Systems Analysis and Design in a Changing World, Fourth Edition

CHAPTER 5

MODELING SYSTEM REQUIREMENTS

Learning Objectives

- Explain the many reasons for creating information system models
- Describe three types of models and list some specific models used for analysis and design
- Explain how events can be used to define activities and use cases
- Identify and analyze events to which a system responds

Learning Objectives (continued)

- Explain how the concept of “things” in the problem domain also defines requirements
- Explain the similarities and the differences between data entities and objects
- Identify and analyze data entities and domain classes needed in the system
- Read, interpret, and create an entity-relationship diagram
- Read, interpret, and create a class diagram

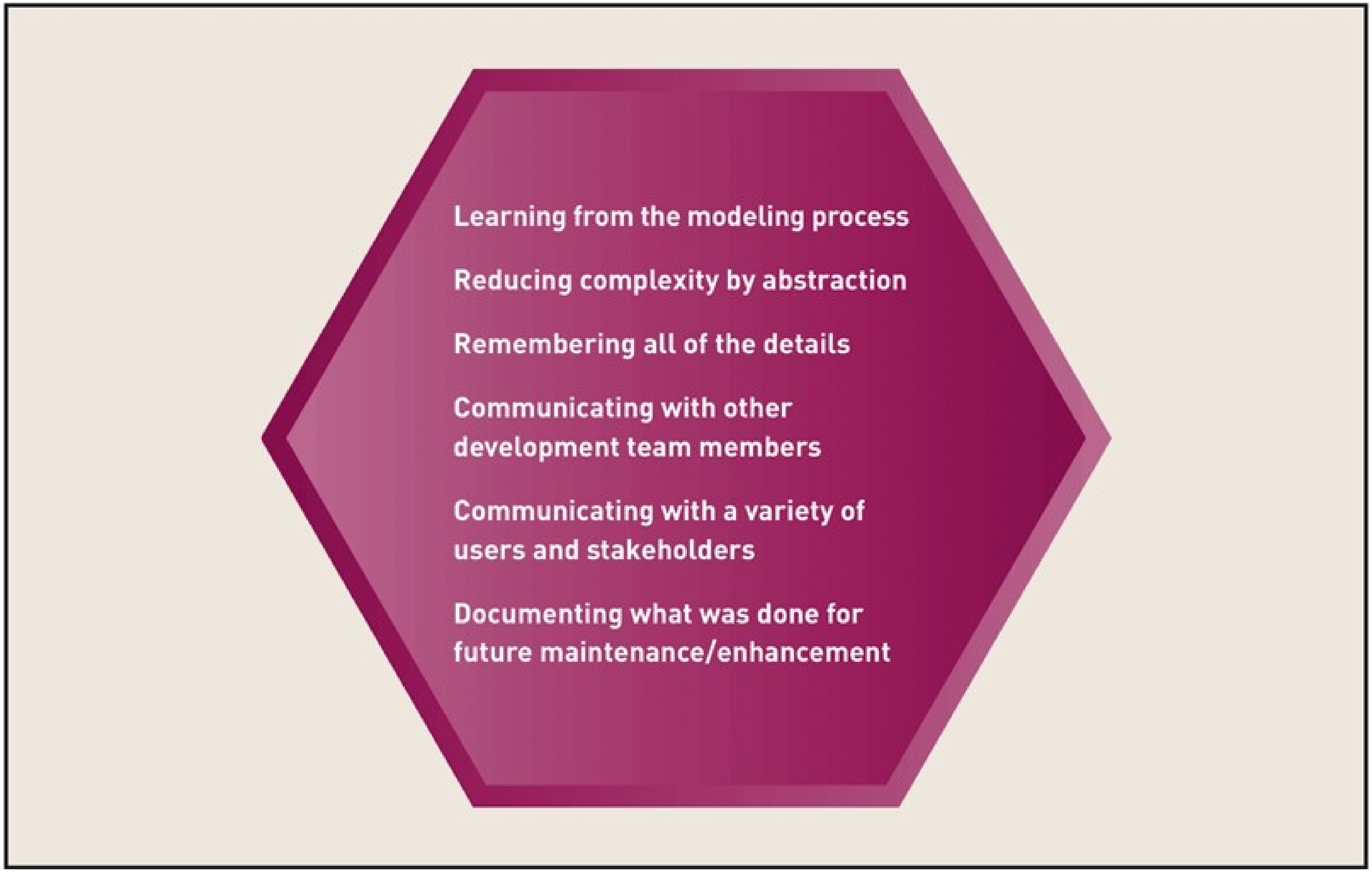
Overview

- Document functional requirements by creating models
- Models created during analysis phase activity – *Define system requirements*
- Two concepts help identify functional requirements in the traditional approach and object-oriented approach
  - Events that trigger use cases
  - Things in the users' work domain

Models and Modeling

- Analyst describes information system requirements using a collection of models
- Complex systems require more than one type of model
- Models represent some aspect of the system being built
- Process of creating models helps analyst clarify and refine design
- Models assist communication with system users

Reasons for Modeling (Figure 5-2)



Learning from the modeling process

Reducing complexity by abstraction

Remembering all of the details

Communicating with other development team members

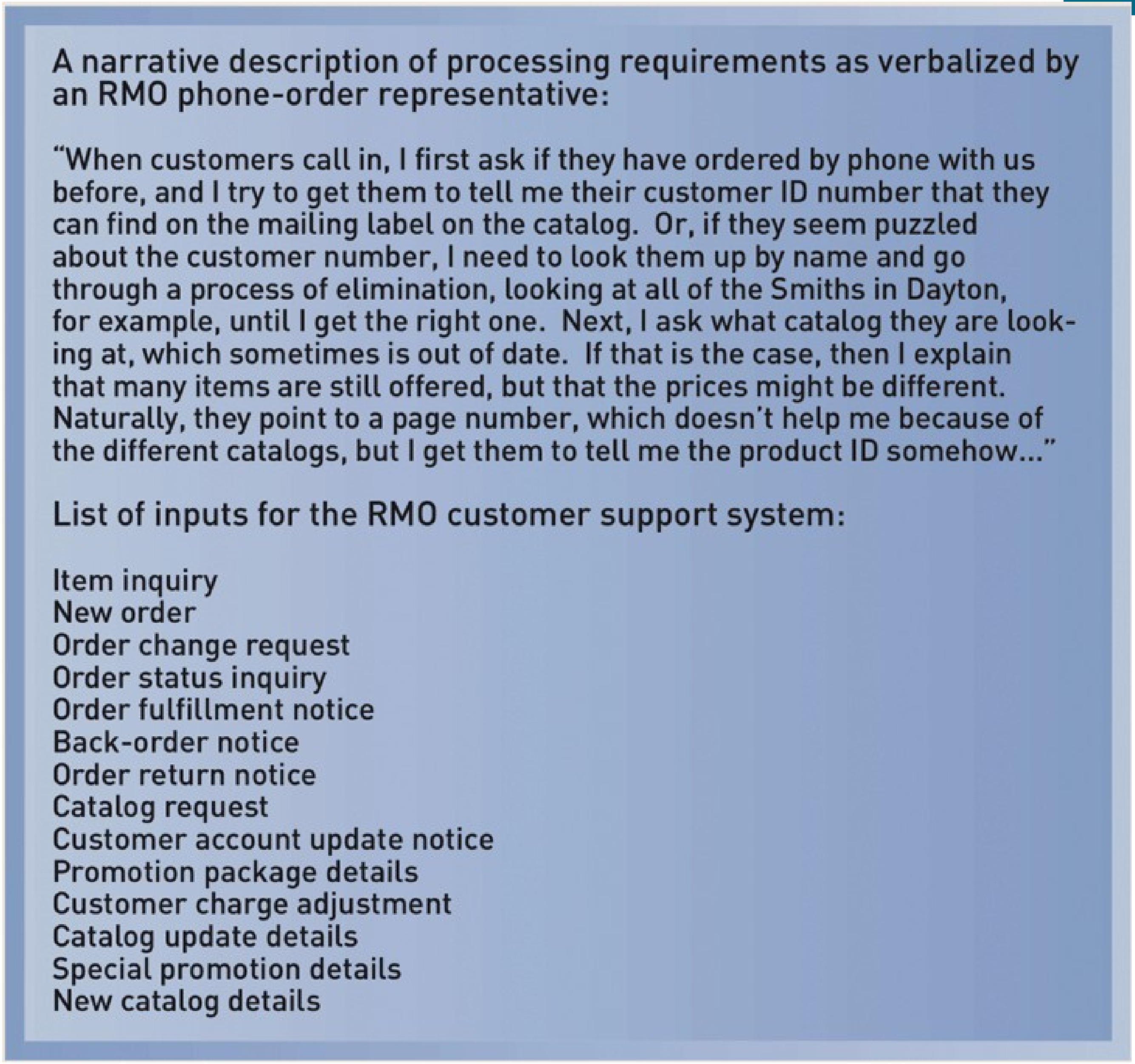
Communicating with a variety of users and stakeholders

Documenting what was done for future maintenance/enhancement

Types of Models

- Different types of models are used in information systems development
  - Mathematical – formulas that describe technical aspects of the system
  - Descriptive – narrative memos, reports, or lists that describe aspects of the system
  - Graphical – diagrams and schematic representations of some aspect of the system

Some Descriptive Models (Figure 5-3)



A narrative description of processing requirements as verbalized by an RMO phone-order representative:

"When customers call in, I first ask if they have ordered by phone with us before, and I try to get them to tell me their customer ID number that they can find on the mailing label on the catalog. Or, if they seem puzzled about the customer number, I need to look them up by name and go through a process of elimination, looking at all of the Smiths in Dayton, for example, until I get the right one. Next, I ask what catalog they are looking at, which sometimes is out of date. If that is the case, then I explain that many items are still offered, but that the prices might be different. Naturally, they point to a page number, which doesn't help me because of the different catalogs, but I get them to tell me the product ID somehow..."

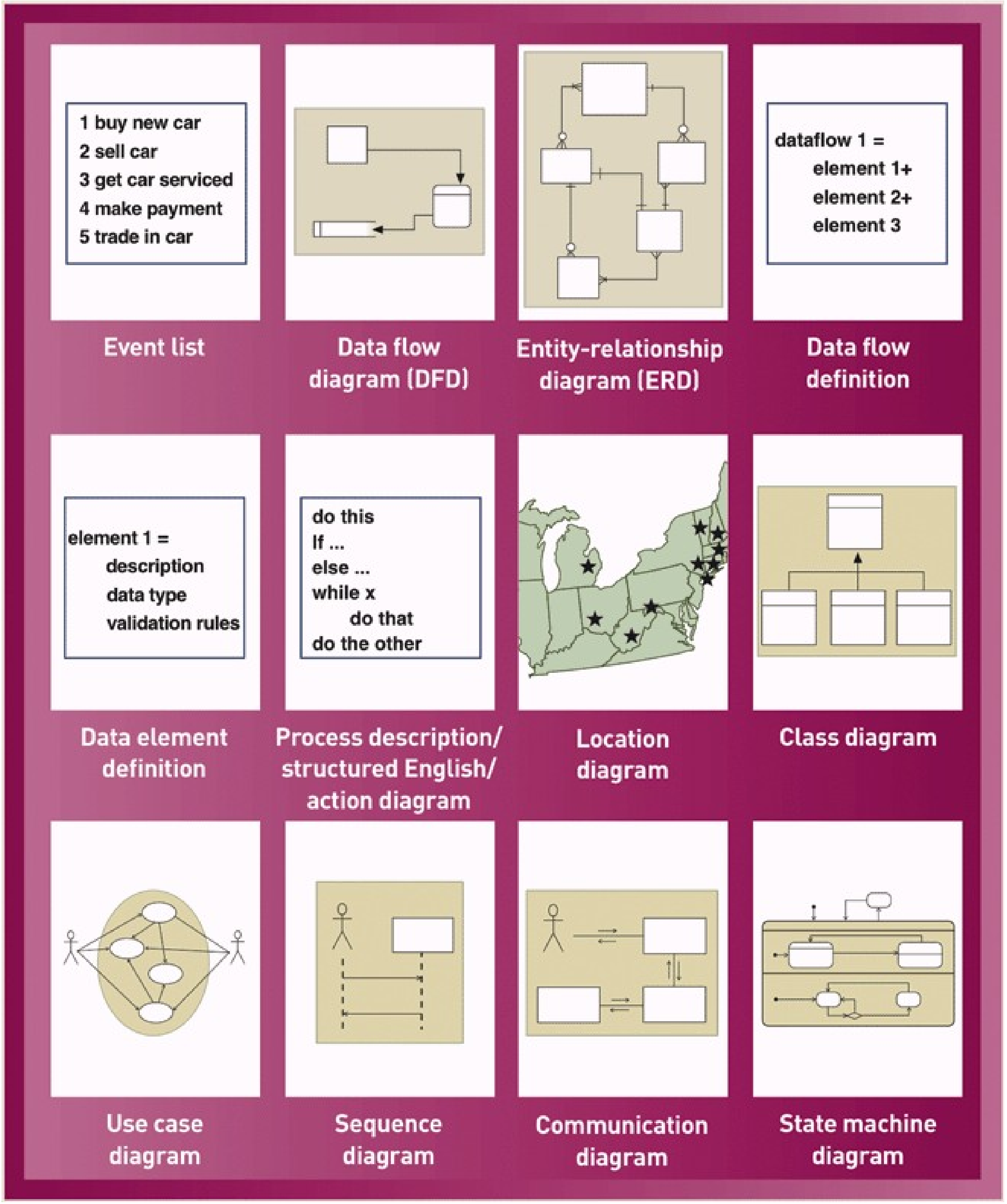
List of inputs for the RMO customer support system:

Item inquiry
New order
Order change request
Order status inquiry
Order fulfillment notice
Back-order notice
Order return notice
Catalog request
Customer account update notice
Promotion package details
Customer charge adjustment
Catalog update details
Special promotion details
New catalog details

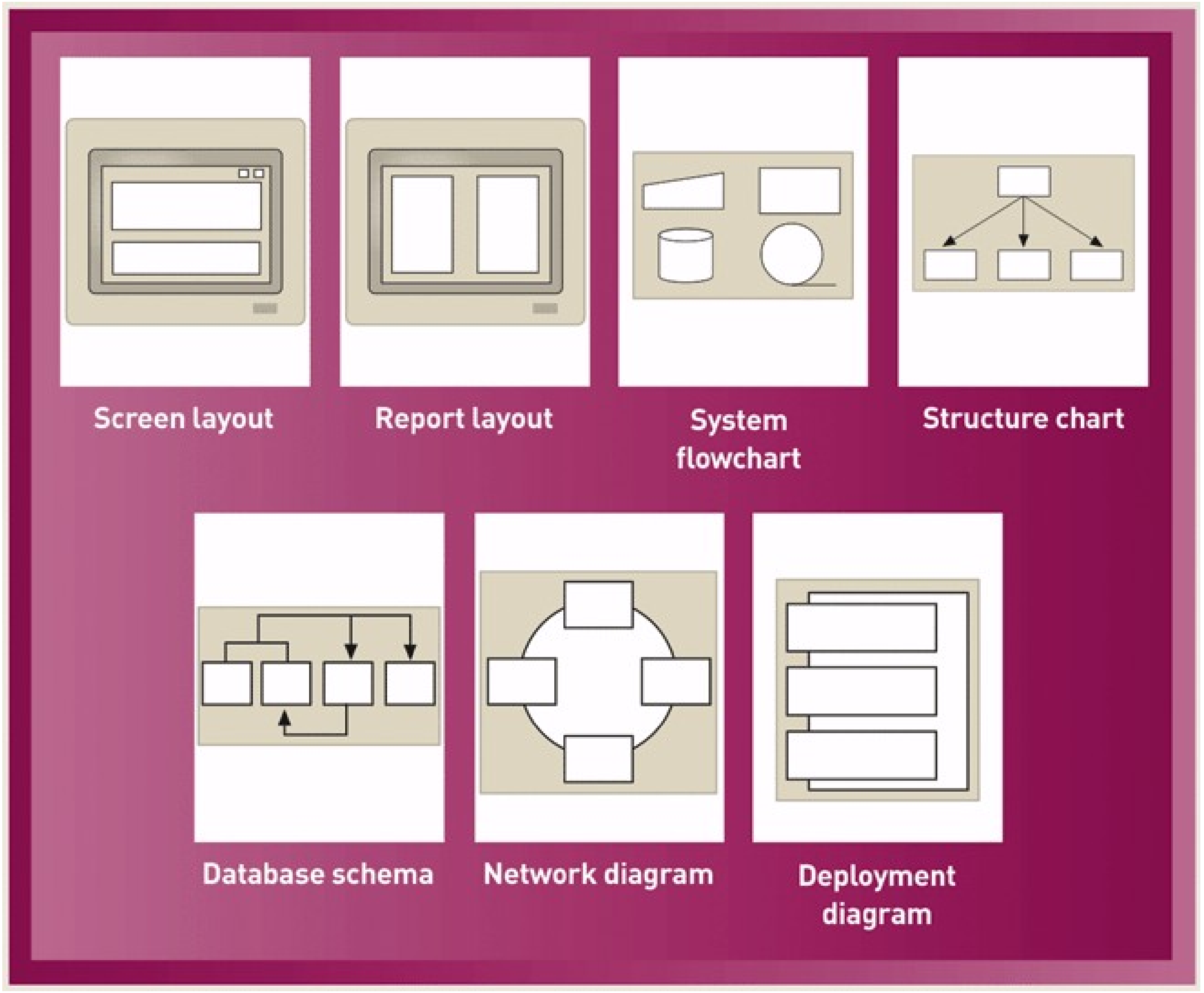
Overview of Models Used in Analysis and Design

- Analysis phase activity named “define system requirements”
  - Logical models
  - Provide detail without regard to specific technology
- Design phase
  - Physical models
  - Provide technical details
  - Extend logical models

Models Created by Analysis Activities
(Figure 5-4)



Models Used in Design (Figure 5-5)



DATA MODELING

- Define system requirements by understanding system information that needs to be stored
- Store information about things in the problem domain that people deal with when they do their work
- What things does the system need to know about and store information about?

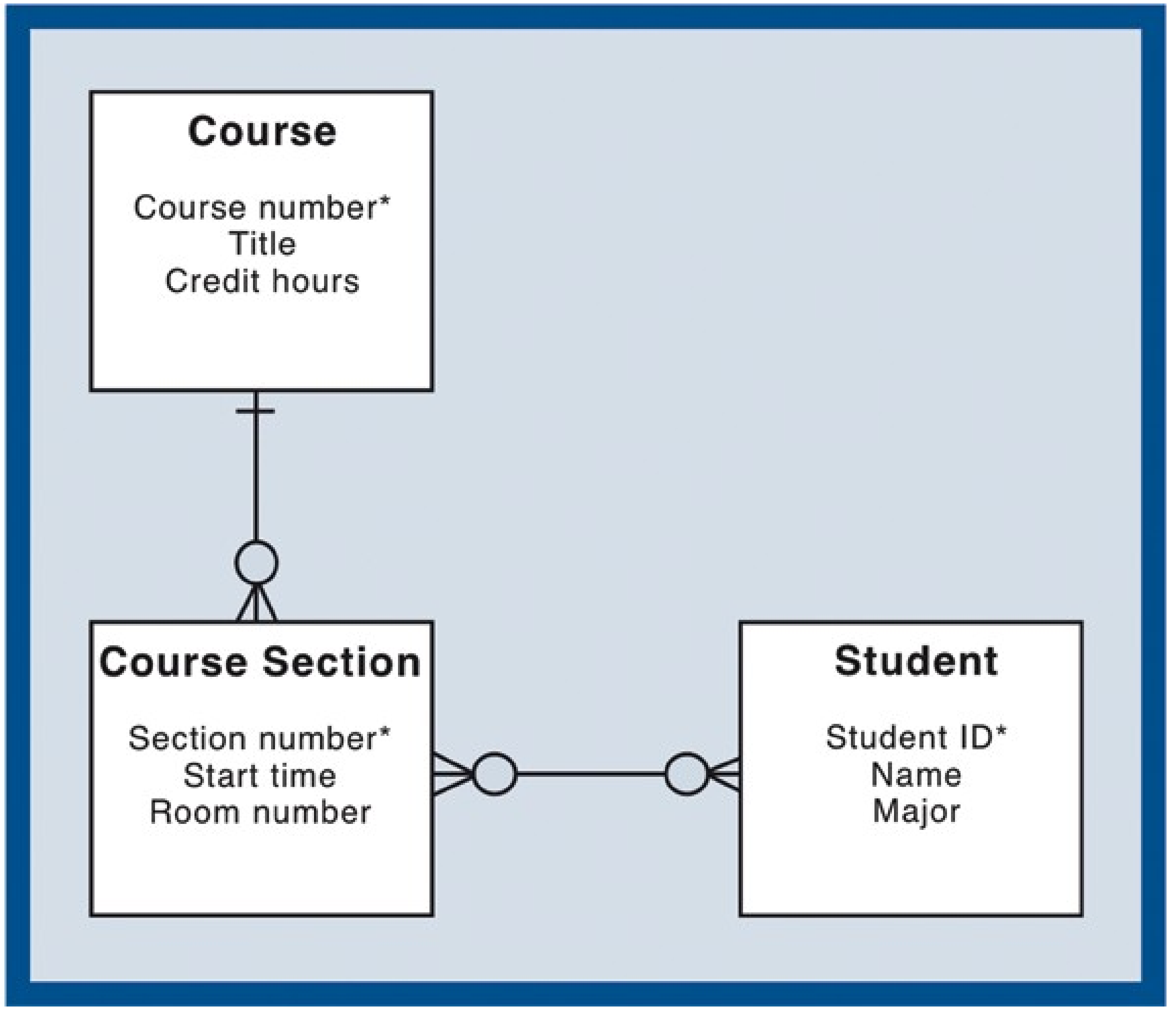
STEPS TO CREATE A DATA MODEL

- 1. DETERMINE CLASSES/ENTITY SETS
- E.g. GUST's data= student +registration+ course schedule….
- 2. DETERMINE ATTRIBUTES/Fields
- E.g. student= id+sname+major….
- 3. DETERMINE RELATIONSHIPS
- E.g. student related to registration
- 4. DETERMINE RELATIONSHIP CARDINALITIES: 1 to many, 1 to 1, many to many

ERD with Many-to-Many Relationship

Figure 5-27

A university course enrollment ERD with a many-to-many relationship



Many-to-Many Relationship Converted to Associative Entity to Store Grade Attribute

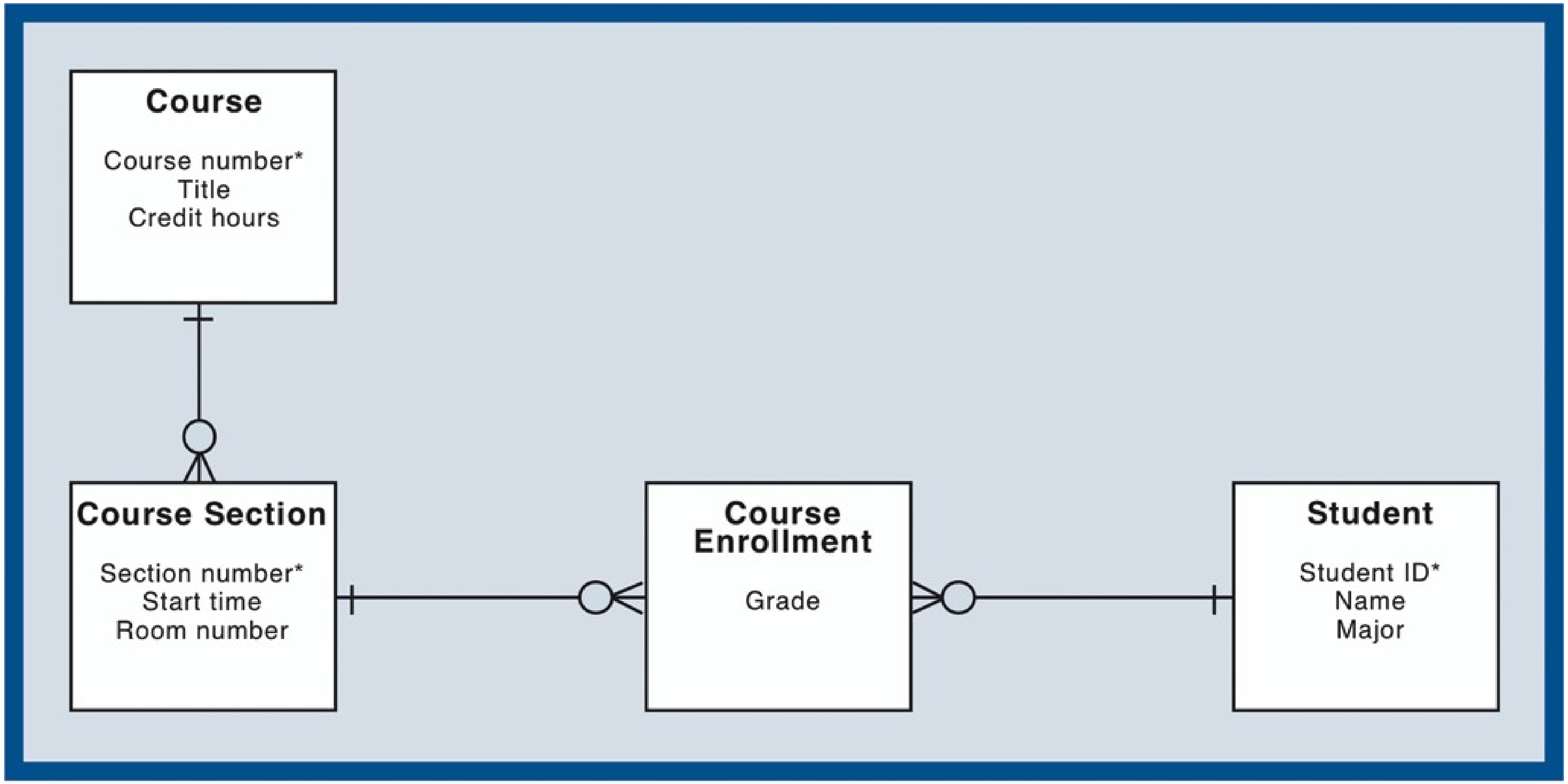
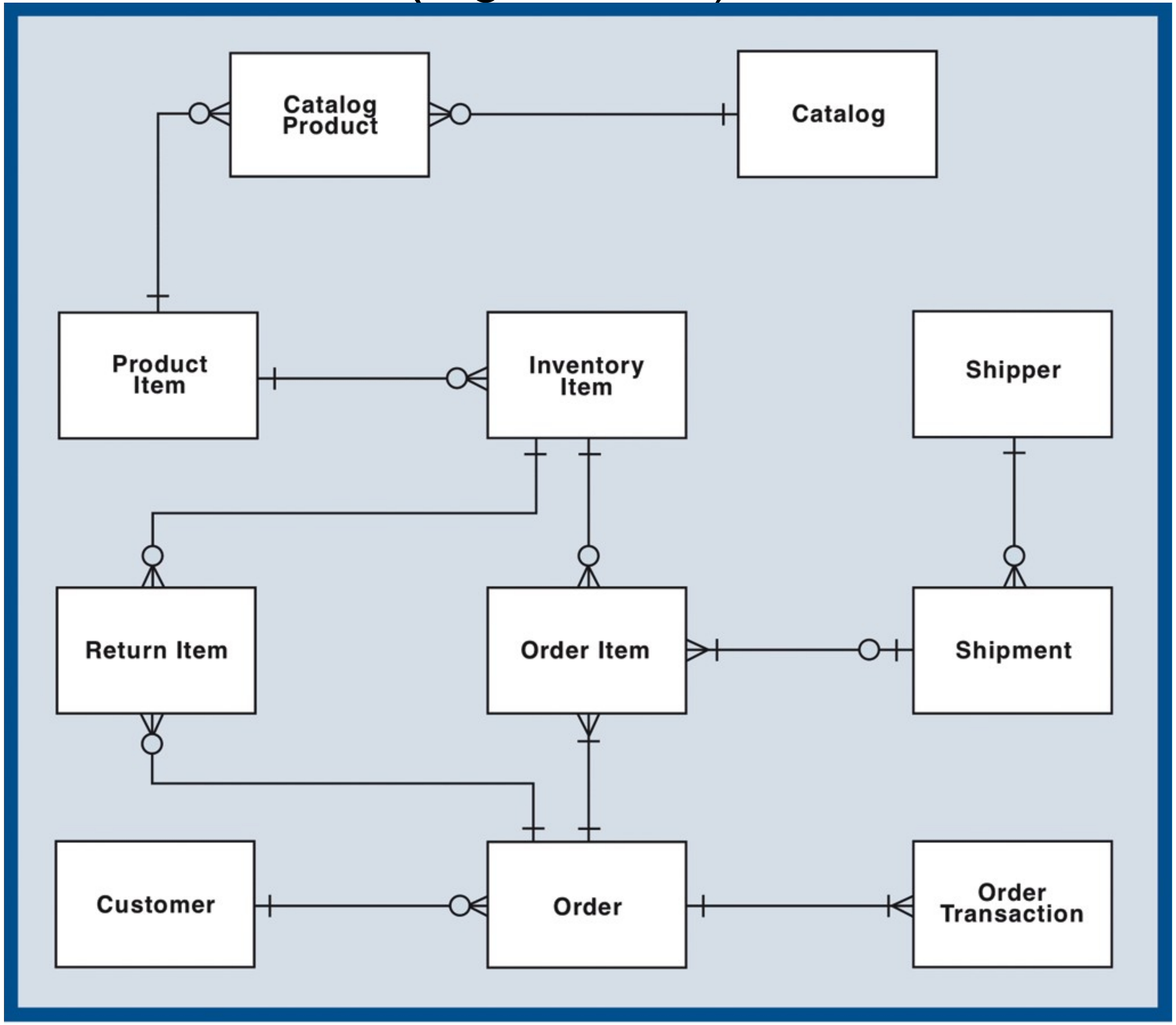


Figure 5-28

A refined university course enrollment ERD with an associative entity

RMO Customer Support System ERD (Figure 5-29)



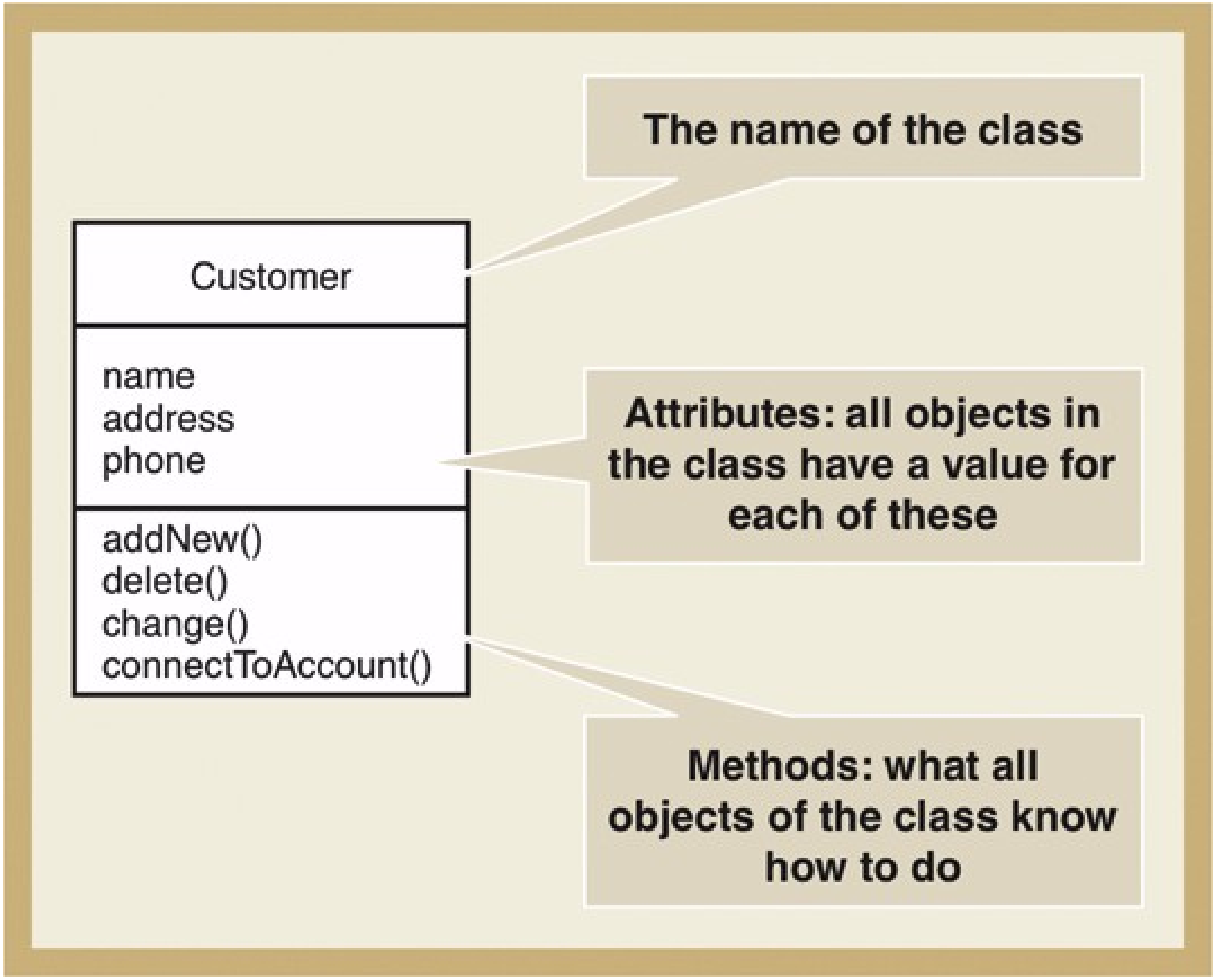
THE MIDTERM

- PART A: 30%
- BOOKWORK: short essays from chapters 1 and 2
- PART B: 70%
- CASE STUDY: QUESTIONS:
- 1. state the scope and objectives
- 2. fill in the functional matrix
- 3. fill in the problem matrix
- 4. do a brief one feasibility type
- 5. Context Diagram
- 6. Gantt Chart
- 7. WBS/fuctional decomposition diagram
- 8. PERT/CPM 9. Data model (Class diagram/ERD)

The Class Diagram

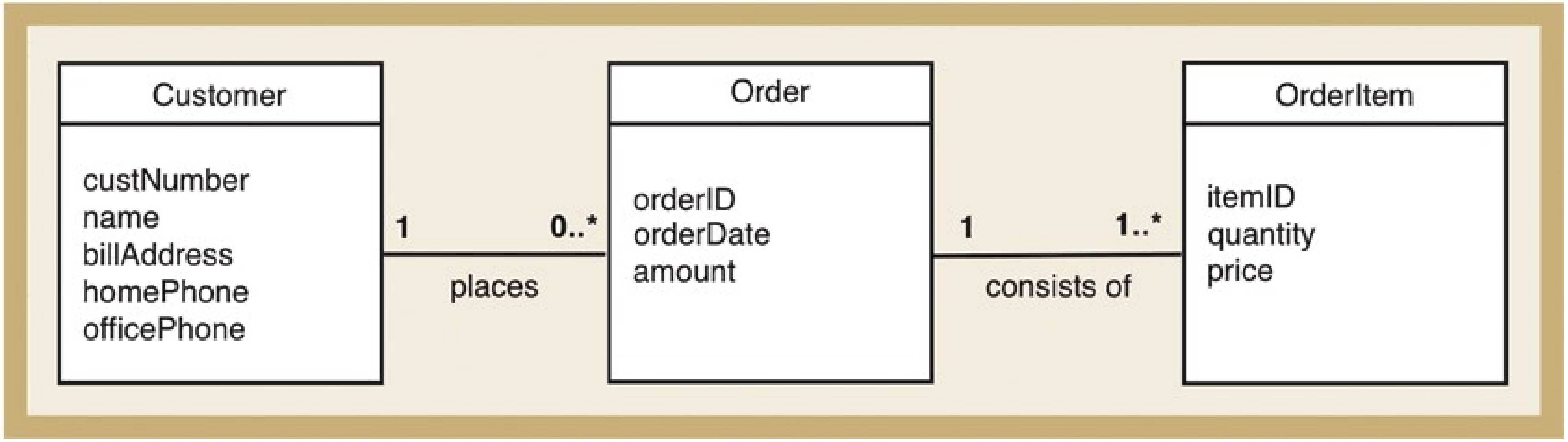
- Unified Modeling Language (UML) diagram
- Domain model class diagram
  - Models things in the users' work domain
  - Used to define requirements for OO (very similar to entities in ERD)
- Design class diagram
  - Models software classes
  - Adds methods as behaviors
  - Used in the design activity

UML Class Symbol (Figure 5-30)


The name of the class
Customer
name
address
phone
Attributes: all objects in the class have a value for each of these
addNew()
delete()
change()
connectToAccount()
Methods: what all objects of the class know how to do

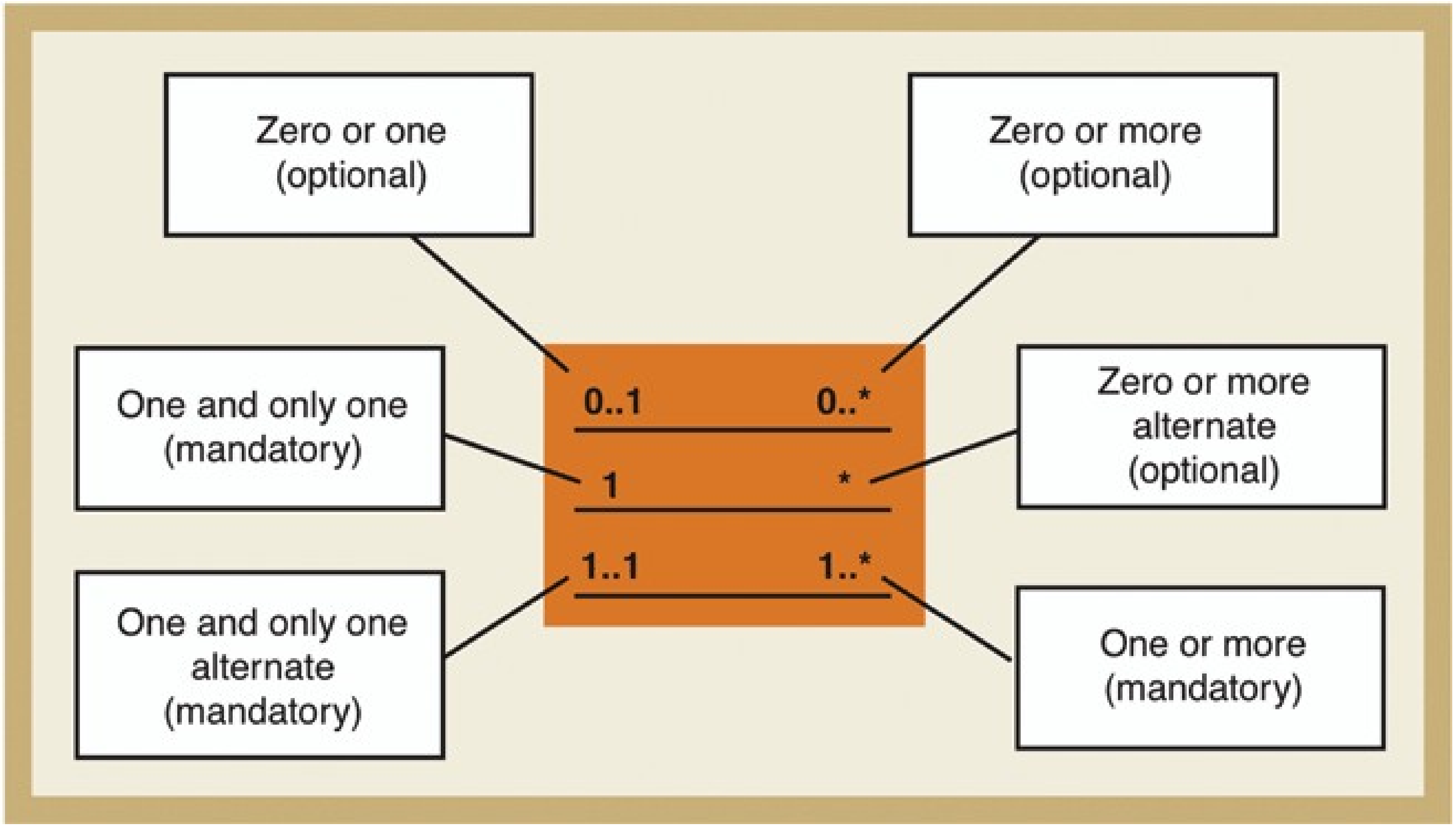
Simple Domain Model Class Diagram

(Figure 5-31)

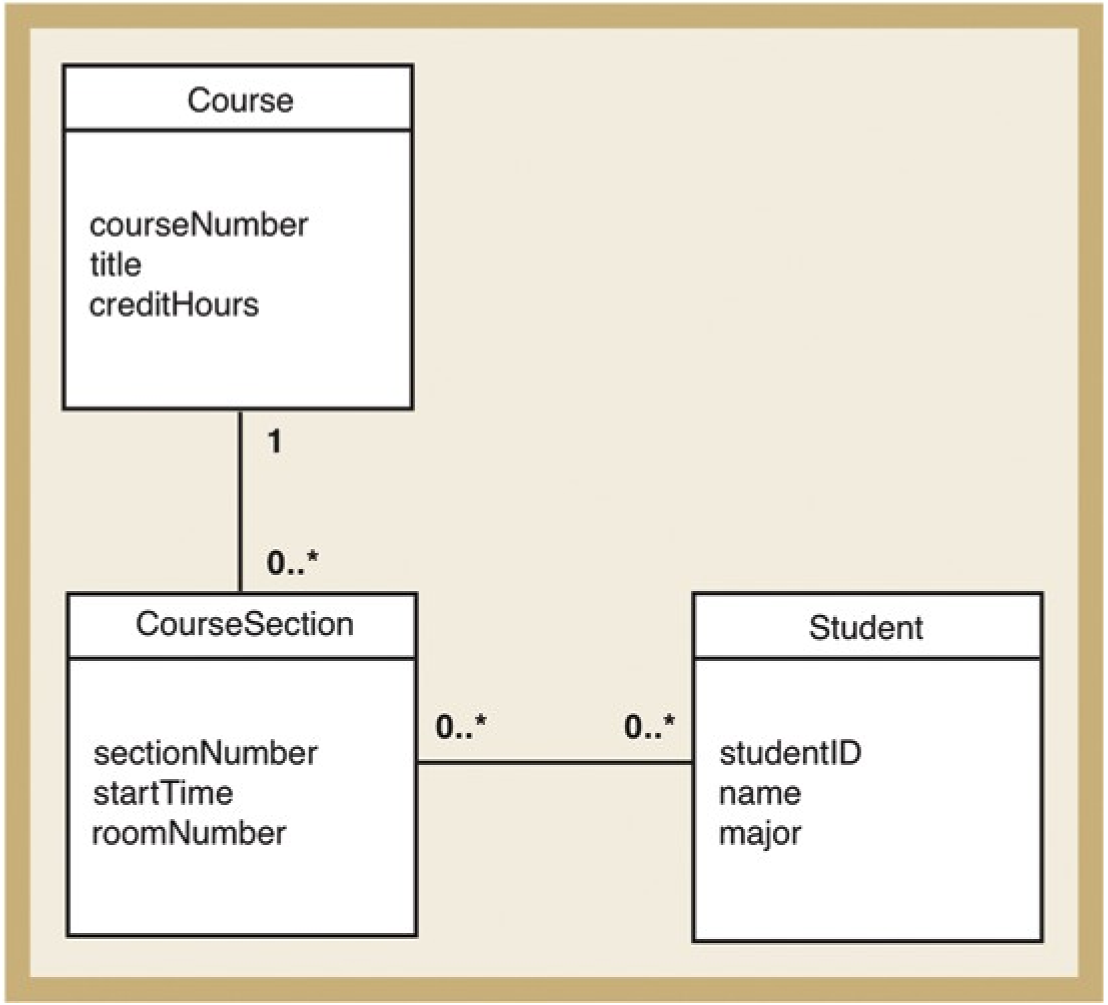


- No methods shown in domain model
  - Domain classes are not software classes
- Very similar to ERD in Figure 5-25
  - UML and domain model can be used in place of ERD in traditional approach

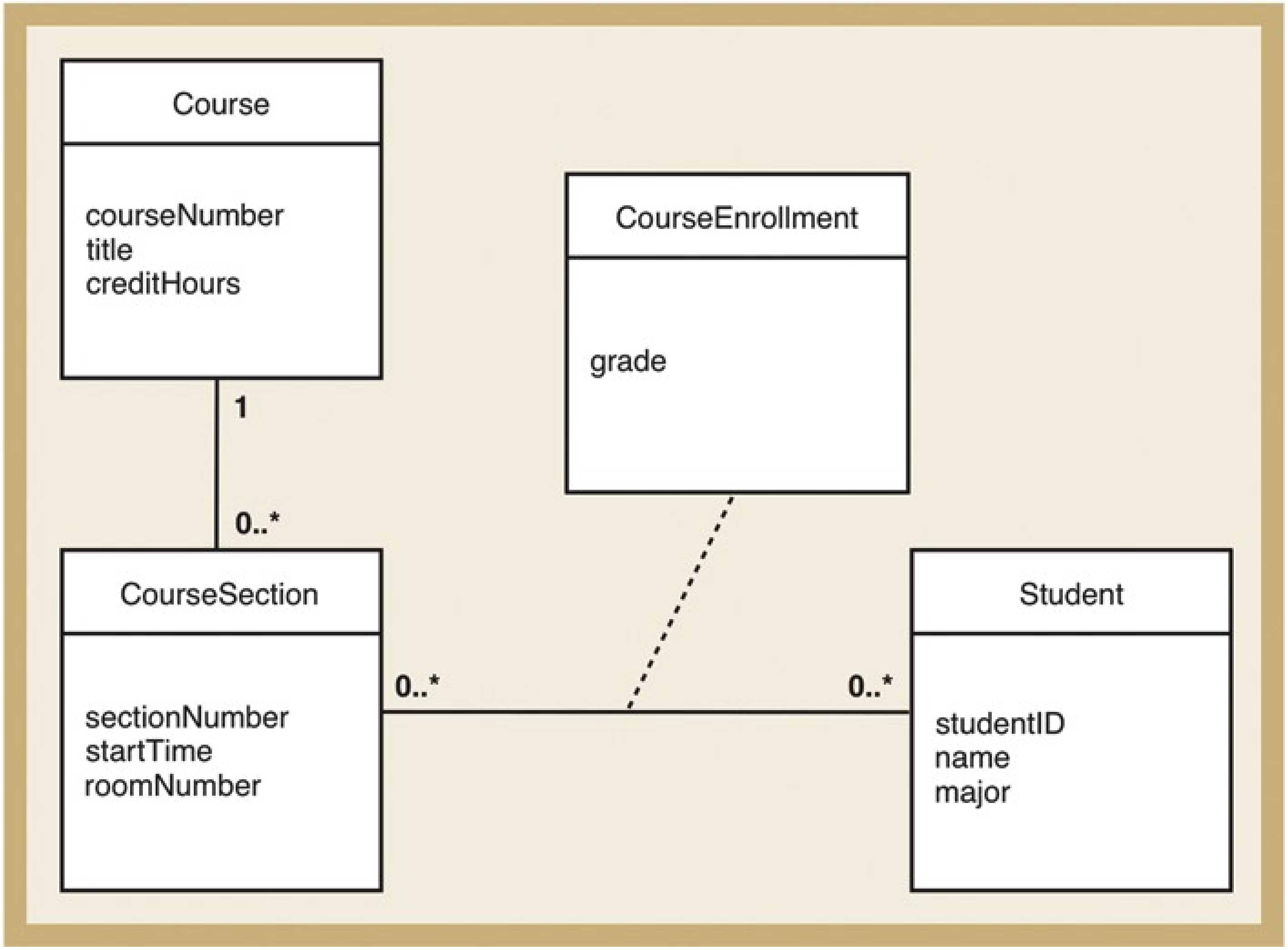
Multiplicity of Associations (Figure 5-32)
=cardinality


Zero or one (optional)
Zero or more (optional)
One and only one (mandatory)
Zero or more alternate (optional)
One and only one alternate (mandatory)
One or more (mandatory)
0..1
0..*
1
*
1..1
1..*

University Course Enrollment Domain Model Class Diagram (Figure 5-33)



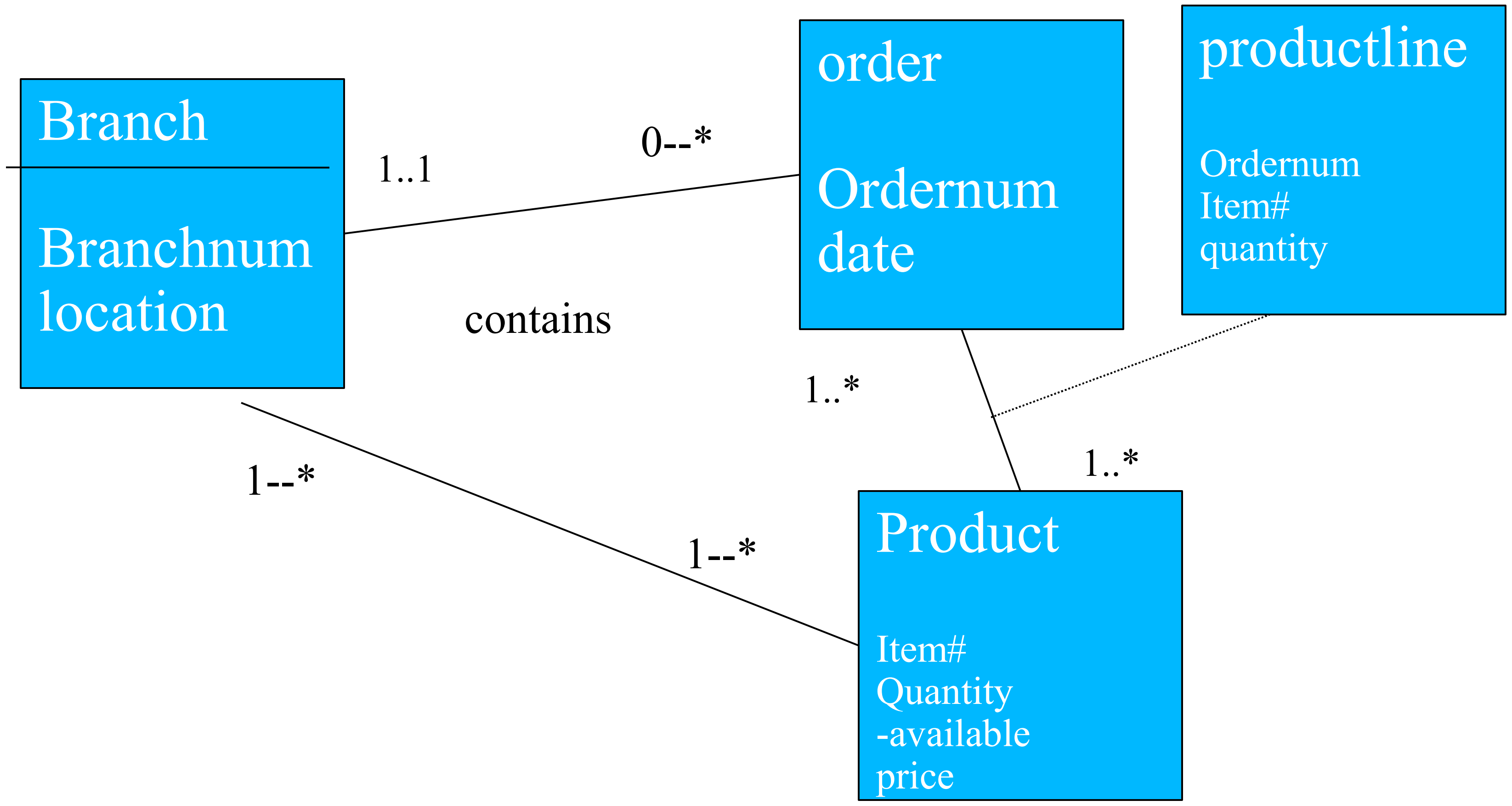
Refined Model with Association Class and Grade Attribute (Figure 5-34)



- Note 1:
- A weak link results from a many-to-many relationship ONLY
- Note 2:
- A link class (weak) usually contains (implicitly) the primary keys of the classes it links.
- Note 3:

A simple e.g.

- A factory has several branches. A plant/branch receives many orders. An order may contain many product lines. A product may be manufactured by any of the existing plants. What does the data model look like?


Branch
Branchnum
location
1..1
contains
0--*
order
Ordernum
date
productline
Ordernum
Item#
quantity
1..*
1..*
1--*
1--*
Product
Item#
Quantity
-available
price

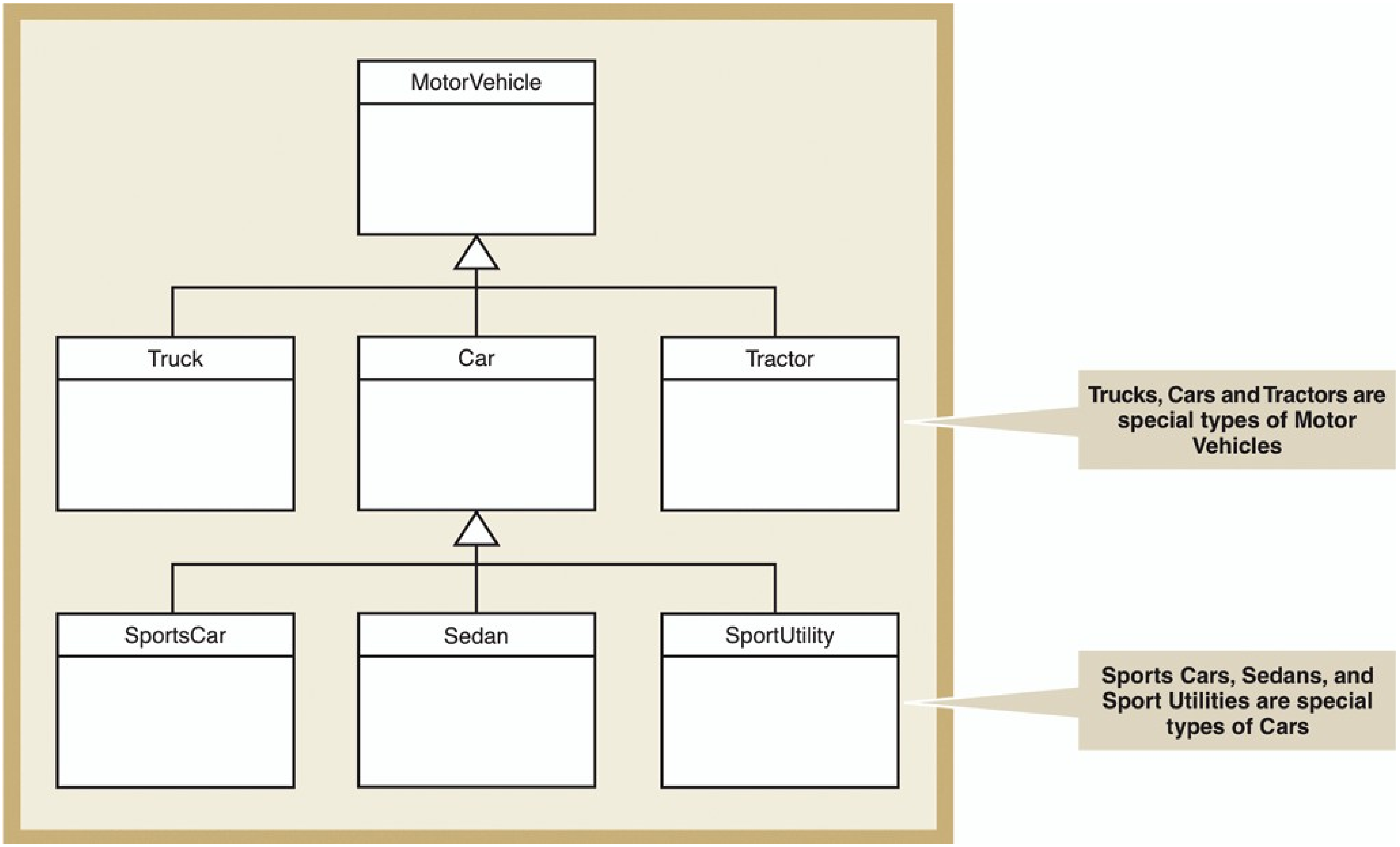
An exercise ☺

- Build a UML model 4 the following:
- A supplier can supply any number of parts to any project the company is handling. A part can be supplied also to any number of projects. Also a supplier can supply any number of projects. You must handle the quantitiy supplied of each part to any project by any supplier. What does the model look like (that's challenging!!!!!!!!!!!!!)

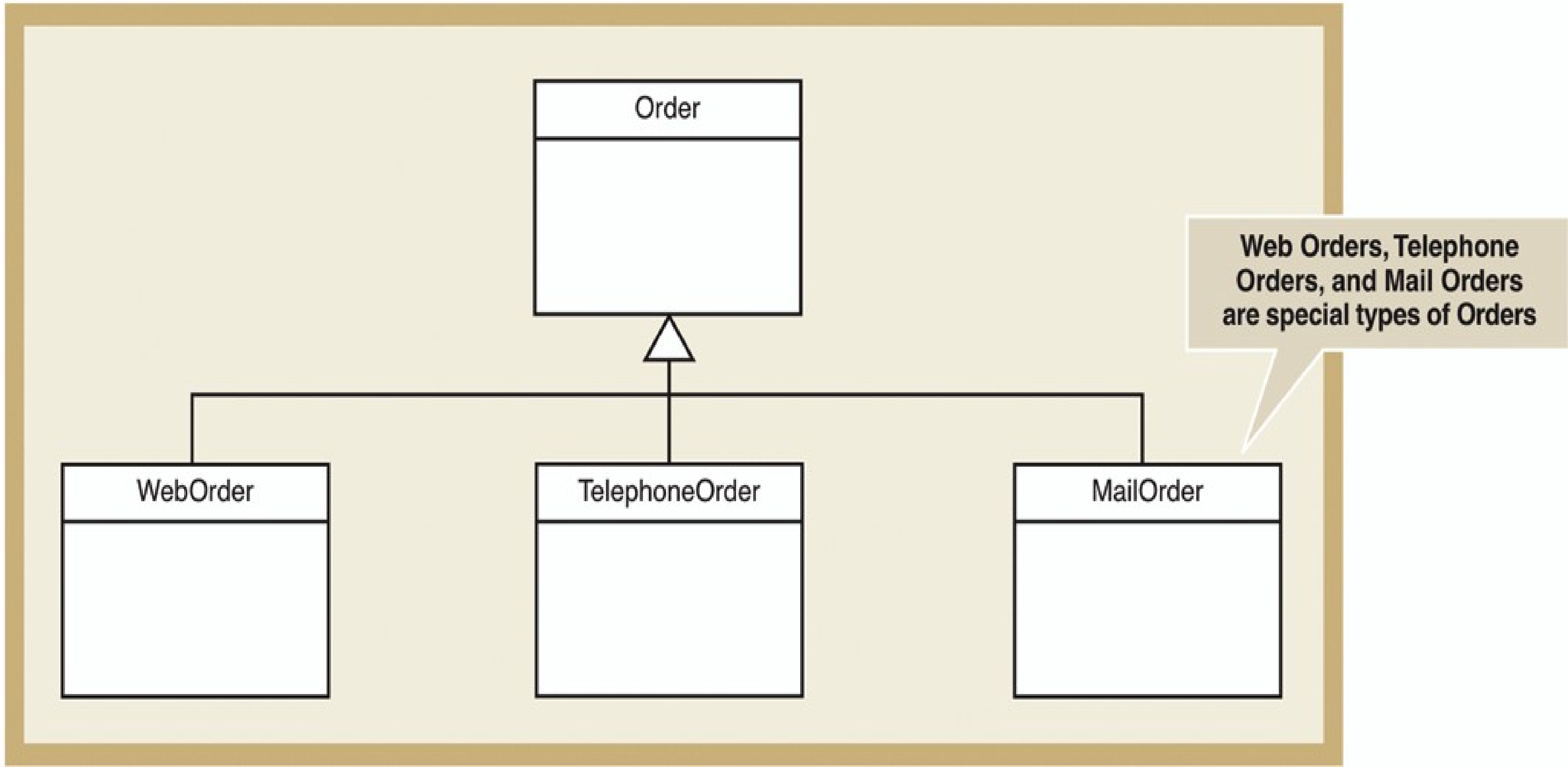
More Complex Class Concepts

- Generalization/specialization hierarchies
  - General superclasses to specialized subclasses
  - Inheritance allows subclasses to share characteristics of their superclasses
- Whole-part hierarchies (object and its parts)
  - Aggregation – parts can exist separately
  - Composition – parts can't exist separately
    - Hand has fingers and thumb

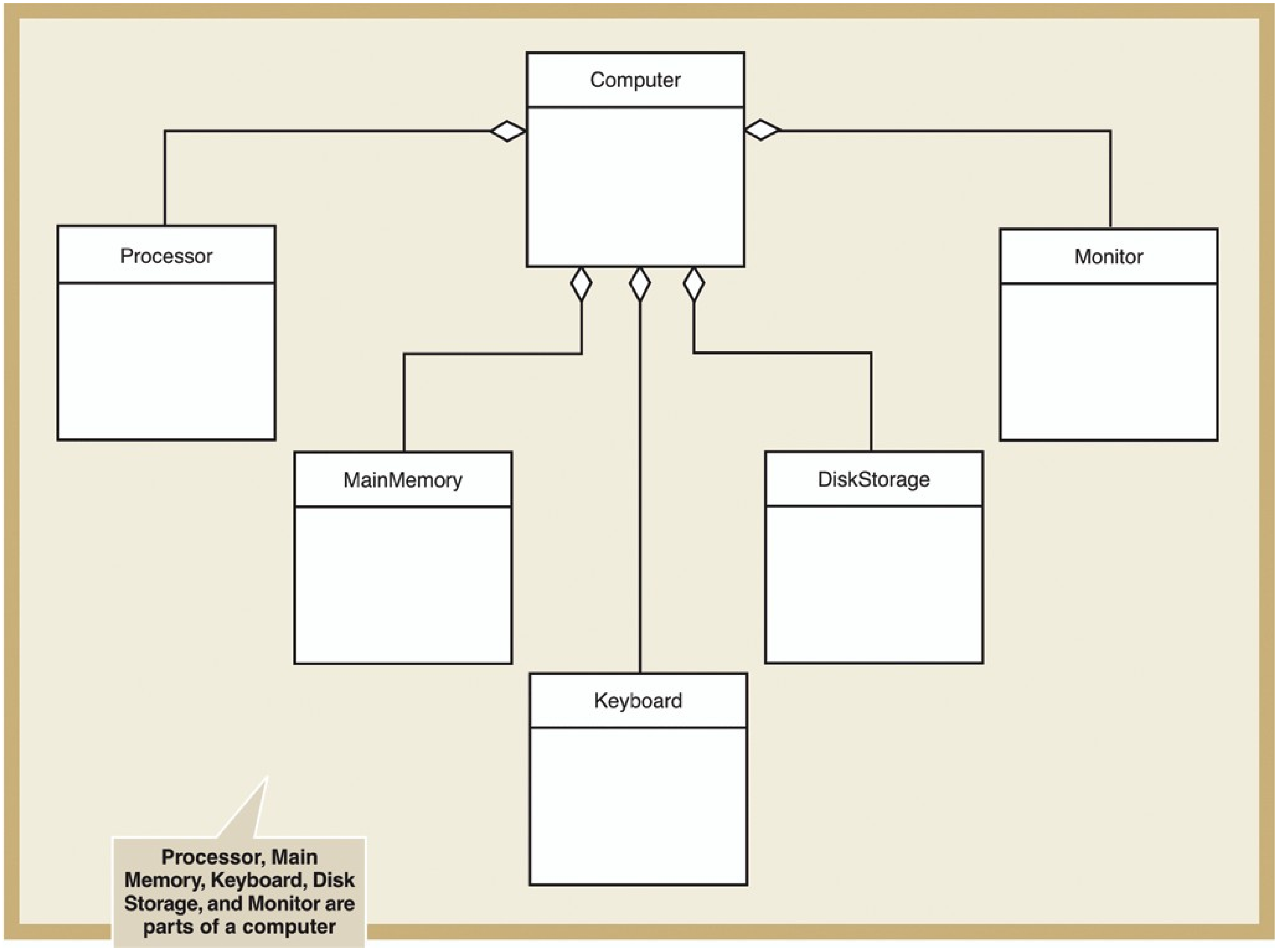
A Generalization/Specialization Class Hierarchy for Motor Vehicles (Figure 5-35)


MotorVehicle
Truck
Car
Tractor
SportsCar
Sedan
SportUtility
Trucks, Cars and Tractors are special types of Motor Vehicles
Sports Cars, Sedans, and Sport Utilities are special types of Cars

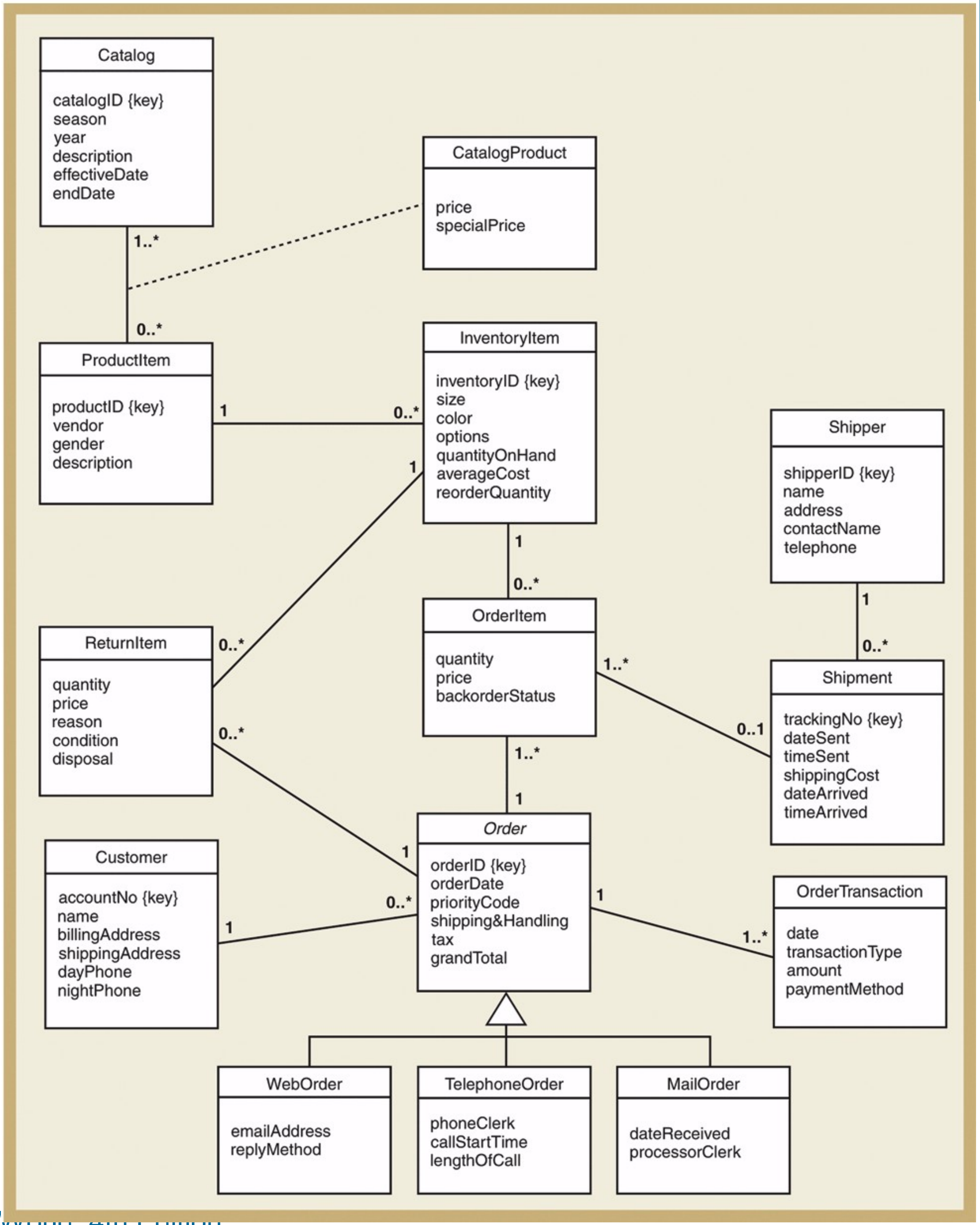
A Generalization/Specialization Class Hierarchy for RMO Orders (Figure 5-36)


Order
WebOrder
TelephoneOrder
MailOrder
Web Orders, Telephone Orders, and Mail Orders are special types of Orders

Whole-Part Aggregation Relationships
(Figure 5-37)


Computer
Processor
Monitor
MainMemory
DiskStorage
Keyboard
Processor, Main Memory, Keyboard, Disk Storage, and Monitor are parts of a computer

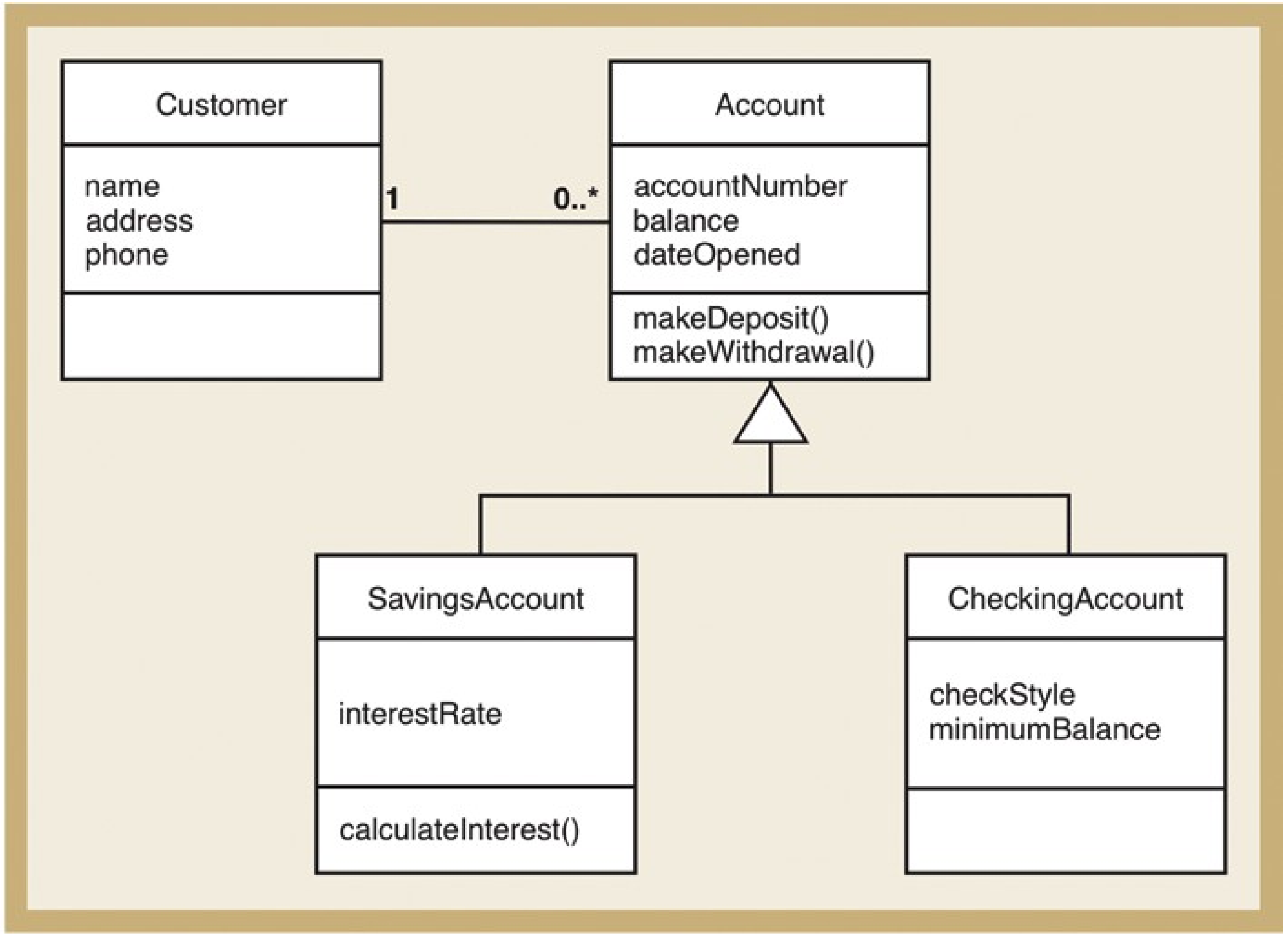
RMO Domain Model Class Diagram
(Figure 5-41)



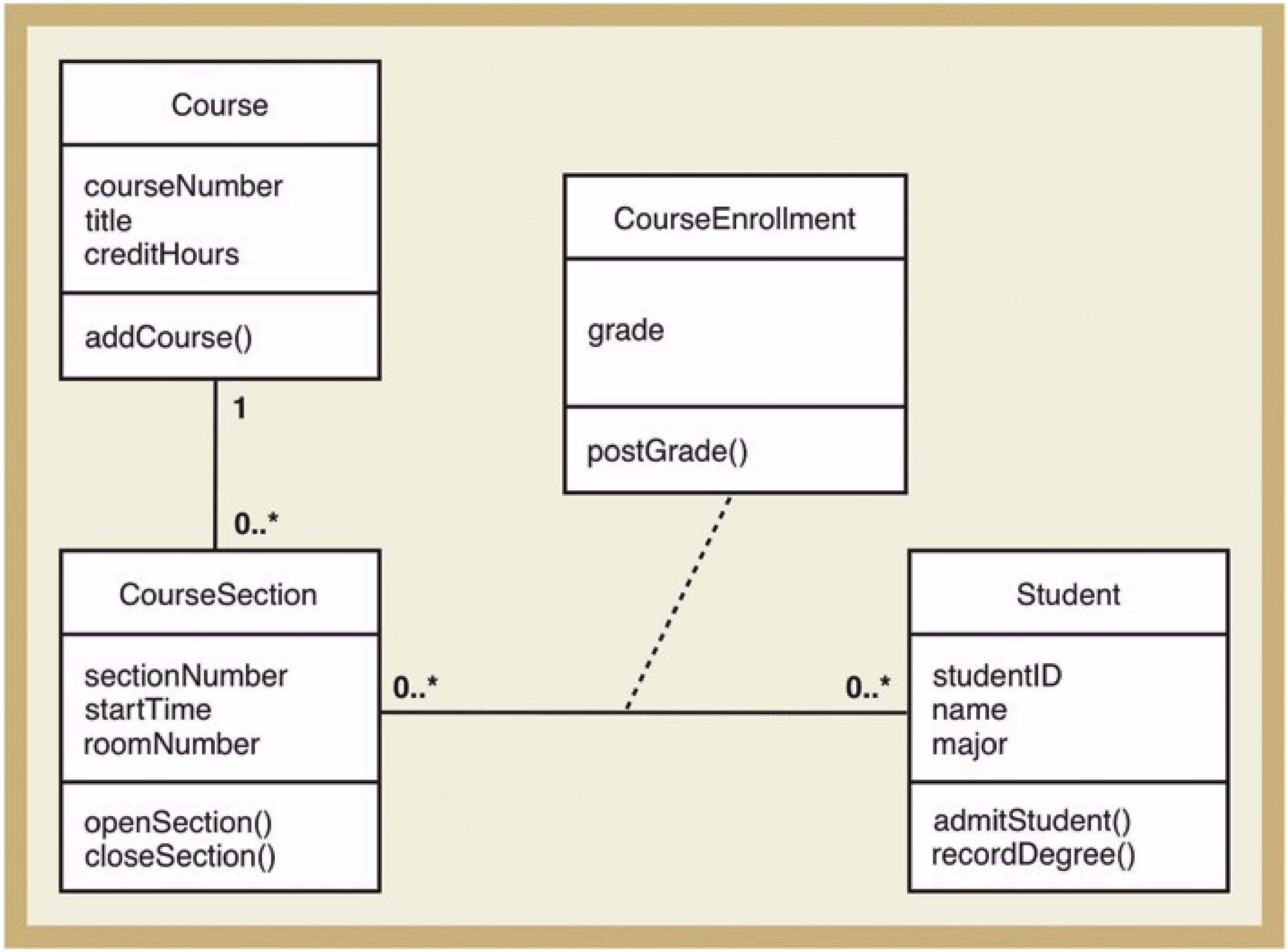
Design Class Diagram Notation:
Software Classes with Methods

Figure 5-38

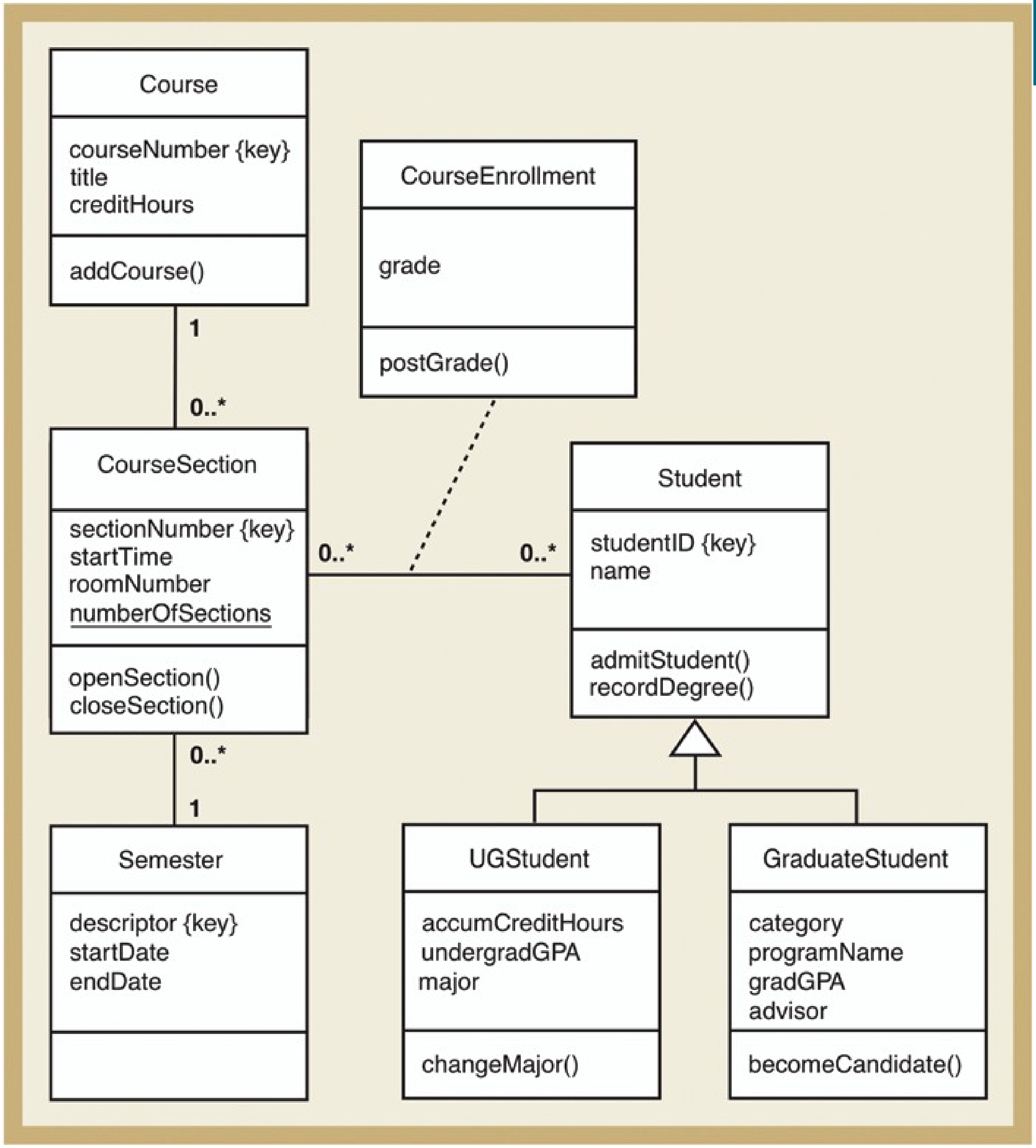
A bank account system design class diagram (with methods)



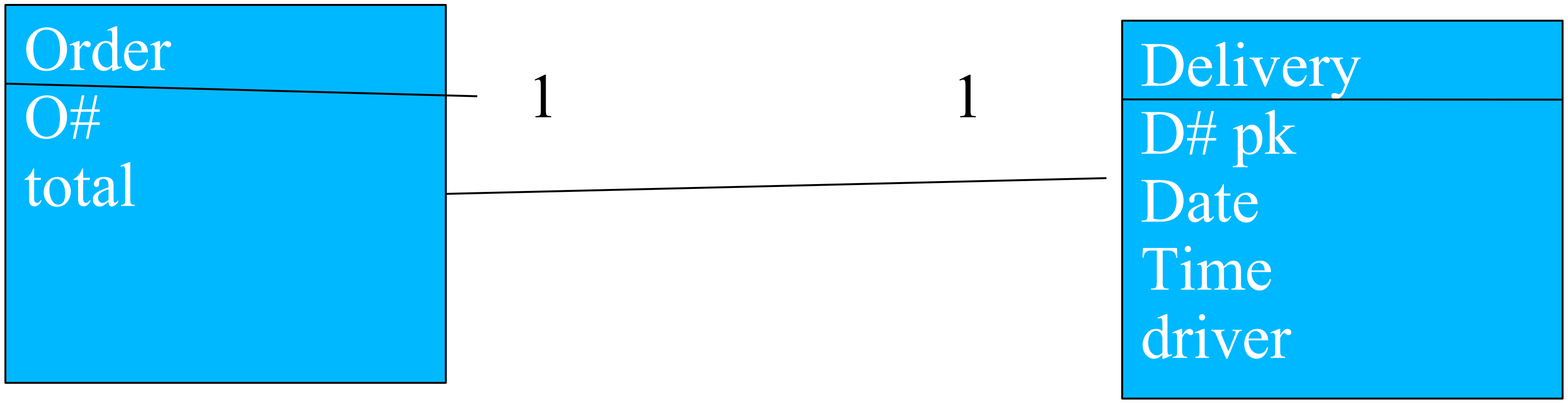
Course Enrollment Design Class Diagram with Association Class (Figure 5-39)



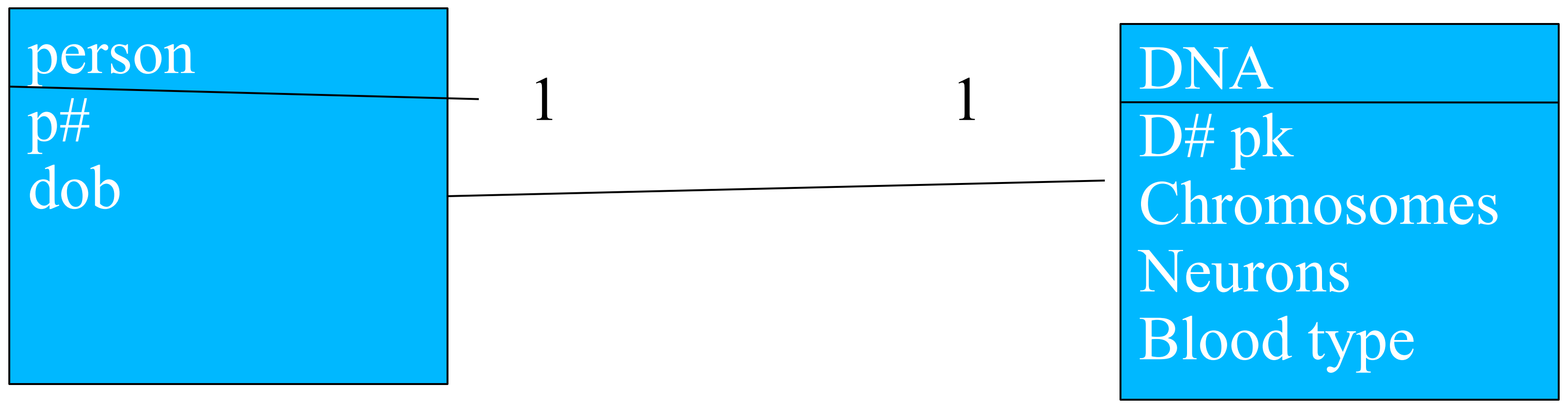
Expanded Course Enrollment Design Class Diagram
(Figure 5-40)



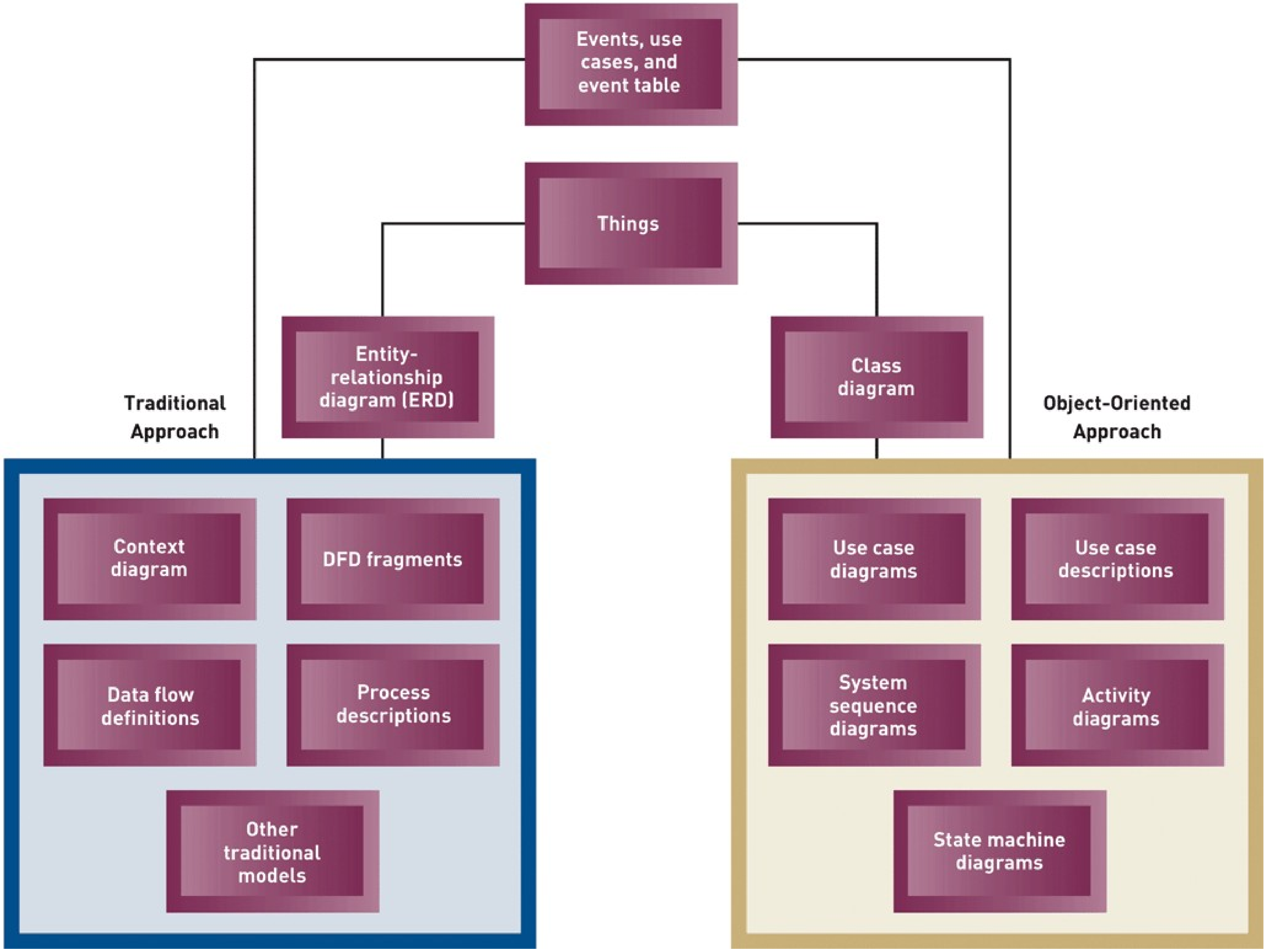
1-to-1 example



1-to-1 example



Where You Are Headed (Figure 5-42)



Summary

- Analysis phase – defines system requirements
- Models created to further learning process, reduce complexity, communicate with team members, and document requirements
- Many types of models used
  - Mathematical, descriptive, graphical
- Key early step in modeling is to identify and list
  - Events that require a use case in the system
  - Things users deal with in work environment

Summary (continued)

- Use cases (activities) are identified from user goals and business events that trigger elementary business processes
- Business events are memorable, can be described, and occur at a specific time and place
  - External events, temporal events, and state events
- Event table records event, trigger, source, use case, response, and destination
  - A catalog of information about each use case

Summary (continued)

- "Things" are what user deals with and system remembers, such as customer placing an order
- Traditional approach uses entity-relationship diagrams (ERD) for data entities, attributes of data entities, and relationships between entities
- Object-oriented approach uses UML class diagrams for classes, attributes, methods of class, and associations among classes
  - Domain model class diagram (requirements activity)
  - Design class diagram (design activity)