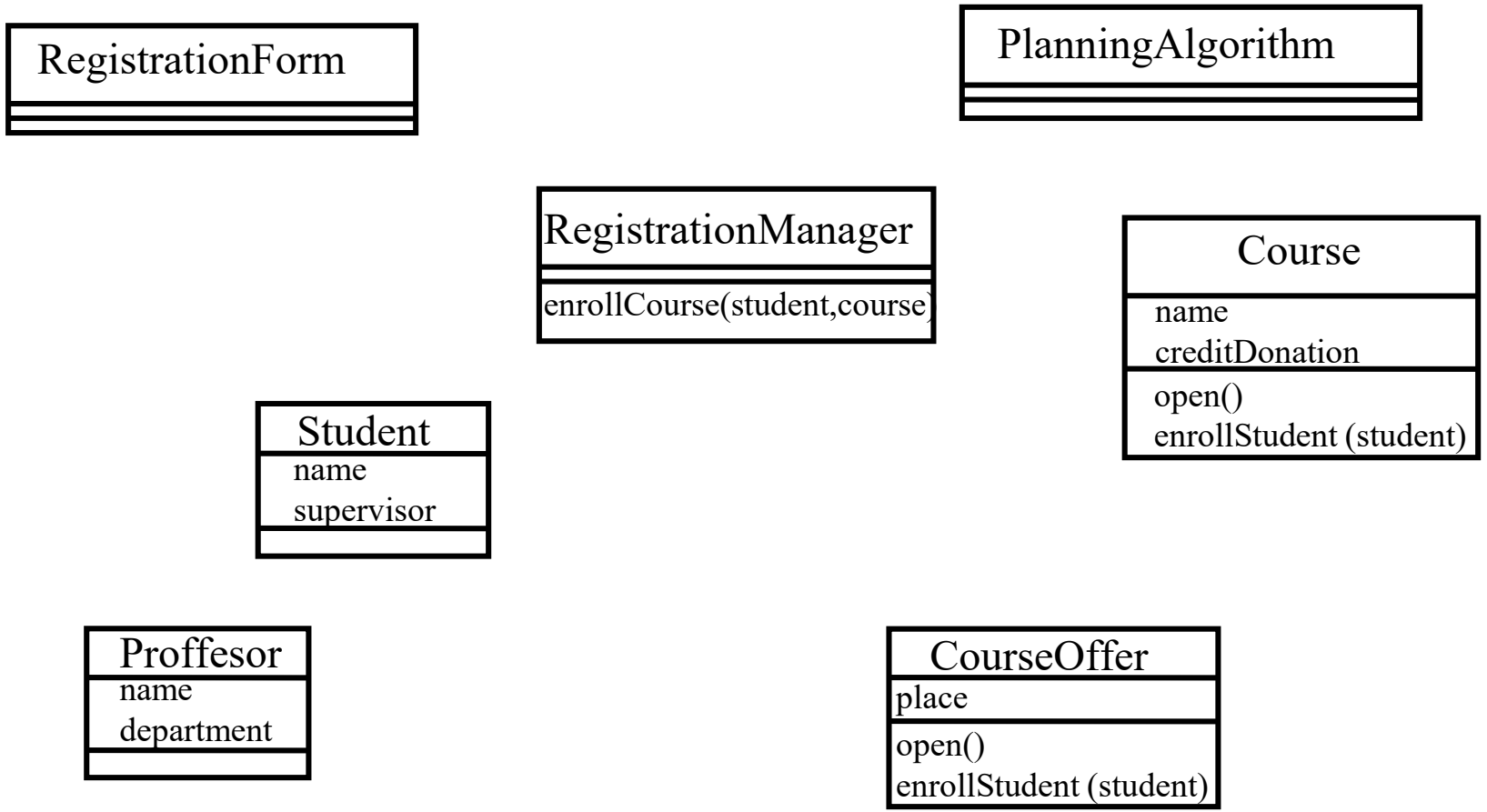
Attributes

- A structure of the class is represented by its attributes
- Attributes can be found by re-examination of class definition, requirements and by using knowledge of given domain

Every course offer consists of code, place and time



Classes


RegistrationForm
PlanningAlgorithm
RegistrationManager
enrollCourse(student,course)
Course
name
creditDonation
open()
enrollStudent (student)
Student
name
supervisor
Proffesor
name
department
CourseOffer
place
open()
enrollStudent (student)

Relationships

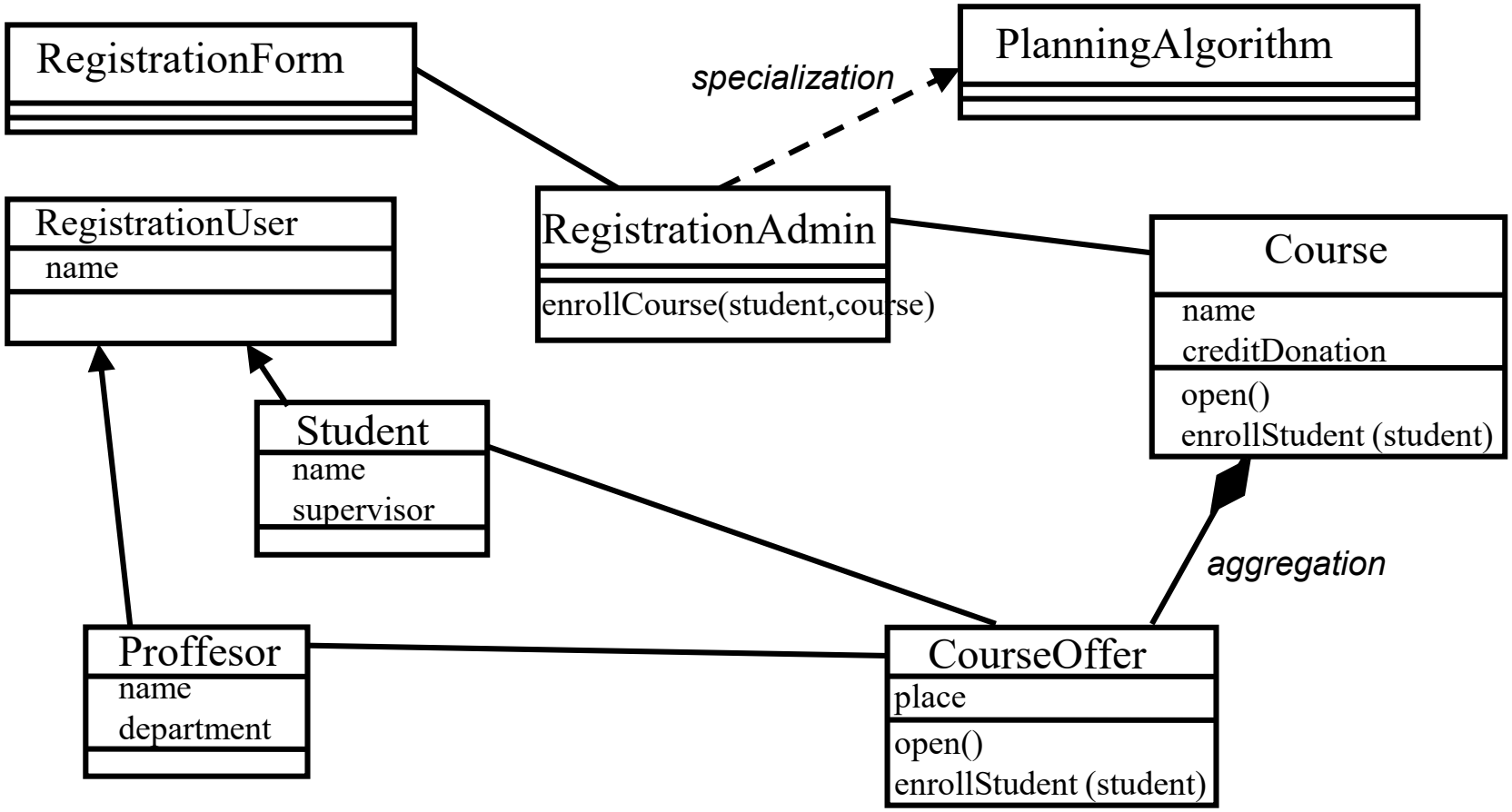
- Relationships provide channel for communication between objects
- Sequence and/or cooperation diagrams are used to define the connections between objects that are needed for certain behavior – if some objects need to „talk“ to each other, then there must be a relationship between
- There are 2 general types of relationships:
  - Between classes
  - Between instances

Relationship between classes

- Generalization / Inheritance is a strong relationship between superclass and its subclasses
- E.g. Shape ← [Triangle; Rectangle; Circle]
- Inherits attributes, methods, relationships
- Common attributes, operations and/or relationships are shown at the highest applicable level of hierarchy

- Specialization / Dependency is a weak relationship where client uses some methods of the provider but has no semantic knowledge of the provider

Inheritance



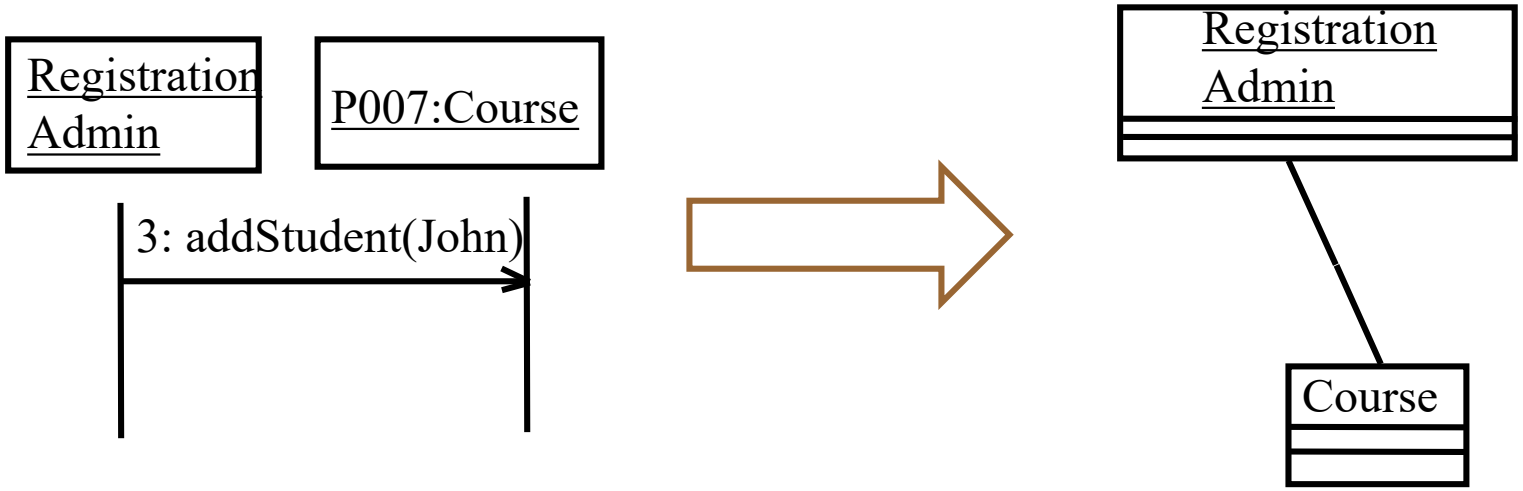
Relationships between instances

- Association is mutual connection between classes
  - Association is represented by a link connecting related classes
- Aggregation is a form of weaker dependency, it portrays a relation between whole and its parts, but the parts can exist on its own (e.g. Student - University)
  - Aggregation is represented by link connecting related classes, while empty diamond sign is located next to class representing the whole
- Composition is a form of stronger dependency, where component ceases to exist if the whole is deleted (e.g. Invoice – Item)
  - Diamond sign is filled

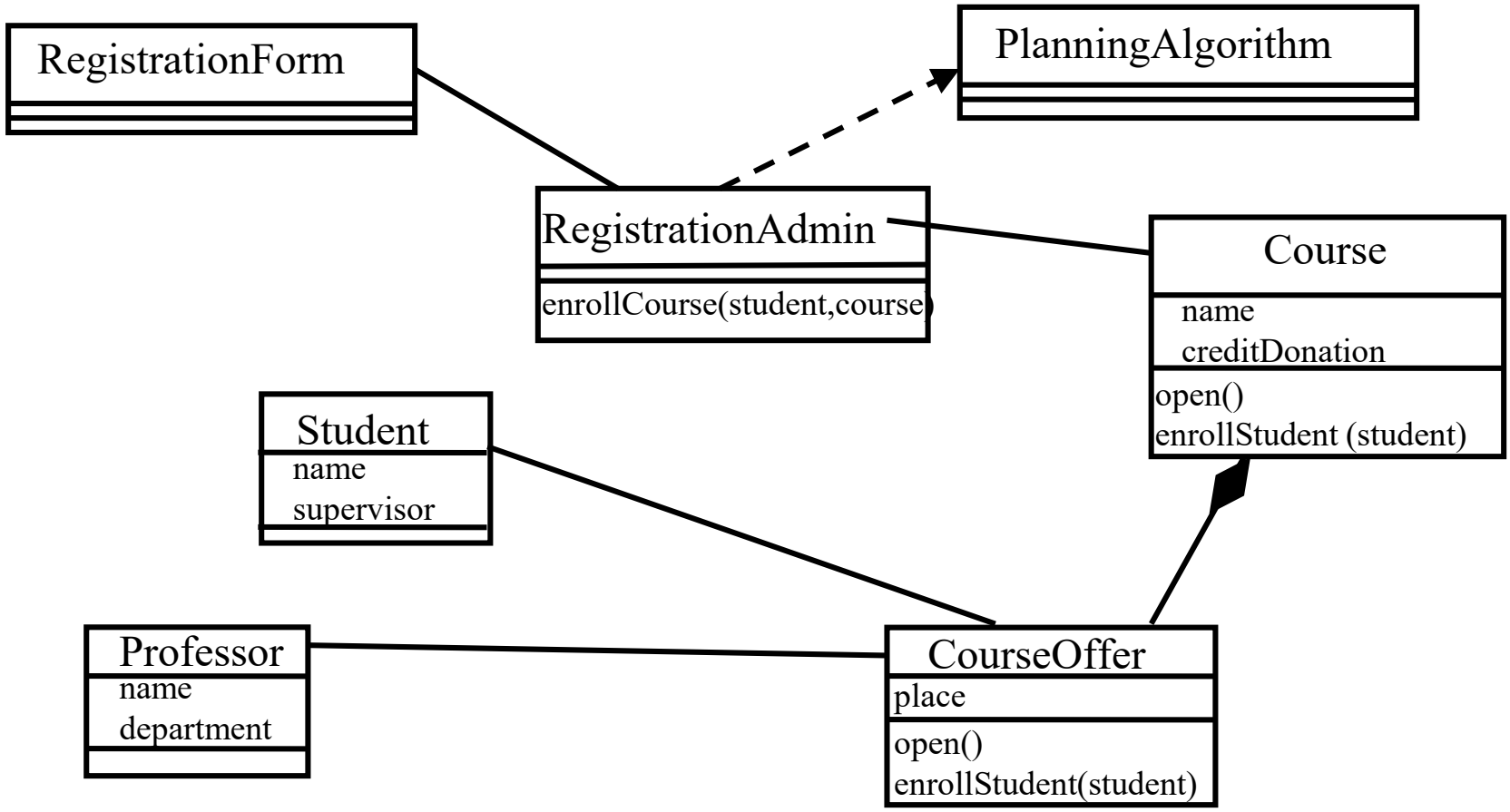
Finding relationships

Relationships are found after re-examination of interaction diagrams

- If two object „communicate“, then there must be a relationship between them



Relationships



Multiplicity and Navigation

Multiplicity defines how many objects are participating in the relationship

- Multiplicity is a number of instances of one class with regard to ONE instance of other class
- There must be two multiplicities for each association and aggregation: for each end of the relationship

Although association and aggregation are implicitly bidirectional, it might be suitable to restrict the navigation to just one direction

The added arrow shows direction of navigation if restricted navigation is applied

Multiplicity and navigation