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

Topics

- State specifications
- Behavior specifications
- State change specifications

3. Behavior specifications

- an *operational view* of the system
- *use cases* in the application domain and which *classes* are involved
- *interactions* reveal layers of classes other than entity classes

Behavior specification

- Depicted in use cases
- Determines which classes are involved in execution of use cases
    - Main class operations identified
    - Message passing between objects captured
    - Control classes and boundary classes considered
- Computations modeled in Activity Diagrams
- Interactions modeled in Sequence Diagrams or Communication Diagrams

Modeling use cases

- Complete piece of functionality
  - Main flow
  - Subflows
  - Alternate flows
- Piece of externally visible functionality
- Orthogonal piece of functionality
- Piece of functionality initiated by an actor
- Piece of functionality that delivers an identifiable value to an actor

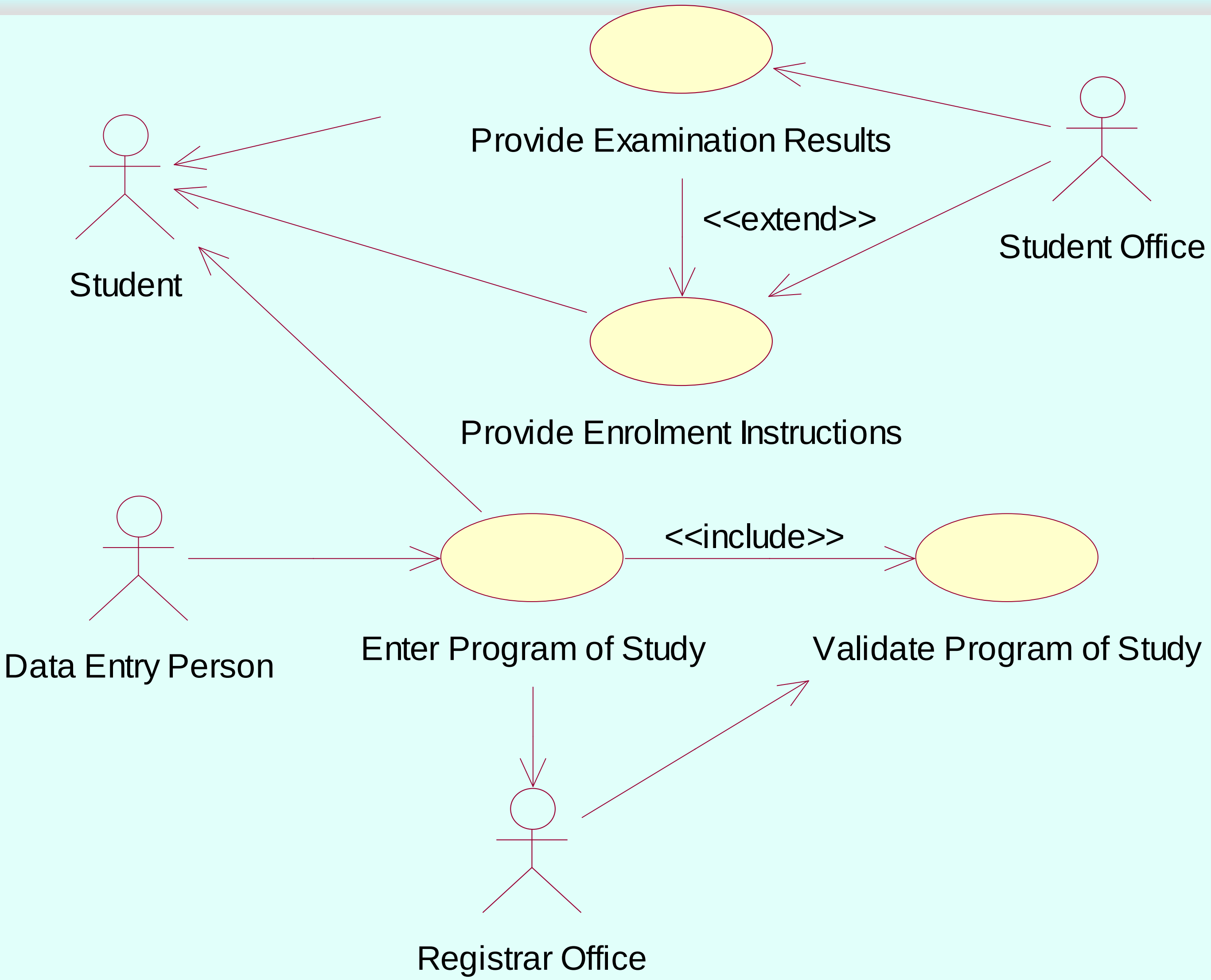
Discovering use cases

- Discovered from
  - Requirements identified in the Requirements Document
  - Actors and their purpose in the system
- Questions to ask
  - What are the main tasks performed by each actor?
  - Will an actor access or modify information in the system?
  - Will an actor inform the system about any changes in other systems?
  - Should an actor be informed about unexpected changes in the system?

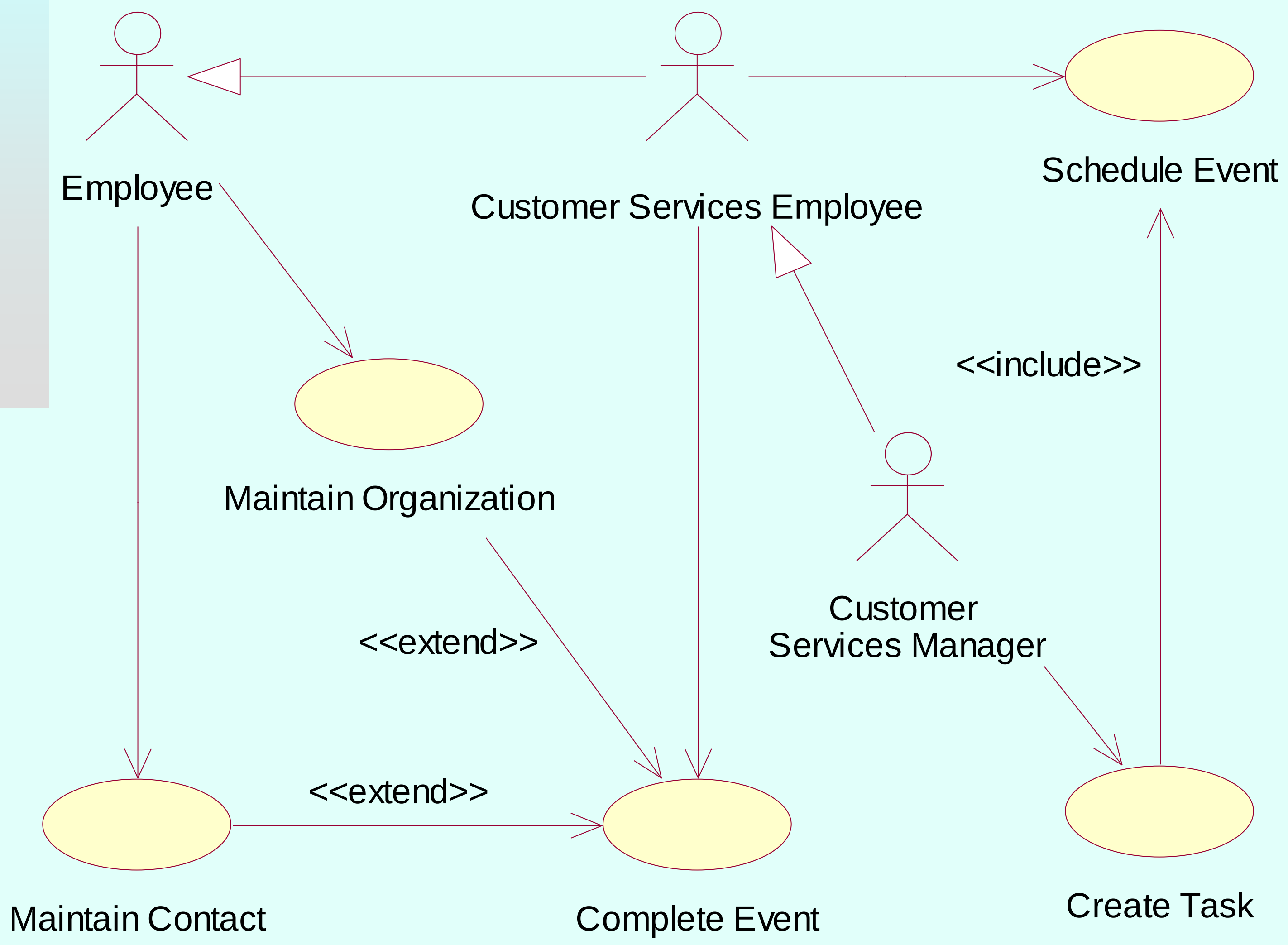
Specifying use cases

- Actors
- Use cases
- Four kinds of relationships
  - Association (between actor and use case)
  - Include (stereotyped with the word: «include»)
    - Included use case is always necessary for the completion of the activating use case
  - Extend (stereotyped with the word: «extend»)
    - Another use is activated occasionally at specific extension point
  - Generalization
- Relationships to be used with restrain

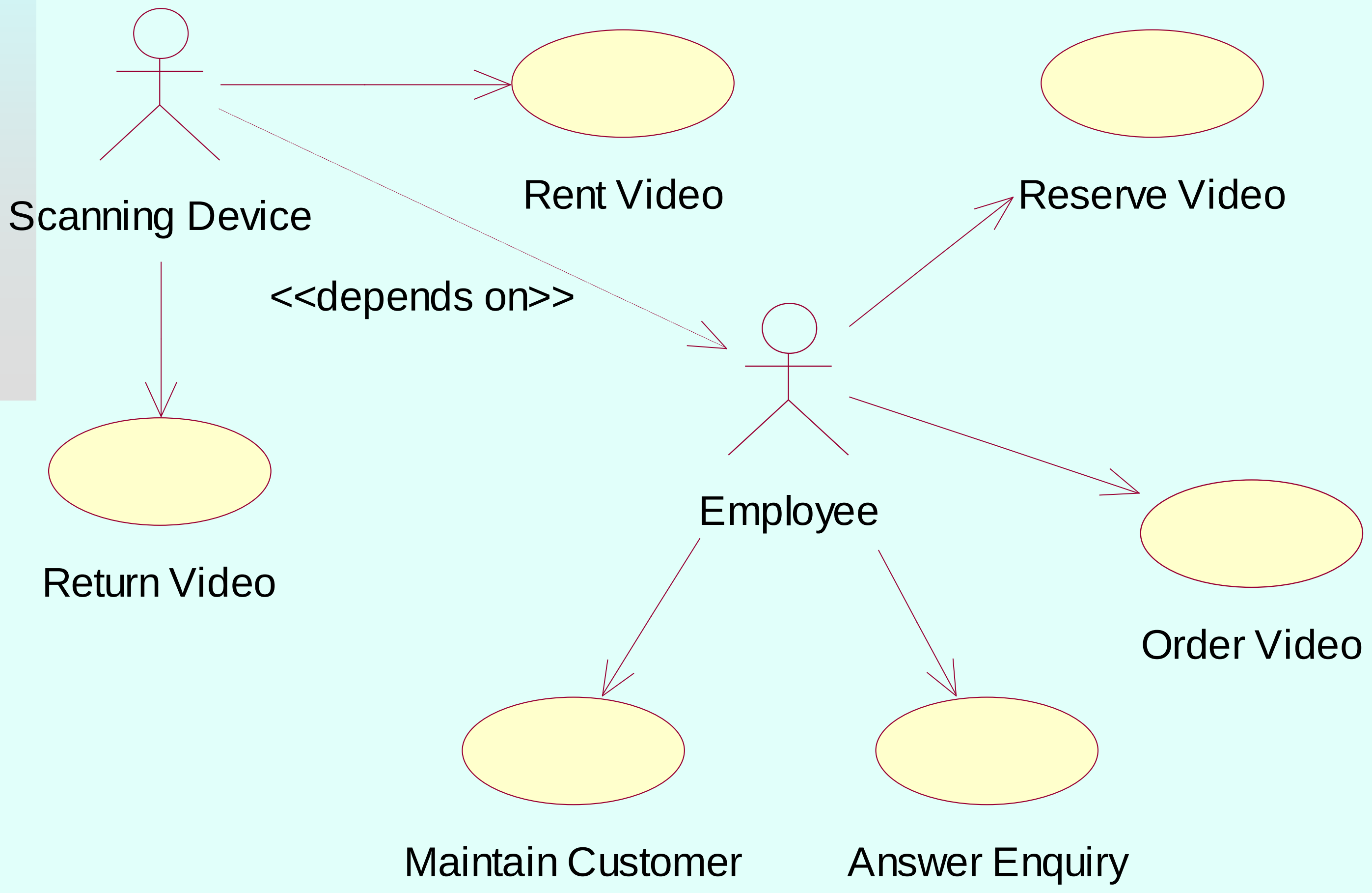
Example 3.13 – University Enrolment



Example 3.14 – Contact Management



Example 3.15 – Video Store



Example 4.15 – Video Store (Rent Video)

Brief Description	*A customer wishes to rent a video tape or disk that is picked from the store's shelves or that has been previously reserved by the customer. Provided that the customer has a non-delinquent account, the video is rented out once payment has been received. If the video is not returned in a timely fashion, an overdue notice is mailed to the customer.*
Actors	*Employee, Scanning Device*
Preconditions	*Video tape or disk is available to be hired. Customer has a membership card. Scanner devices work correctly.* *Employee at the front desk knows how to use the system.*

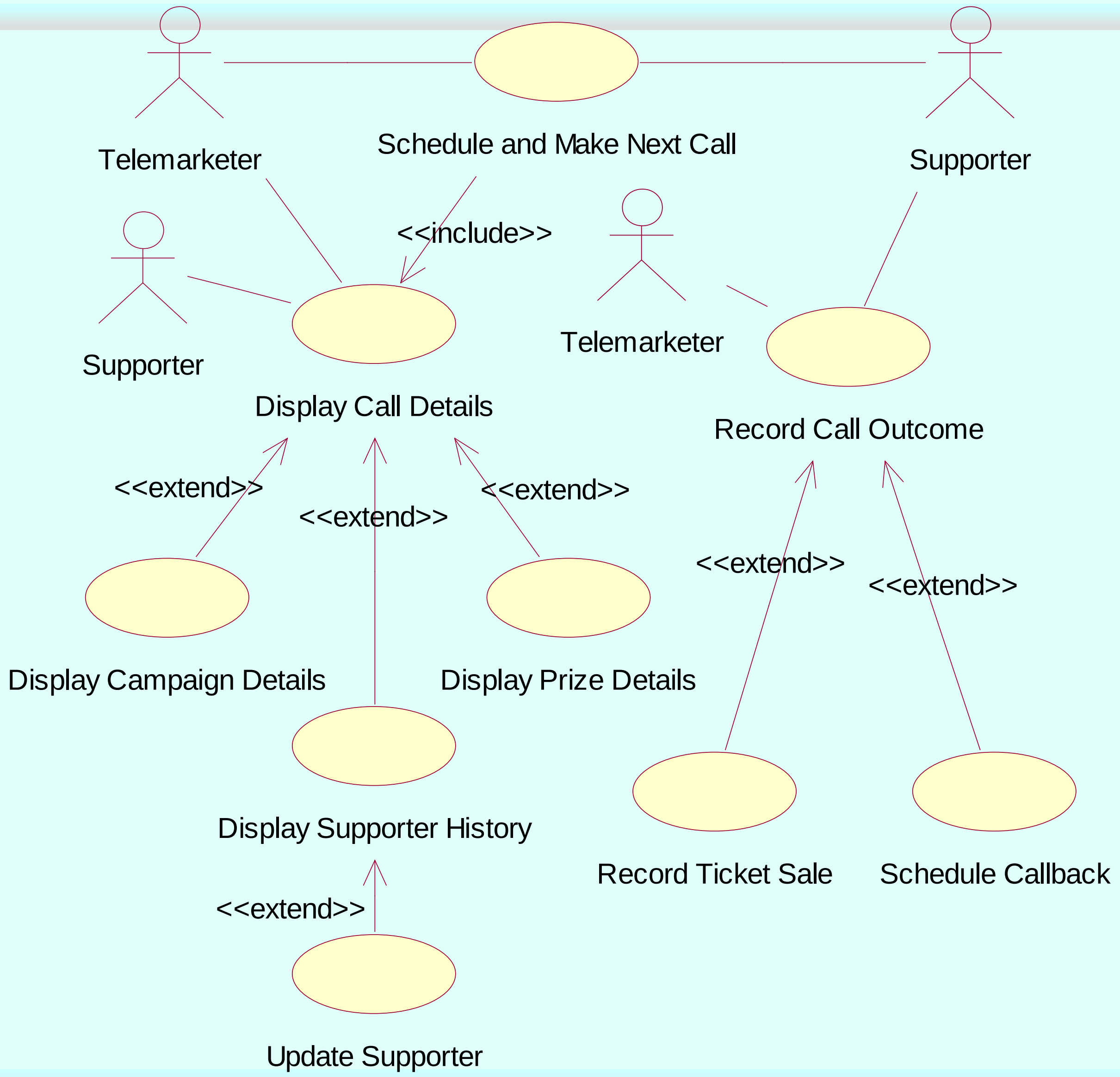
Example 3.15 – Video Store (Rent Video)

Main Flow	*A customer may ask an employee about video availability (including a reserved video) or may pick a tape or disk from the shelves. The video and the membership card are scanned and any delinquent or overdue details are brought up for the employee to query the customer about. If the customer does not have a delinquent rating, then he/she can hire up to a maximum of eight videos. However, if the rating of the customer is “unreliable,” then a deposit of one rental period for each tape or disk is requested. Once the amount payable has been received, the stock is updated and the tapes and disks are handed to the customer together with the rental receipt. The customer pays by cash, credit card or electronic transfer. Each rental record stores (under the customer’s account) the check-out and due-in dates together with the identification of the employee. A separate rental record is created for each video hired.* The use case will generate an overdue notice to the customer if the video has not been returned within two days of the due date, and a second notice after another two days (and at that time the customer is noted as “delinquent”).

Example 3.15– Video Store (Rent Video)

Alternative Flows	*A customer does not have a membership card. In this case, the 'Maintain Customer' use case may be activated to issue a new card.* *An attempt to rent too many videos.* *No videos can be rented because of the customer's delinquent rating.* *The video medium or membership card cannot be scanned because of damage to them.* *The electronic transfer or credit card payment is refused*.
Postconditions	*Videos are rented out and the database is updated accordingly.*

Example 3.16 – Telemarketing (solution)



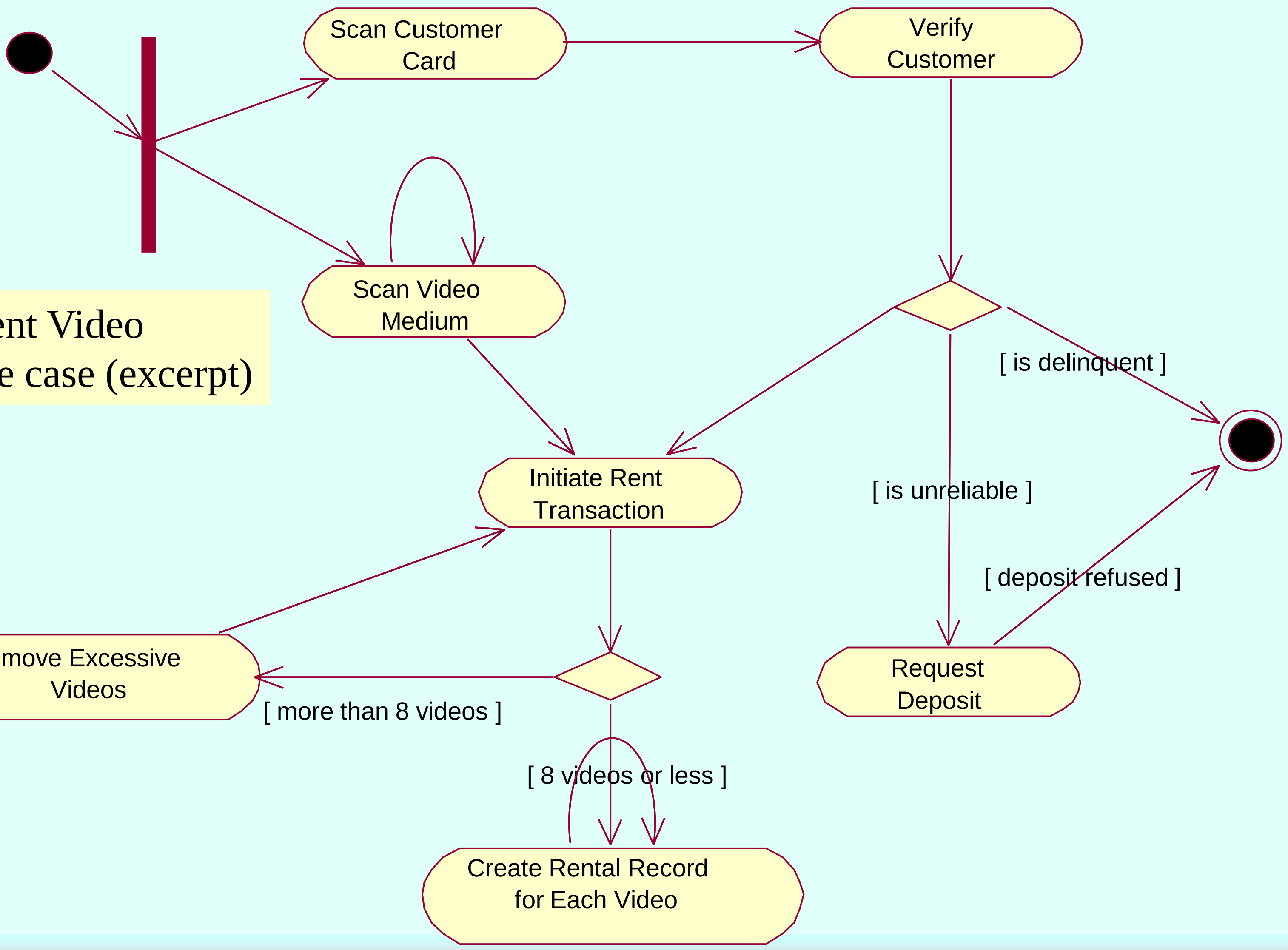
Modeling activities

- Activity Diagrams
- Flow of logic
  - Sequential control
  - Concurrent control
- Can be used at different levels of abstraction
  - To define execution of a use case
  - To define execution of an operation

Discovering and specifying actions

- The execution proceeds from one **action** to the next
- An **action** completes when its computation is completed
- **Actions** can be discovered from the narrative specifications of **use cases**
- Actions are connected by **transition lines**
- **Synchronization** bars (fork and re-join)
- **Branch** diamonds (branch and merge)
- **External events** not normally modeled on activity graphs

Example 3.17 – Video Store (solution)



Rent Video
use case (excerpt)

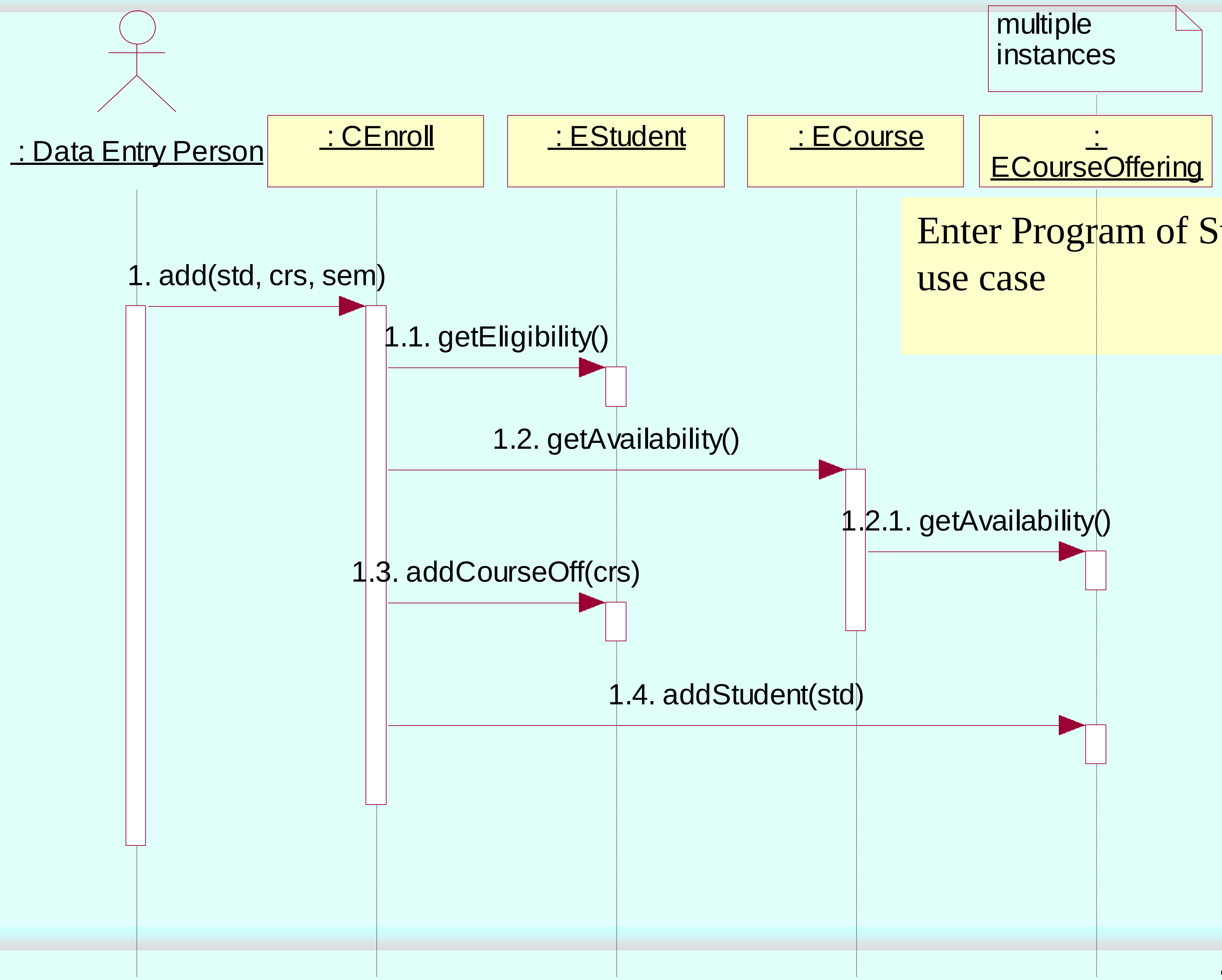
Modeling interactions

- Sequence Diagrams
  - Show an exchange of messages between objects arranged in a time sequence
  - More useful in analysis
- Communication Diagrams
  - Emphasize the relationships between objects along which the messages are exchanged
  - More useful in design
- Can be used to determine operations in classes

Message sequences

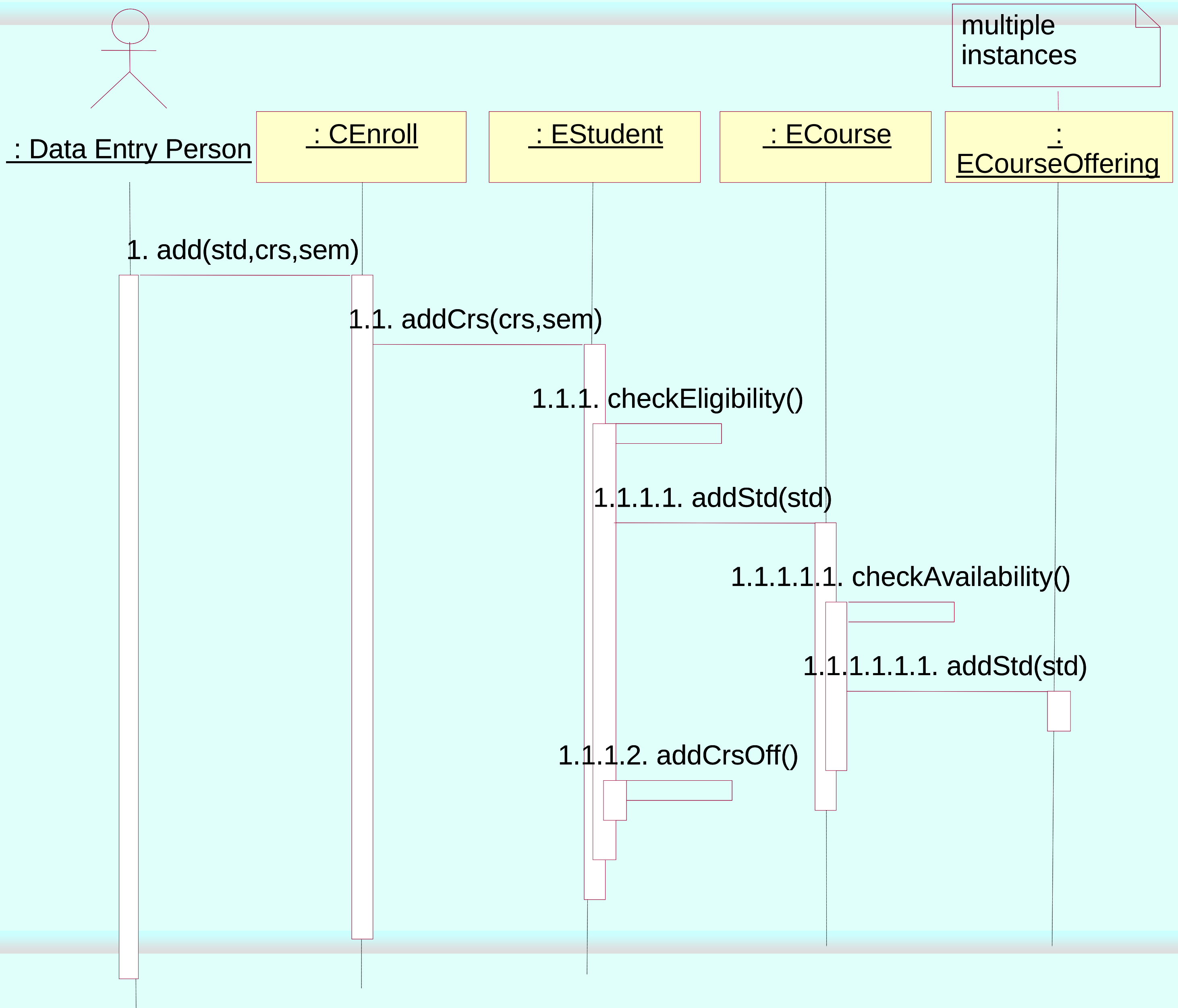
- Actions in Activity Diagrams are mapped to messages to Sequence Diagrams
- Message can be a:
  - Signal
    - Denotes asynchronous inter-object communication
    - The sender continues executing after sending the signal message
    - Implies *event processing*
  - Call
    - Denotes synchronous invocation of an operation
    - The return message can return some values to the caller or it can just acknowledge that the operation completed
    - Implies *message passing*

Example 3.18 – UE (centralized interaction)



Enter Program of Study use case

Example 3.18 – UE (distributed interaction)



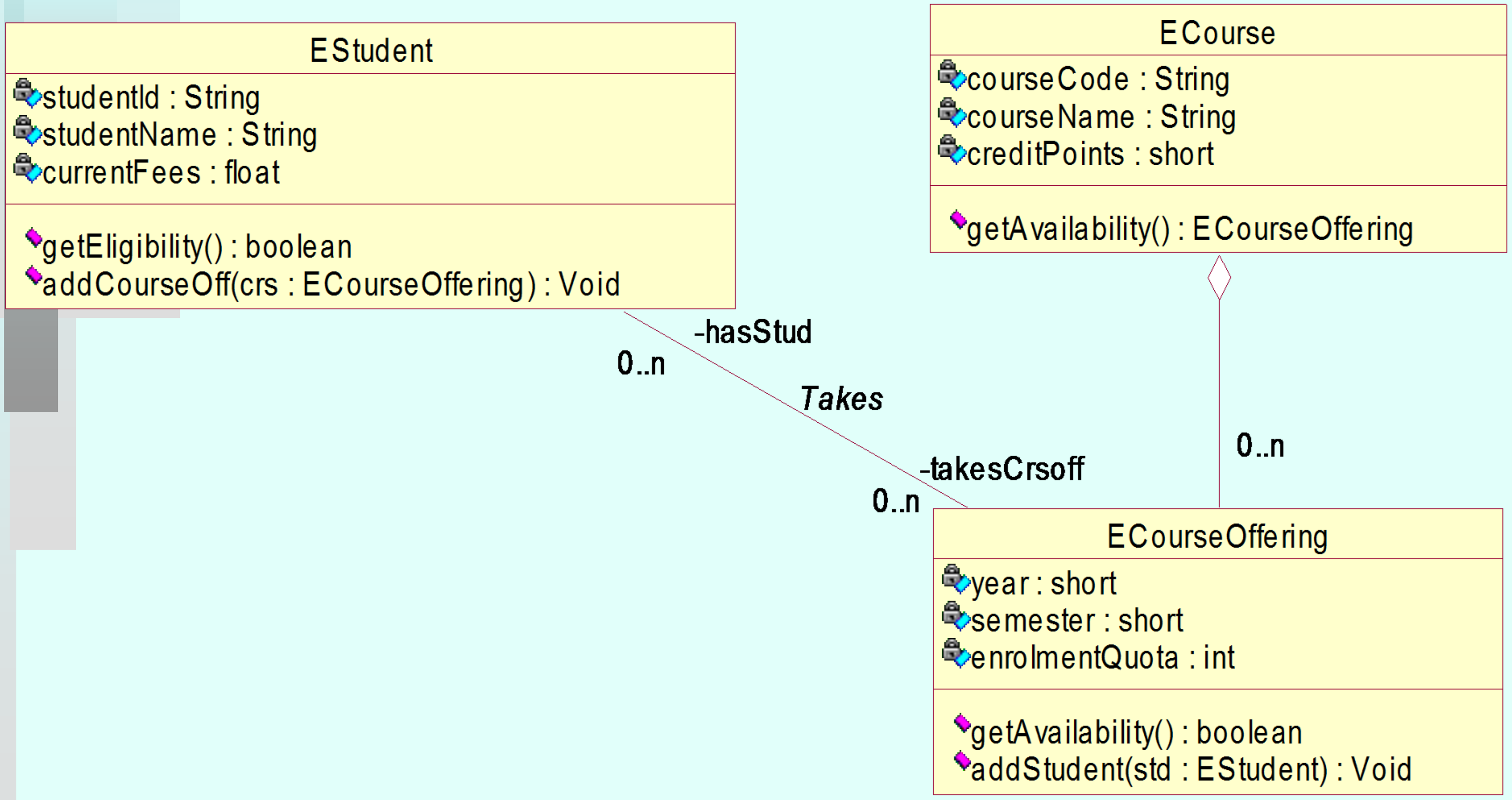
Modeling public interfaces

- Determined by the set of operations that the class offers as its service
- In analysis
  - Signature of each operation is defined
    - Operation name
    - List of formal arguments
    - Return type
- In design
  - Algorithm of a method that implements the operation is defined
- Operation can have
  - Instance scope
  - Class (static) scope ($ in front of operation name)

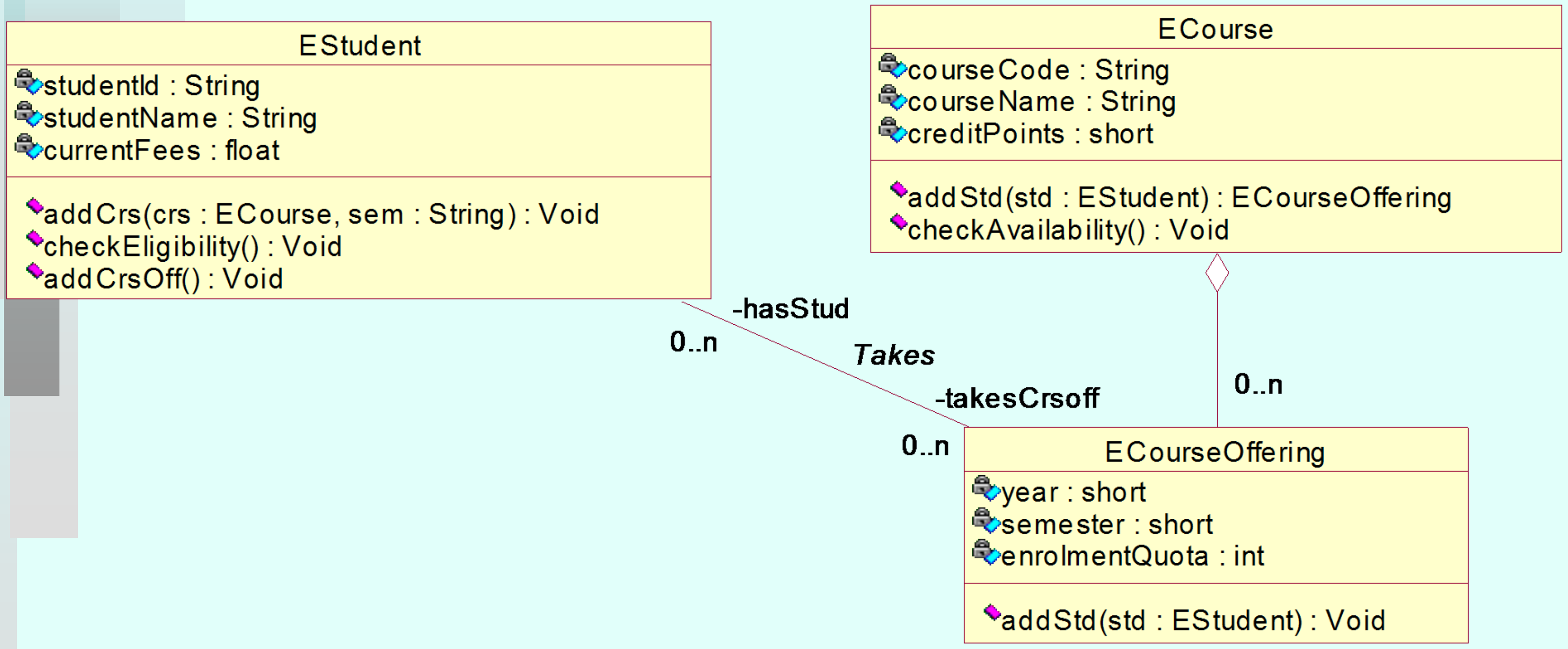
Discovering class operations

- From Sequence Diagrams
  - Message to an object must be serviced by an operation in that object
- From expected object responsibilities, including the CRUD operations
  - Create – a new object instance
  - Read –the state of an object
  - Update – the state of an object
  - Delete – i.e. destroy itself

Example 3.19 – UE (centralized interaction)



Example 3.19 – UE (distributed interaction)



Review Quiz 3.3

1. Do use cases have the possibility of representing the concurrent flow of control?
2. Can activity diagrams be constructed before a class model is developed?
3. A message can denote an asynchronous inter-object communication. How is such a message called?
4. What UML modeling technique is most useful for discovering class operations?