

MACIASZEK, L.A. (2007):
Requirements Analysis and System Design, 3rd ed.
Addison Wesley, Harlow England
ISBN 978-0-321-44036-5

Chapter 3
Requirements Specification

© Pearson Education Limited 2007

Topics

- State specifications
- Behavior specifications
- State change specifications

2. *State specifications*

- provide a *structure* (or *static*) view of the system
- the main task is to define *classes* in an application domain

State

- **Object state** is determined by the values of its attributes and associations
- **State specification:**
 - Model of data structures
 - Static view on the system
 - Class operations left out in initial specs
 - Emphasis on entity classes ('business objects')

Modeling classes

- Cornerstone of OO development – a system is a set of collaborating (and classified) objects
- Iterative and incremental process
- CASE tool
 - For collaborative development
 - For personal productivity otherwise

Discovering classes

- No two analysts will come up with the identical class models for the same application domain
- Discovering classes
 - Noun phrase
 - Common class patterns
 - Use case driven
 - CRC
 - Mixed

Noun phrase approach

- Nouns considered candidate classes
- Three kinds of candidate classes
 - Irrelevant (can be skipped)
 - Relevant
 - Fuzzy
- Assumes that the Requirements Document is complete and correct

Common class pattern approach

- Derives candidate classes from the classification theory of objects
- One possible classification pattern:
 - Concept (e.g. `Reservation`)
 - Event (e.g. `Arrival`)
 - Organization (e.g. `Department`)
 - People (e.g. `Passenger`)
 - Place (e.g. `TravelOffice`)
- Just a guidance
- Only loosely bound to user requirements
- Possible naming misinterpretations

Use case driven approach

- Assumes that:
 - Use Case Diagrams (and possibly some high-level Sequence Diagrams) have been developed
 - Narrative descriptions for each use case exist
- Similar to the noun phrase approach
- Function-driven (problem-driven)
- Relies on the completeness of use case models

CRC approach

- CRC – classes, responsibilities, collaborators
- More than a technique for class discovery
- Animated brainstorming sessions
- Identifies classes from the analysis of how objects collaborate to perform business functions (use cases)
- Suitable also for:
 - Verification of classes discovered with other methods
 - Determination of class properties

Mixed approach

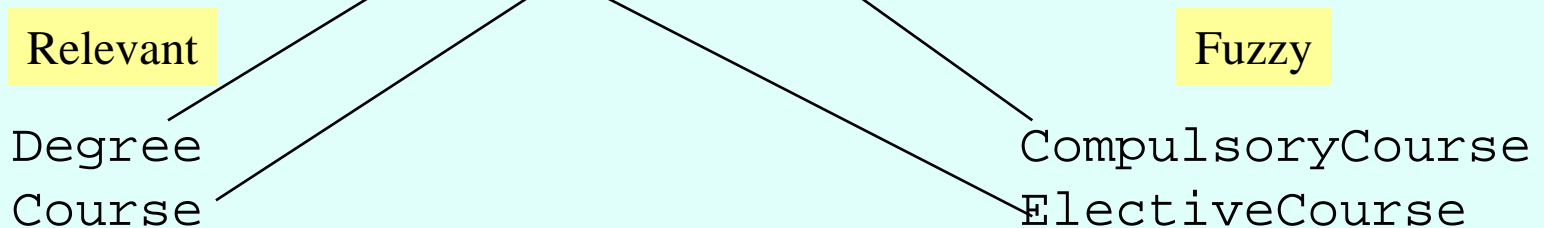
- Perhaps with elements of all four previous approaches
- Middle-out rather than top-down or bottom-up
- One possible scenario:
 - Initial classes – domain knowledge
 - Common class patterns approach to guide
 - Noun phrase approach to add more classes
 - Use case approach to verify
 - CRC to brainstorm

Guidelines for class discovery

- Statement of purpose
- Description for a set of objects
 - Singleton classes
- Houses a set of attributes
 - Identifying attributes - keys
 - OID
- Class or attribute?
- Houses a set of operations (what does the class do?)

Example 3.1 – University Enrolment

- Consider the following requirements for the University Enrolment system and identify the candidate classes:
 - Each university degree has a number of compulsory courses and a number of elective courses.



Example 3.1 – University Enrolment

■ More requirements:

- Each course is at a given level and has a credit-point value
- A course can be a part of any number of degrees
- Each degree specifies a minimum total credit points value required for degree completion
- Students may combine course offerings into programs of study suited to their individual needs and leading to the degree in which a student is enrolled

Example 3.1– University Enrolment (solution)

Relevant classes

Fuzzy classes

Course

CompulsoryCourse

Degree

ElectiveCourse

Student

StudyProgram

CourseOffering

Example 3.2 – Video Store

- Consider the following requirements for the Video Store system and identify the candidate classes:
- The video store keeps an extensive library of current and popular movie titles in stock . A particular movie may be held on videotapes or disks.

Relevant

MovieTitle
VideoTape
VideoDisk

Irrelevant

VideoStore
Stock
Library

Example 3.2 – Video Store

■ More requirements:

- A movie can be rented for a particular rental duration (expressed in days), with a rental charge for that period.
- The video store must be able to answer immediately any enquiries about a movie's stock availability.
- The current condition of each tape and disk must be known and recorded, together with generic information about the percentage of videotapes and disks in excellent renting condition.

Example 3.2 – Video Store (solution)

Relevant classes

MovieTitle

VideoMedium

VideoTape

VideoDisk

Fuzzy classes

RentalRates

Example 3.3 – Contact Management

- Consider the following requirements for the Contact Management system and identify the candidate classes:
 - The system supports the function of “keeping in touch” with all current and prospective customers so as to be responsive to their needs and to win new contracts for products and services offered by the organization.
 - The system stores the names, phone numbers, postal and courier addresses, etc. of organizations and contact persons in these organizations.
 - The system allows employees to schedule tasks and events that need to be carried out with regard to relevant contact persons. Employees schedule the tasks and events for other employees or for themselves.
 - A task is a group of events that take place to achieve a result. The result may be to convert a prospective customer to a customer, to organize product delivery, or to solve a customer’s problem. Typical types of event are phone call, visit, sending a fax, and arranging for training.

Example 3.3 – Contact Management (solution)

Relevant classes

Organization

Contact

Employee

Task

Event

Fuzzy classes

CurrentOrg

ProspectiveOrg

PostalAddress

CourierAddress

Specifying classes

■ In Class Diagram

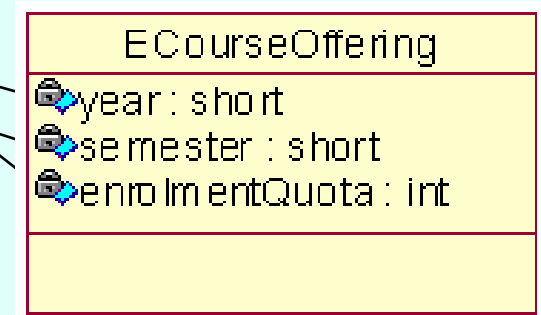
- Each class given a name (and possibly a code)
- Singular noun
 - Recommendation – multiple words joined; each word starting with a capital letter (e.g. `PostalAddress`)
- Meaningful
- Short (less than 30 characters)

■ Class properties to be defined

- Attributes (initially those that capture interesting object states)
 - Recommendations:
 - small letters; underscore to separate words (e.g. `street_name`)
 - start with a small letter; capitalize successive words (`streetName`)
- Operations (can be delayed till later analysis stages or even till design)

Example 3.4 – University Enrolment

- Refer to Example 3.1
- Consider the following additional requirements from the Requirements Document:
 - A student's choice of courses may be restricted by timetable clashes and limitations on the number of students who can be enrolled on the current course offered.





Example 3.4 – University Enrolment




- More requirements:
 - A student's proposed program of study is entered in the on-line enrolment system.
 - The system checks the program's consistency and reports any problems.
 - The problems need to be resolved with the help of an academic adviser.
 - The final program of study is subject to academic approval by the delegate of the Head of Division and it is then forwarded to the Registrar.

Example 3.4 – University Enrolment (solution)



EDegree

 <<PK>> degreeName : String
 totalCreditPoints : int



ECourse

 <<PK>> courseCode : String
 <<CK>> courseName : String
 creditPoints : short




EStudyProgram

 year : short
 semester : short

EStudent

 <<PK>> studentId : String
 studentName : String

ECourseOffering

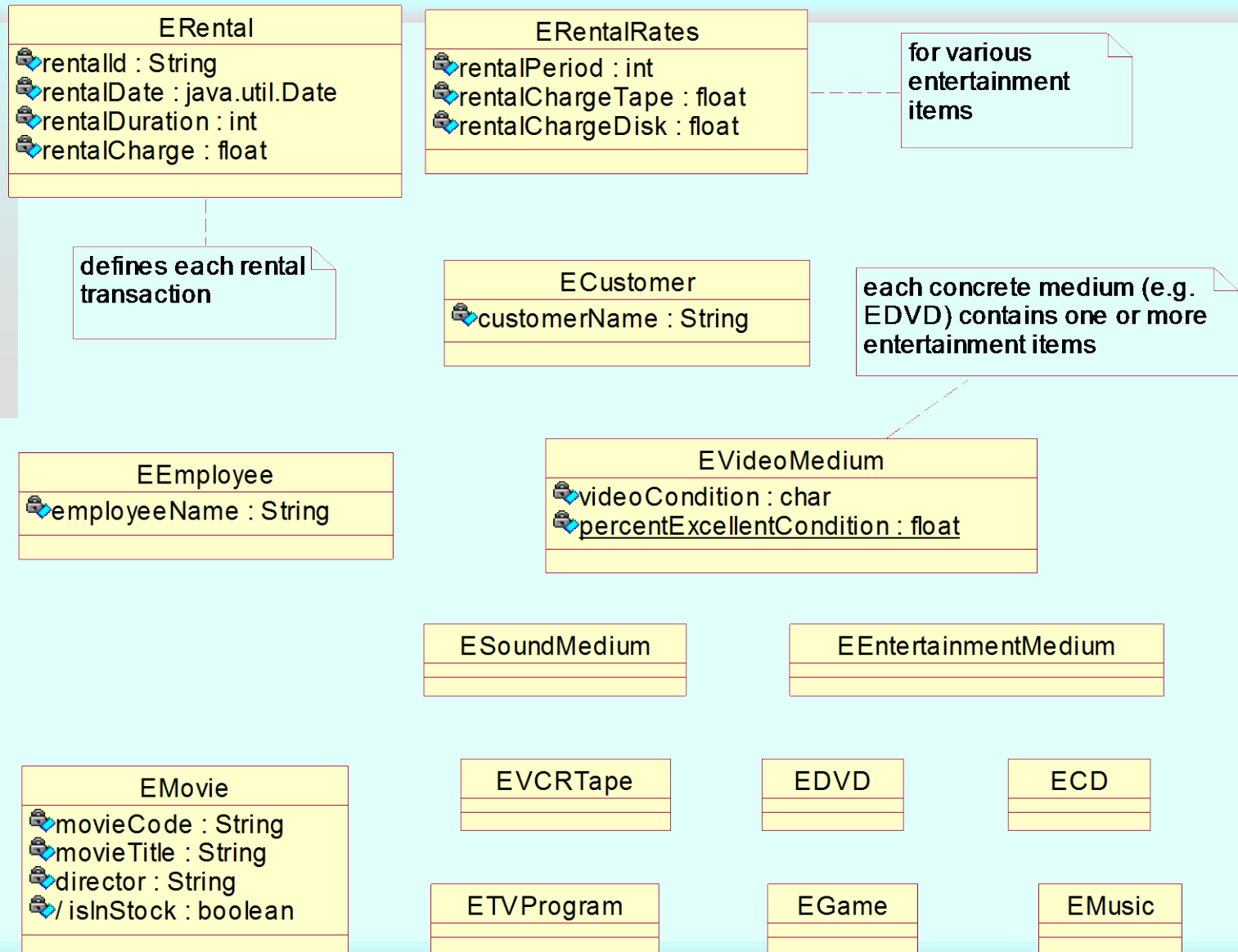
 year : short
 semester : short
 enrolmentQuota : int

Example 3.5 – Video Store

■ More requirements:

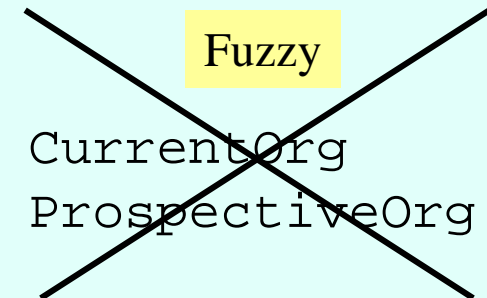
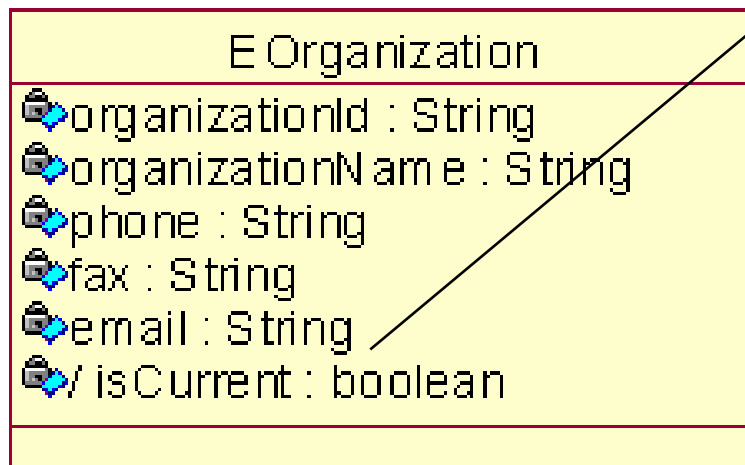
- The rental charge for entertainment items (such as movie) differs depending on the item category and on the entertainment medium containing the item. In general, an entertainment medium can be a video medium or just a sound medium. A video medium can be a VCR tape, DVD or game CD. A music CD is a sound medium, but some music CDs are available in video formats, such as VCD or DVD-A.
- The system stores information about employees and identifies the employee responsible for each rental transaction performed on behalf of a customer.
- A separate transaction is generated for renting for different durations. Items in each rental transaction can relate to more than one medium (as long as all items are rented for the same duration). A medium can contain movies, TV programs, games, or music.
- The employees of the video store tend to remember the codes of the most popular movies. They frequently use a movie code, instead of movie title, to identify the movie. This is a useful practice, because the same movie title may have more than one release by different directors.

Example 3.5 – Video Store (solution)



Example 3.6 – Contact Management

- Refer to Example 3.3 and consider the following additional information
 - A customer is considered current if a contract with that customer exists for the delivery of our products or services. However, contract management is outside the scope of our system.

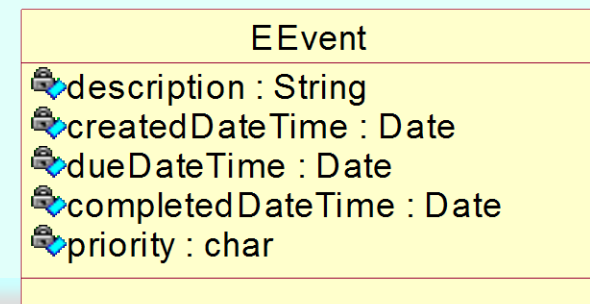
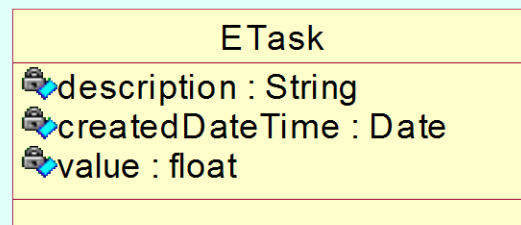
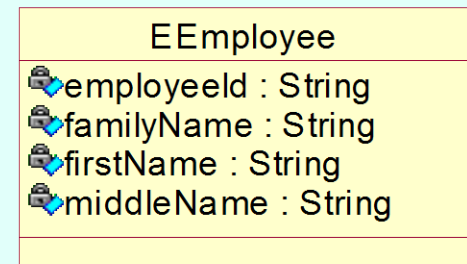
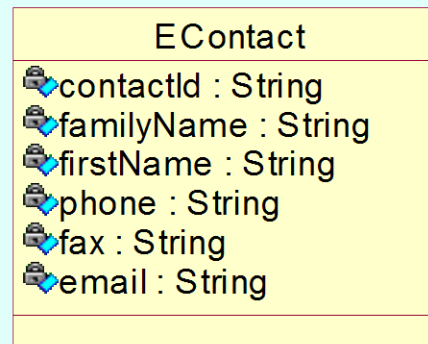
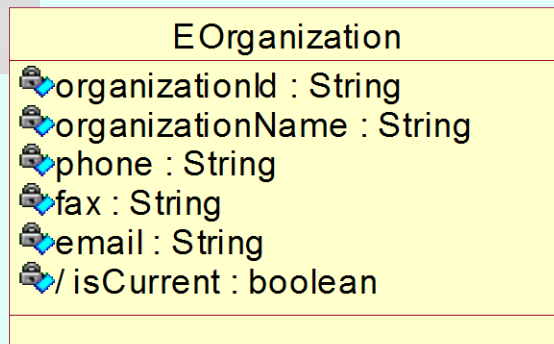
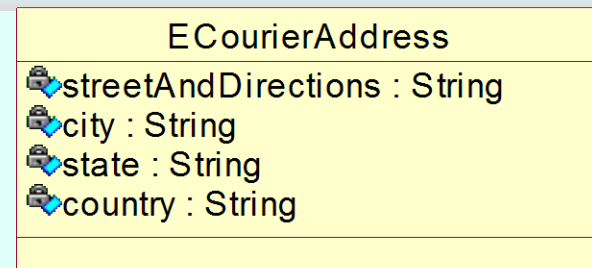
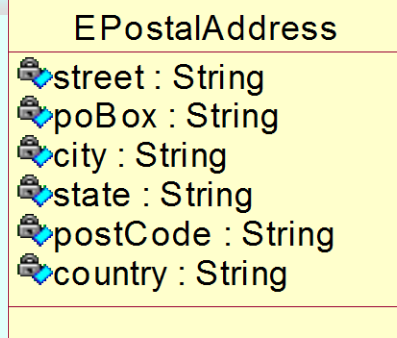


Example 3.6 – Contact Management

■ More requirements:

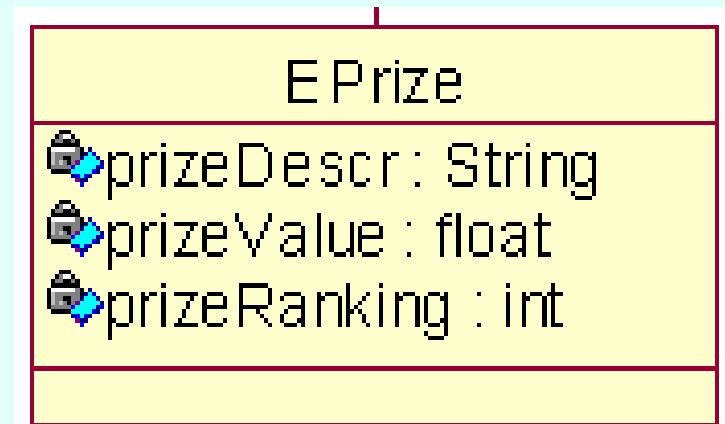
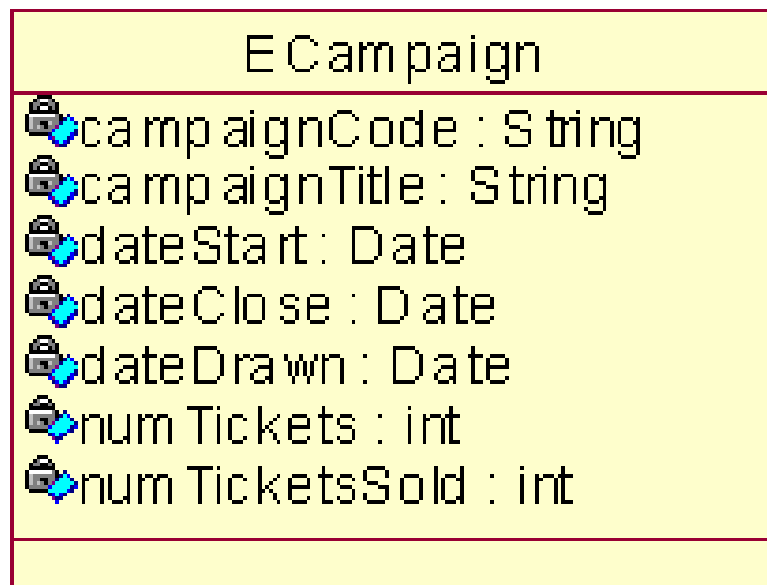
- Reports on contacts based on postal and courier addresses (e.g. find all customers by post code)
- Date and time of the task creation are recorded
- The "money value" of a task can be stored
- Events for the employee are displayed on the employee's screen in the calendar-like pages (one day per page).
 - The priority of each event (low, medium or high) is visually distinguished on the screen
- Not all events have a “due time” - some are “untimed”
- Event creation time cannot be changed, but the due time can.
- Event completion date and time are recorded
- The system stores identifications of employees who created tasks and events, who are scheduled to do the event (“due employee”), and who completed the event

Example 3.6 – Contact Management (solution)



Example 3.7 - Telemarketing

- Consider the following additional information
 - Each campaign
 - Has a title that is generally used for referring to it
 - Has also a unique code for internal reference
 - Runs over a fixed period of time
 - Soon after the campaign is closed, the prizes are drawn and the holders of winning tickets are advised



Example 3.7 - Telemarketing

■ More requirements:

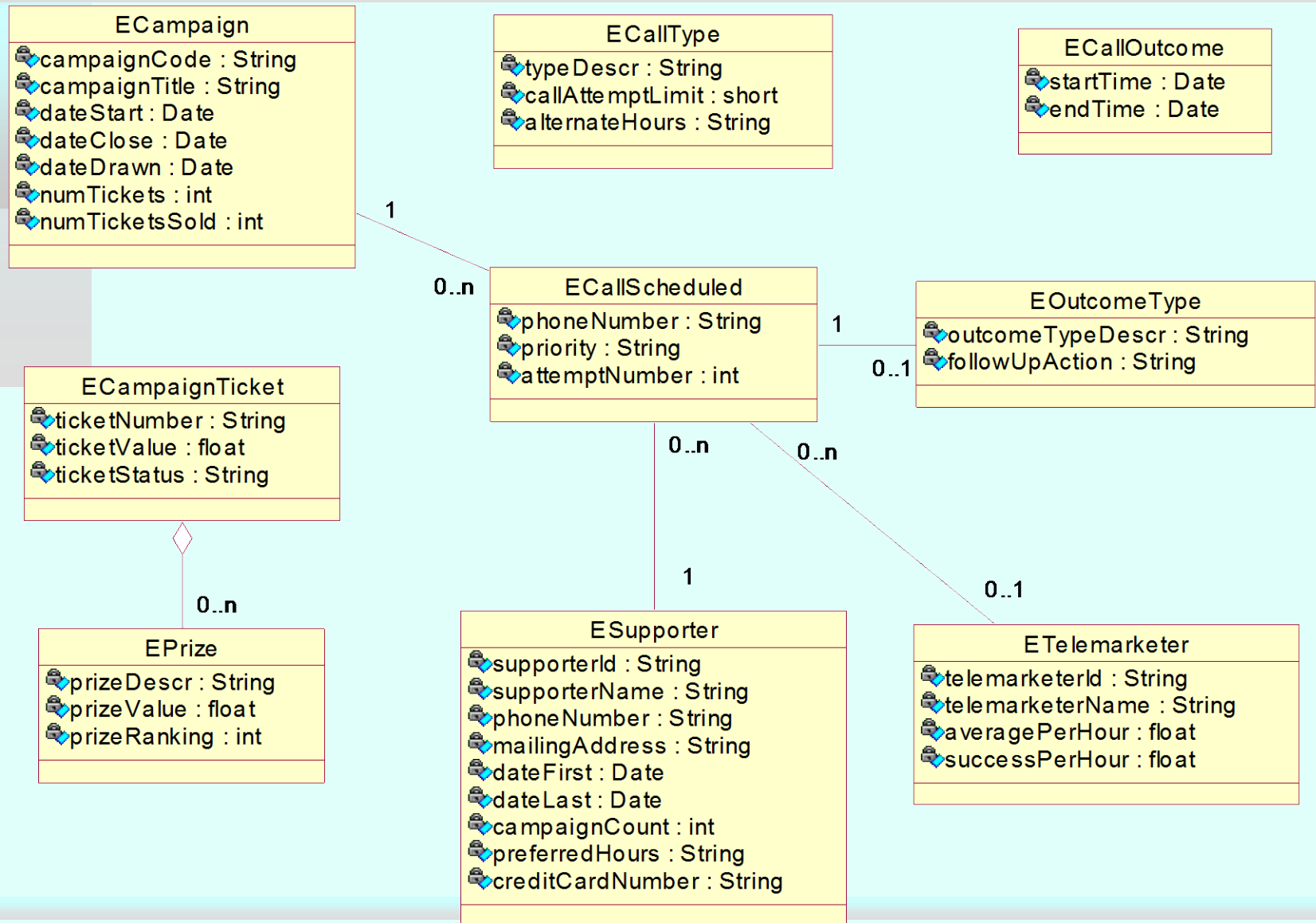
- Tickets are uniquely numbered within each campaign
- The total number of tickets in a campaign, number of tickets sold so far, and the current status of each ticket are known (e.g. available, ordered, paid for, prize winner)
- To determine the performance of the society's telemarketers, the duration of calls and the successful call outcomes (i.e. resulting in ordered tickets) are recorded
- Extensive information about supporters is maintained
 - Contact details (address, phone number, etc.)
 - Historical details such as the first and most recent dates when a supporter had participated in a campaign
 - Any known supporter's preferences and constraints (e.g. times not to call, usual credit card number)

Example 3.7 - Telemarketing

■ More requirements:

- Telemarketing calls are made according to their priorities
- Calls which are unanswered or where an answering machine was found, are rescheduled
 - Times of repeat calls are alternated
 - Number of repeat calls is limited
 - Limits may be different for different call types (e.g. a normal "solicitation" call may have different limit than a call to remind a supporter of an outstanding payment)
- Call outcomes are categorized - success (i.e. tickets ordered), no success, call back later, no answer, engaged, answering machine, fax machine, wrong number, disconnected.

Example 3.7 – Telemarketing (solution)



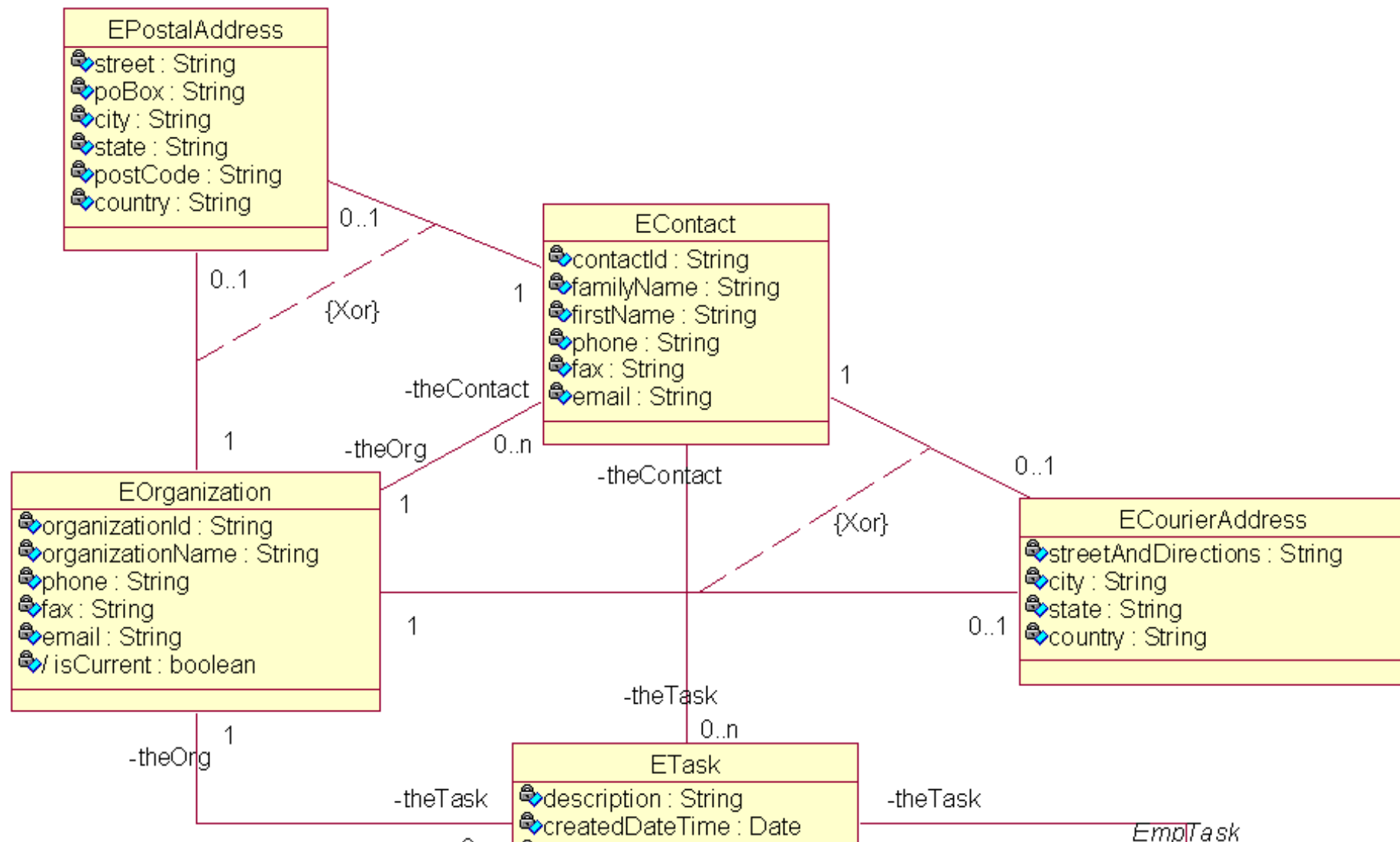
Discovering associations

- Side effect of discovering classes
- Some attributes are associations
- “Dry-run” of use cases to discover more associations
- Avoid ternary associations
- Cycles of associations that do not commute

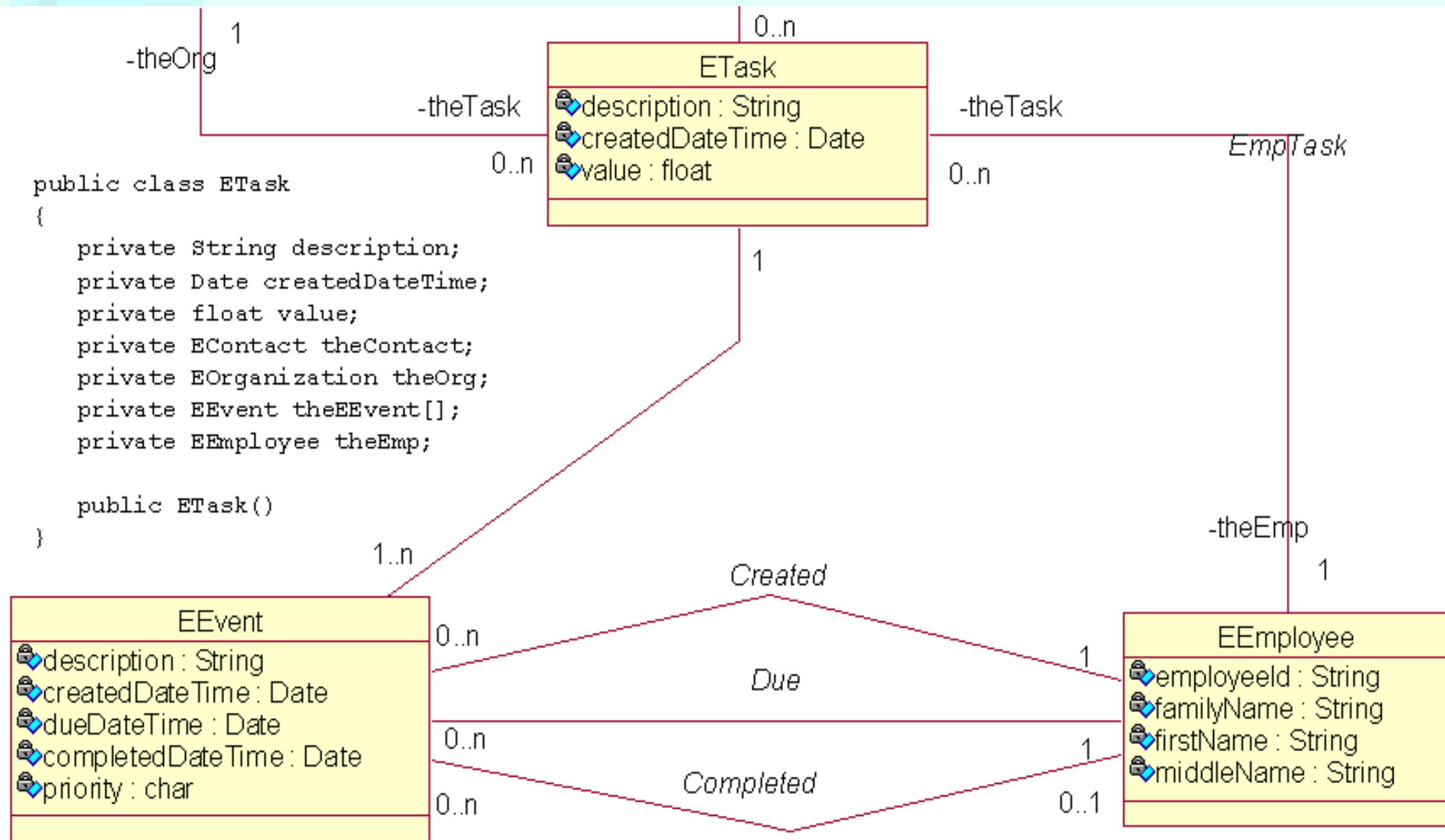
Specifying associations

- Naming associations
 - Recommendation – small letters; capitalizing the first letters of successive words (e.g. empTask)
- Naming association roles
- Determining multiplicity
 - Lower and/or upper multiplicity bounds can be omitted initially
- Role names for recursive associations

Example 3.8 – Contact Management (solution – 1)



Example 3.8 – Contact Management (solution – 2)



Modeling aggregation and composition

- Four semantics for aggregation possible
 - ExclusiveOwns (e.g. Book has Chapter)
 - Existence-dependency
 - Transitivity
 - Asymmetry
 - Fixed property
 - Owns (e.g. Car has Tire)
 - No fixed property
 - Has (e.g. Division has Department)
 - No existence dependency
 - No fixed property
 - Member (e.g. Meeting has Chairperson)
 - No special properties except membership

Discovering aggregation

- Discovered in parallel with discovery of associations
- The litmus test phrases
 - “has”
 - “is-part-of”
- Can relate more than two classes

Specifying aggregation

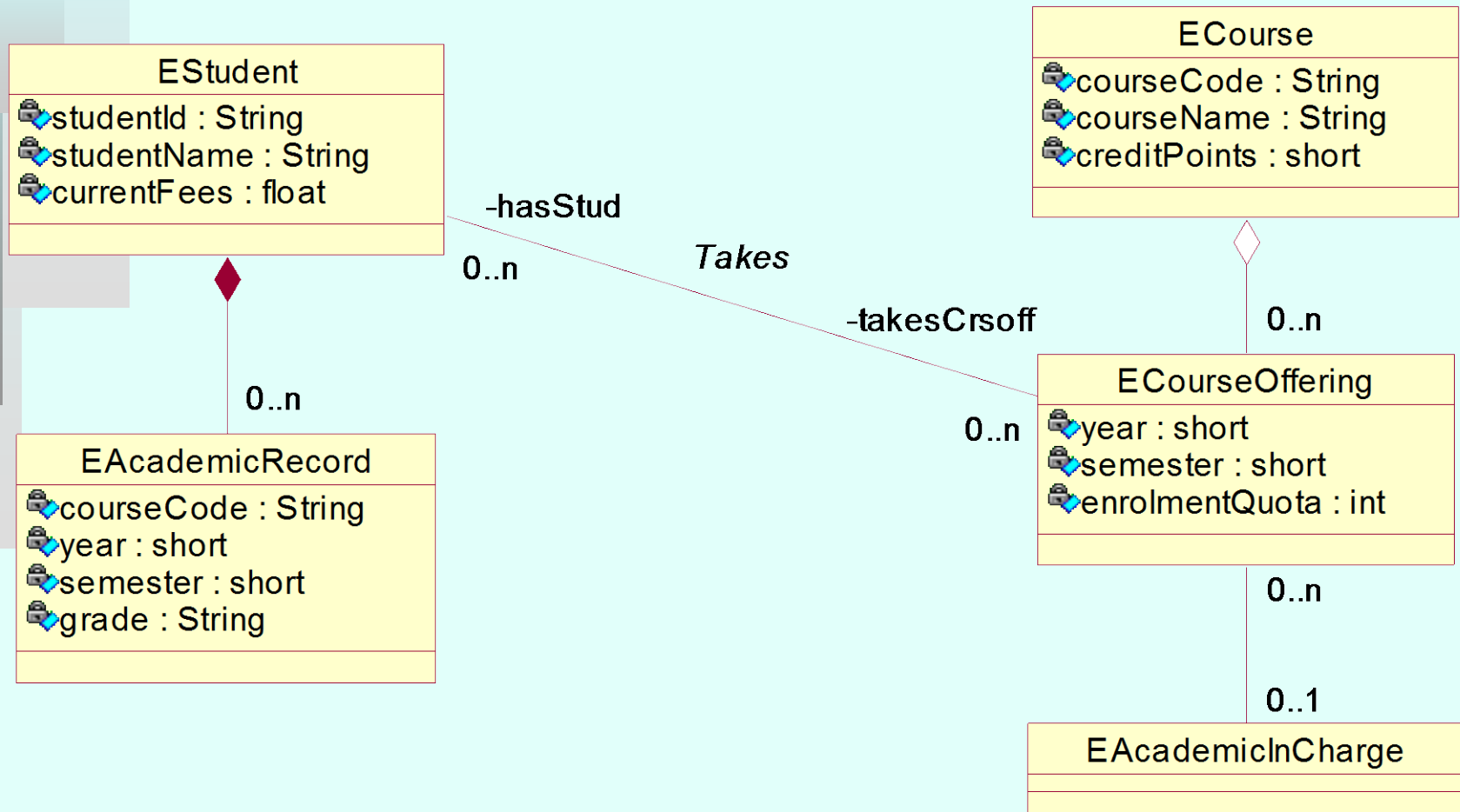
■ UML supports

- Aggregation
 - By-reference semantics
 - Hollow diamond
 - Corresponds to Has and Member aggregations
- Composition
 - By-value semantics
 - Solid diamond
 - Corresponds to ExclusiveOwns and Owns aggregations

Example 3.9 – University Enrolment

- Consider the following additional requirements:
 - The student's academic record to be available on demand.
 - The record to include information about the student's grades in each course that the student enrolled in (and has not withdrawn without penalty).
 - Each course has one academic in charge of a course, but additional academics may also teach in it.
 - There may be a different academic in charge of a course each semester.
 - There may be different academics for each course each semester.

Example 3.9 – University Enrolment (solution)



Modeling generalization

- Common features abstracted into a more generic class
- Subclasses **inherit** (reuse) superclass features
- **Substitutability** – subclass object is a legal value for a superclass variable
(e.g. a variable holding `Fruit` objects can have an `Apple` object as its value)
- **Polymorphism** – the same operation can have different implementations in different classes
- **Abstract operation** – implementation provided in subclasses
- **Abstract class** – class with no direct instance objects
 - A class with an abstract operation is abstract

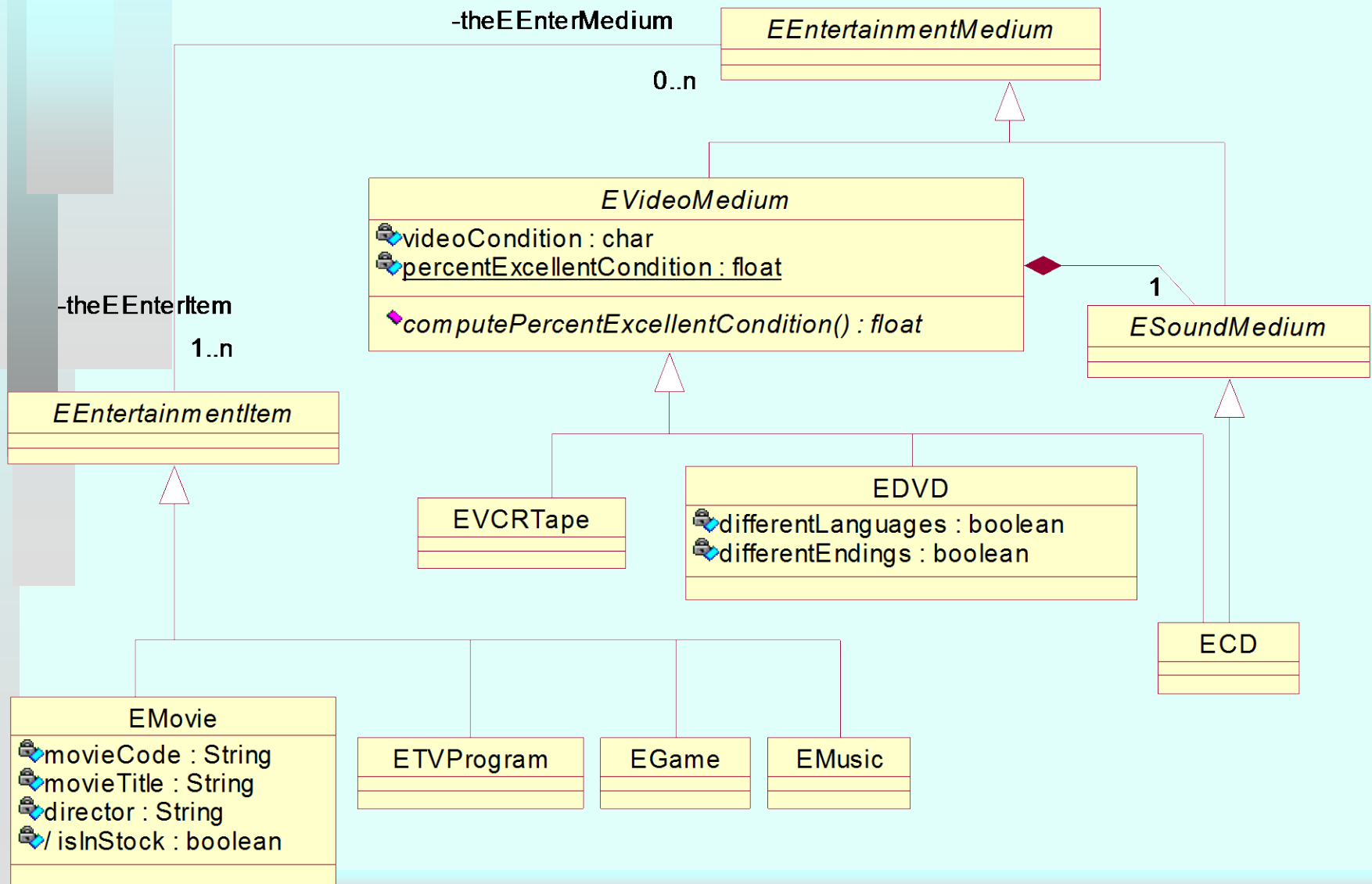
Discovering and specifying generalization

- Some discovered in parallel with discovery of associations
- The litmus test phrases
 - “can-be”
 - “is-a-kind-of”
- Multiple inheritance possible
- Solid line with an arrowhead pointing to the superclass

Example 3.10 – Video Store

- The classes identified so far imply a generalization hierarchy rooted at the class `EntertainmentMedium`
- They also imply a parallel generalization hierarchy rooted at a class that can be named `EEntertainmentItem` or `EEntertainmentItemCategory`
- To capture some state differences between classes in the generalization hierarchy originating from the class `EEntertainmentMedium`, assume that the storage capacity of an EDVD allows multiple versions of the same movie to be held, each in a different language or with different endings.
- There is no need to decipher the peculiarities of game and music CDs in the model.

Example 3.10 – Video Store (solution)



Modeling interfaces

- Interfaces
 - do not have attributes (except constants), associations or states
 - they only have operations, but all operations are implicitly public and abstract
 - operations are declared (i.e. turned into implemented methods) in classes which implement these interfaces
- Interfaces do not have associations to classes but they may be targets of one-way associations from classes
 - this happens when an attribute that implements an association is typed with an interface, rather than with a class
 - the value of any such attribute will be a reference to some class that implements the interface
- An interface may have a generalization relationship to another interface
 - this means that an interface can extend another interface by inheriting its operations

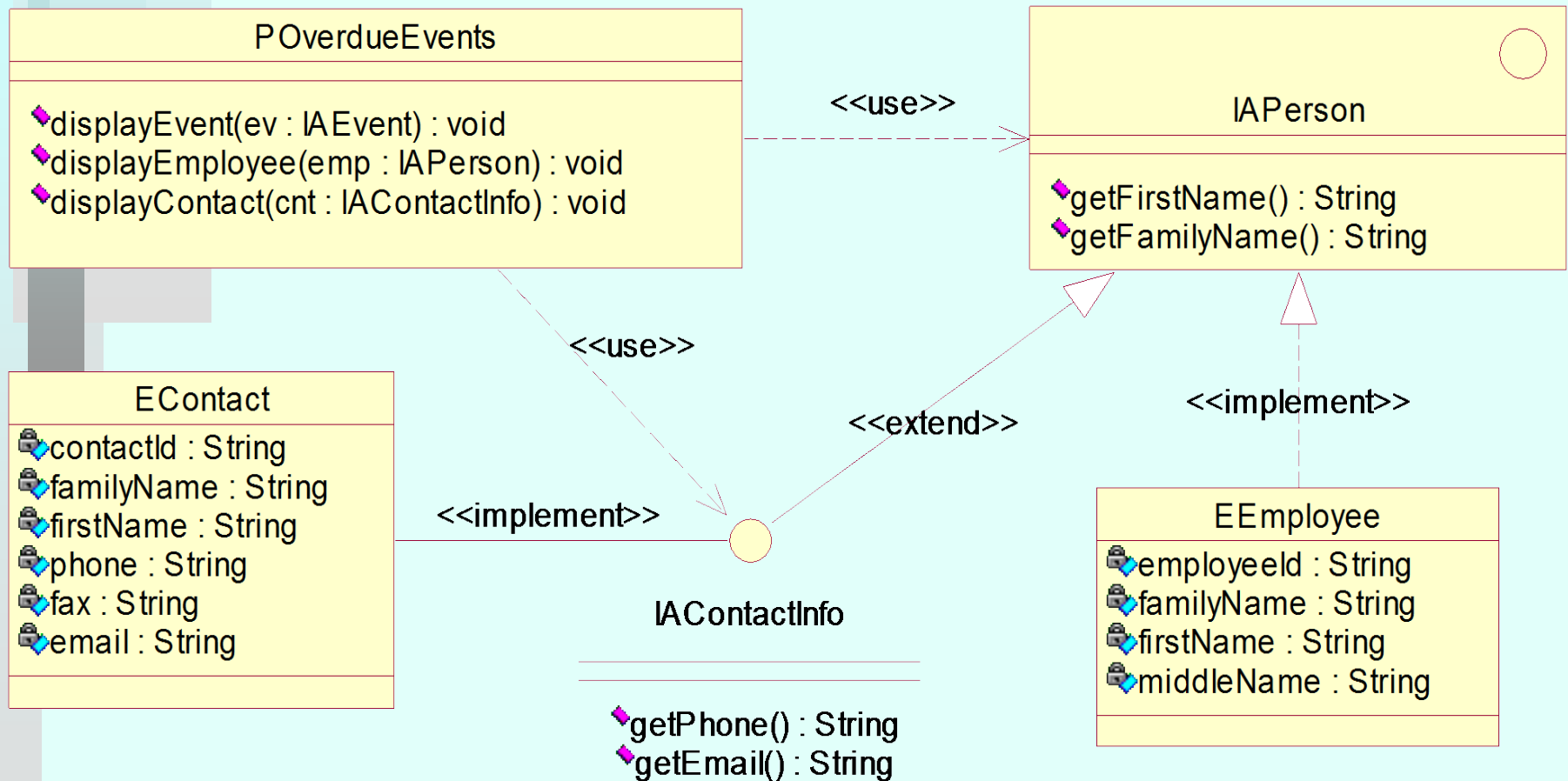
Discovering and specifying interfaces

- Interfaces are not discovered from the analysis of the application domain
- They are discovered based on design considerations
 - fundamental for enforcing architectural frameworks, such as the PCBMER framework
 - interface reveals only a limited portion of the behavior of an actual class
- Class that uses (requires) the interface can be indicated by a dashed arrow pointing to the interface
 - the arrow can be stereotyped with the keyword «use»
- Class that implements (realizes) the interface is indicated by a dashed lined with a triangular end
 - the line can be stereotyped with the keyword «implement»

Example 3-11 – Contact Management

- Consider classes `EContact` and `EEmployee`
 - some attributes in common (`firstName`, `familyName`)
 - operations that provide access to these attributes can be extracted into a single interface.
- There is a need to display information about overdue events to the screen
 - presentation-layer class has the responsibility to display a list of overdue events together with names of contacts and employees, as well as with the additional contact details (phone and email) to contacts
- Propose a model such that the presentation class uses one or more interfaces implemented by `EEmployee` and `EContact` to support part of the “display overdue events” functionality

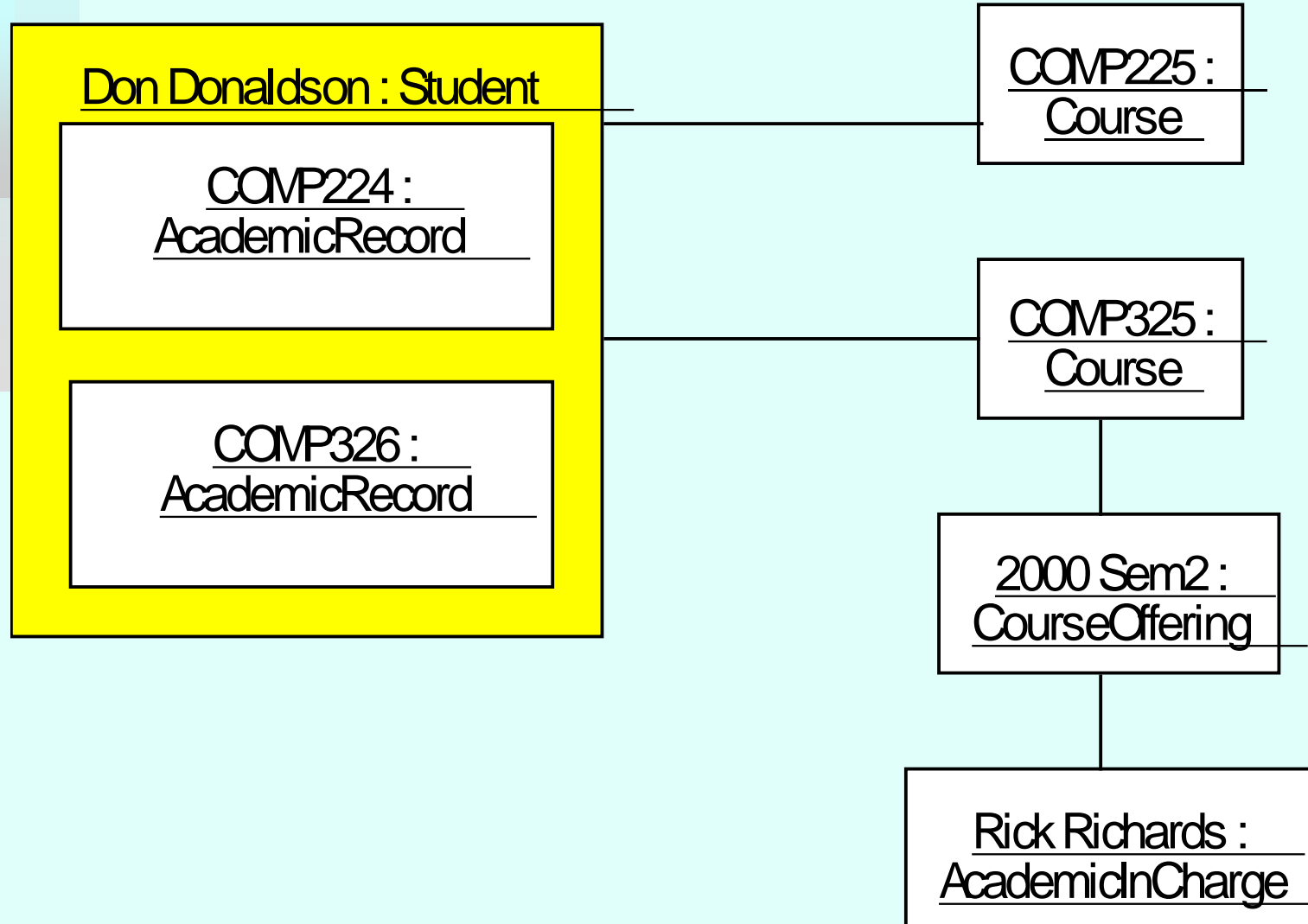
Example 3-11 – Contact Management



Modeling and specifying objects

- Only to exemplify
 - To illustrate complex relationships between objects
 - To demonstrate changes to objects over time
 - To illustrate object collaboration

Example 3.12 – University Enrolment



Review Quiz 3.2

1. What is the CRC approach used for?
2. What role names are used for?
3. What does transitivity mean in the context of aggregations?
4. A subclass object may be a legal value for a superclass variable. Associated with that observation is the principle of what?