Software Engineering & Information System Design

Course Code: CSE-3319

Lecture 5

Depicting Systems Graphically

- Context-Level Data Flow Diagrams
- Data Flow Diagrams (DFD)
- Entity-Relationship (E-R) Model
- Use Case Modeling

Entity-Relationship Model

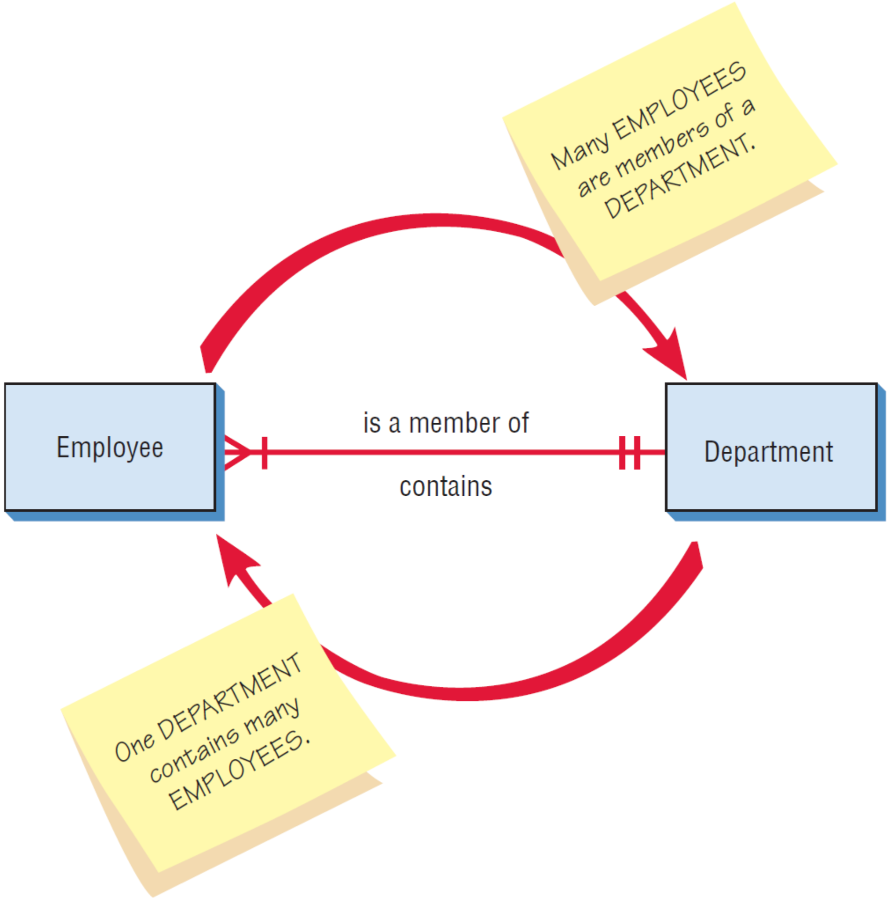
- Focus is on the entities and their relationships within the organizational system.
- An entity may be a person, a place, thing, or an event.
- A relationship is the association that describes the interaction among the entities.

Relationships

- Relationships show how the entities are connected.
- Three types of relationships:
  - One-to-one
  - One-to-many
  - Many-to-many

- One-to-one – one employee is assigned to one phone extension.
- One-to-many – many employees are assigned to a department.
- Many-to-many – many passengers fly to many destinations.

Entity-Relationship Example



Many EMPLOYEES are members of a DEPARTMENT.

Employee

is a member of

contains

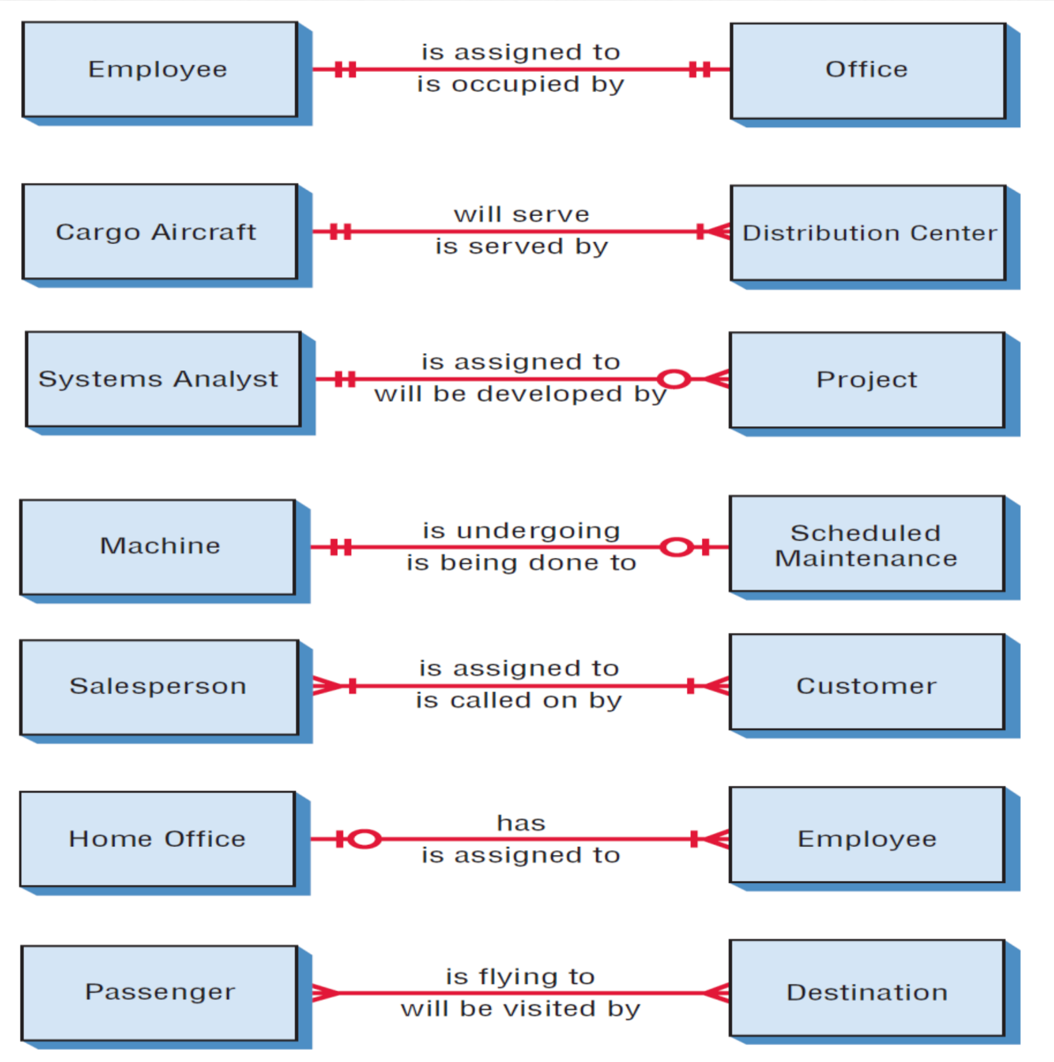
Department

One DEPARTMENT contains many EMPLOYEES.

FIGURE 2.7

An entity-relationship diagram showing a many-to-one relationship.

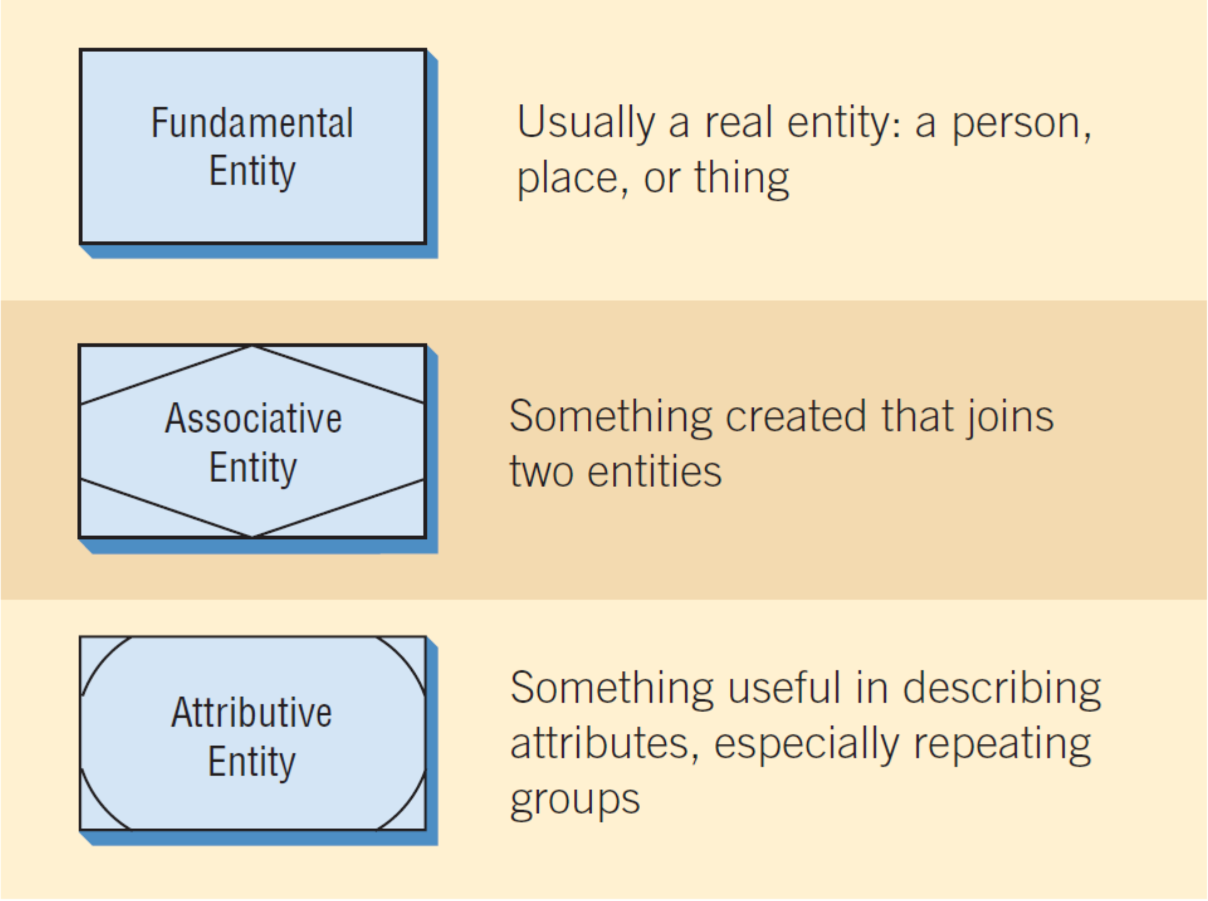
Figure 2.8 Examples of different types of relationships in E-R diagrams



Entities

- Fundamental entity
- Associative entity
- Attributive entity

Figure 2.9 Three different types of entities used in E-R diagrams

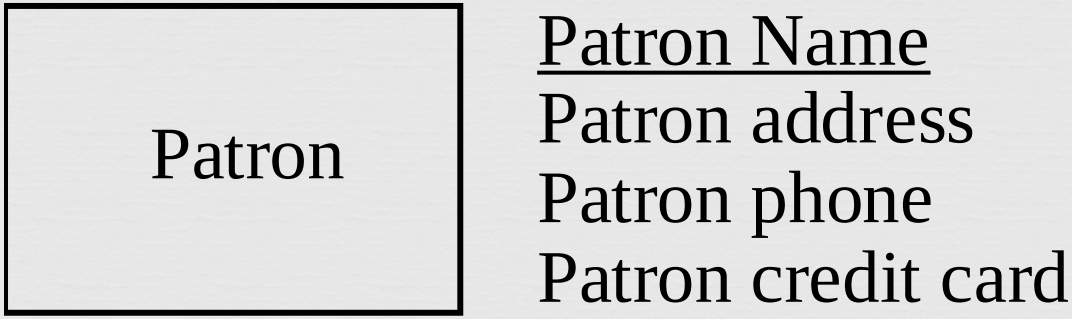


Fundamental Entity	Usually a real entity: a person, place, or thing
Associative Entity	Something created that joins two entities
Attributive Entity	Something useful in describing attributes, especially repeating groups

- Different symbols are used to represent different types of entities.
- An associative entity can only exist if it is connected to at least two other entities.
- An attributive entity is used when we want to show data that are completely dependent on the existence of an fundamental entity.

Attributes

- Data attributes are what make up or define the entity.
- Data attributes may be added to the diagram.



Patron

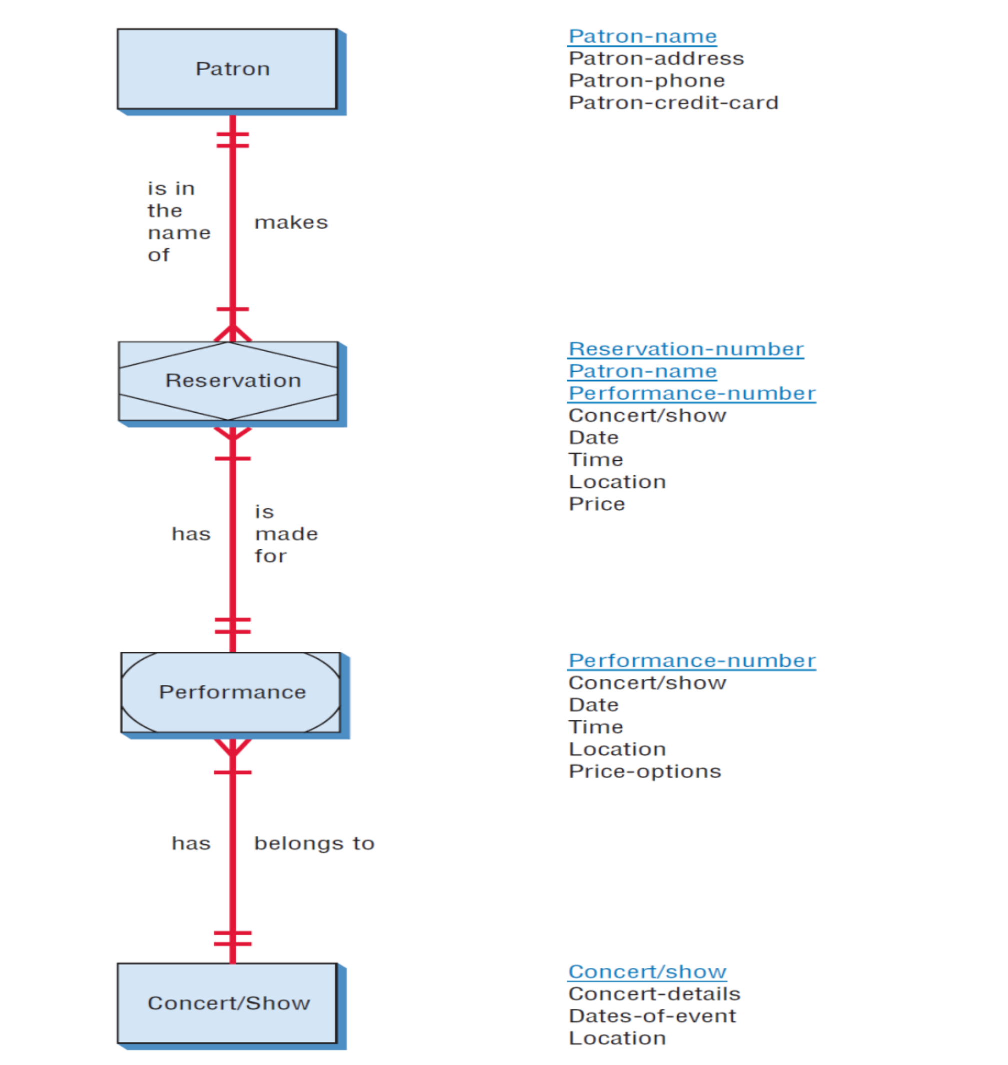
Patron Name
Patron address
Patron phone
Patron credit card

Creating Entity-Relationship Diagrams

- List the entities in the organization.
- Choose key entities to narrow the scope of the problem.
- Identify what the primary entity should be.
- Confirm the results of the above through data gathering.

- ER diagrams are generally used to model the database.
- ER diagrams help the analyst understand what business the organization is actually in, determine the size of the problem, and discern whether the right problem is being addressed. The E-R diagram needs to be confirmed or revised as the data-gathering process takes place.

Figure 2-12 A more complete E-R diagram showing data attributes of the entities



Use Case Modeling

- Describes what a system does without describing how the system does it; that is, it is a logical model of the system.

Use Case Diagram

- Actor
  - Refers to a particular role of a user of the system.
  - Similar to external entities; they exist outside of the system.
- Use case symbols
  - An oval indicating the task of the use case.
- Connecting lines
  - Arrows and lines used to diagram behavioral relationships.

Actor

- Divided into two groups:
  - Primary actors
    - Supply data or receive information from the system.
    - Provide details on what the use case should do.
  - Supporting actors
    - Help to keep the system running or provide help.
    - The people who run the help desk, the analysts, programmers, and so on.

A Use Case Always Provides Three Things

- An actor that initiates an event.
- The event that triggers a use case.
- The use case that performs the actions triggered by the event.

Use Case Relations

- Behavioral relationships:
  - Communicates
    - Used to connect an actor to a use case.
  - Includes
    - Describes the situation in which a use case contains behavior that is common to more than one use case.

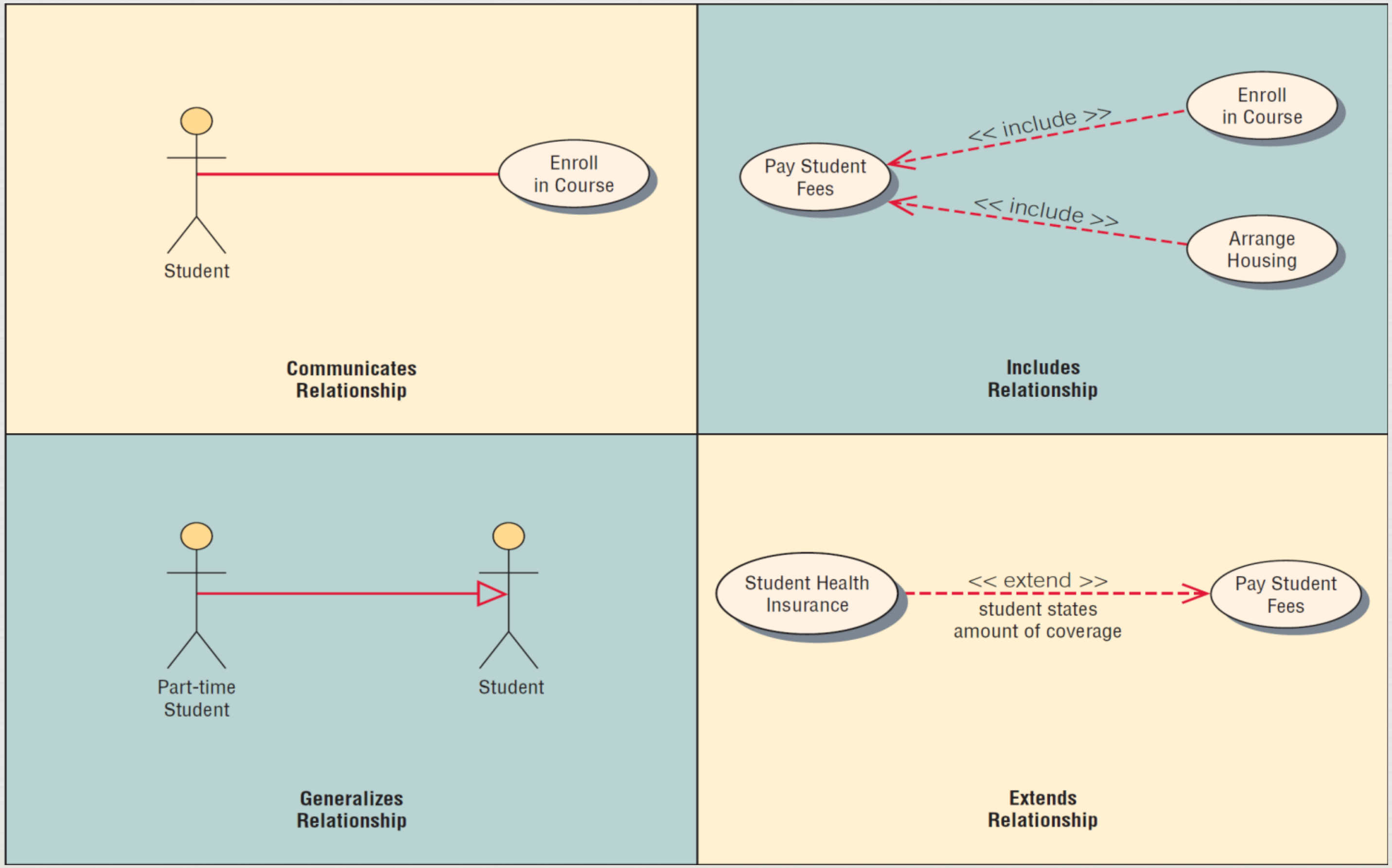
Use Case Relations

- Behavioral relationships (Continued):
  - Extends
    - Describes the situation in which one use case possesses the behavior that allows the new case to handle a variation or exception from the basic use case.
  - Generalizes
    - Implies that one thing is more typical than the other thing.

Figure 2.13 Some components of use case diagrams showing actors, use cases, and relationships for a student enrollment example

Relationship	Symbol	Meaning
Communicates	———————	An actor is connected to a use case using a line with no arrowheads.
Includes	<< include >> ←- - - - - - - - - -	A use case contains a behavior that is common to more than one other use case. The arrow points to the common use case.
Extends	<< extend >> - - - - - - - - - -→	A different use case handles exceptions from the basic use case. The arrow points from the extended to the basic use case.
Generalizes	———————▷	One UML "thing" is more general than another "thing." The arrow points to the general "thing."

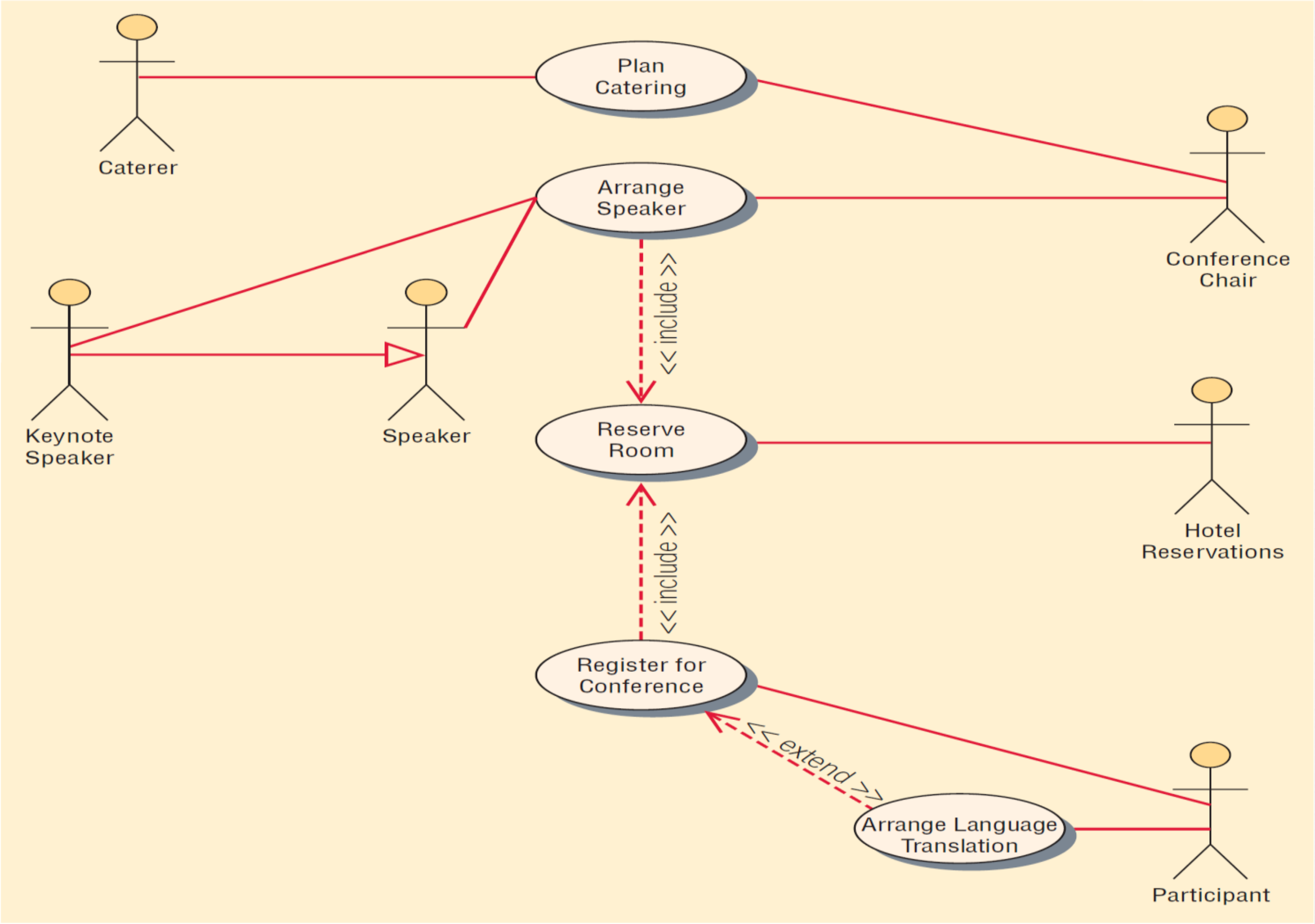
Figure 2.14 Examples of use cases and behavioral relationships for student enrollment



Developing Use Case Diagrams

- Review the business specifications and identify the actors involved.
- Identify the high-level events and develop the primary use cases that describe those events and how the actors initiate them.
- Review each primary use case to determine the possible variations of flow through the use case.
- The context-level data flow diagram could act as a starting point for creating a use case.

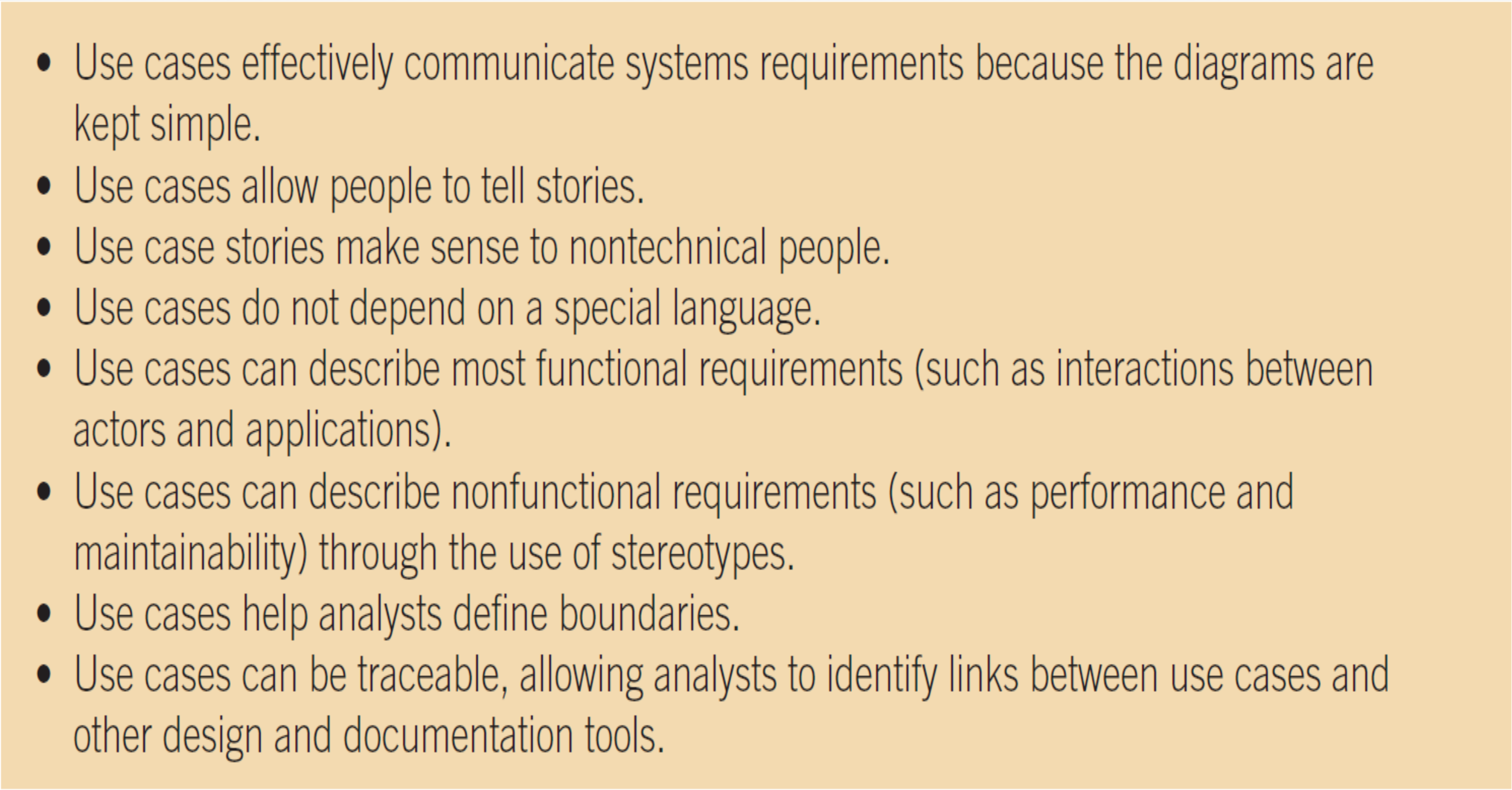
Figure 2.15 A use case diagram representing a system used to plan a conference



Why Use Case Diagrams Are Helpful?

- Identify all the actors in the problem domain.
- Actions that need to be completed are also clearly shown on the use case diagram.
- The use case scenario is also worthwhile.
- Simplicity and lack of technical detail.

Figure 2.17 The main reasons for writing use cases are their effectiveness in communicating with users and their capturing of user stories



- Use cases effectively communicate systems requirements because the diagrams are kept simple.
- Use cases allow people to tell stories.
- Use case stories make sense to nontechnical people.
- Use cases do not depend on a special language.
- Use cases can describe most functional requirements (such as interactions between actors and applications).
- Use cases can describe nonfunctional requirements (such as performance and maintainability) through the use of stereotypes.
- Use cases help analysts define boundaries.
- Use cases can be traceable, allowing analysts to identify links between use cases and other design and documentation tools.