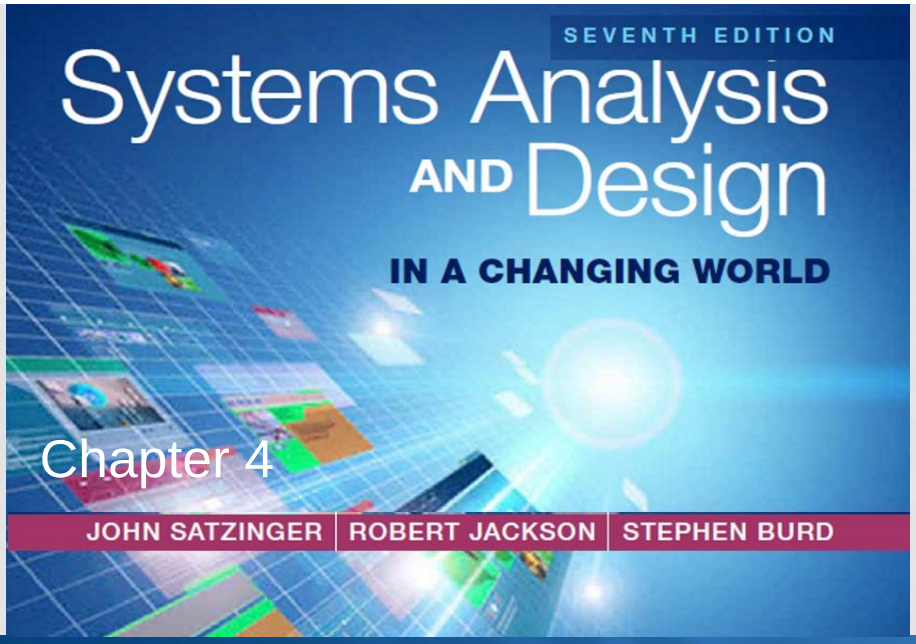


Domain Modeling

Chapter 4
Lecture 1

Systems Analysis and Design in a Changing World 7th Ed

Satzinger, Jackson & Burd

Chapter 4 Outline

- "Things" in the Problem Domain
- The Entity-Relationship Diagram
- The Domain Model Class Diagram
- The State Machine Diagram – Identifying Object Behavior

Learning Objectives

- Explain how the concept of "things" in the problem domain also define requirements
- 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 domain model class diagram
- Understand the domain model class diagram for the RMO Consolidated Sales and Marketing System
- Read, interpret, and create a state machine diagram that models object behavior

Overview

- This chapter focuses on another key concept for defining requirements— data entities or domain classes
- In the RMO Tradeshow System from Chapter 1, some domain classes are Supplier, Product, and Contact
- In this chapter's opening case Waiters on Call, examples of domain classes are Restaurants, Menu Items, Customers, Orders, Drivers, Routes, and Payments

Things in the Problem Domain

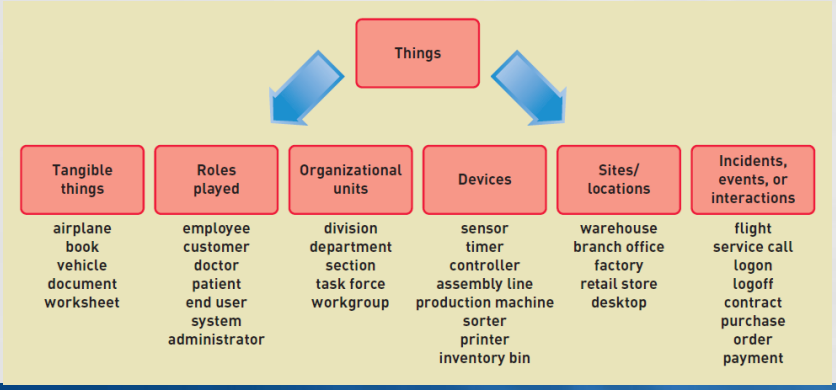
- Problem domain—the specific area (or domain) of the users' business need that is within the scope of the new system.
- "Things" are those items users work with when accomplishing tasks that need to be remembered
- Examples of "Things" are products, sales, shippers, customers, invoices, payments, etc.
- These "Things" are modeled as domain classes or data entities
- In this course, we will call them domain classes. In database class you may call them data entities

Things in the Problem Domain
Two Techniques for Identifying Them

- Brainstorming Technique
  - Use a checklist of all of the usual types of things typically found and brainstorm to identify domain classes of each type
- Noun Technique
  - Identify all of the nouns that come up when the system is described and determine if each is a domain class, an attribute, or not something we need to remember

Brainstorming Technique

- Are there any tangible things? Are there any organizational units? Sites/locations? Are there incidents or events that need to be recorded?



Brainstorming Technique: Steps

1. Identify a user and a set of use cases
2. Brainstorm with the user to identify things involved when carrying out the use case—that is, things about which information should be captured by the system.
3. Use the types of things (categories) to systematically ask questions about potential things, such as the following: Are there any tangible things you store information about? Are there any locations involved? Are there roles played by people that you need to remember?
4. Continue to work with all types of users and stakeholders to expand the brainstorming list
5. Merge the results, eliminate any duplicates, and compile an initial list

The Noun Technique

- A technique to identify problem domain classes (things) by finding, classifying, and refining a list of nouns that come up in in discussions or documents
- Popular technique. Systematic.
- Does end up with long lists and many nouns that are not things that need to be stored by the system
- Difficulty identifying synonyms and things that are really attributes
- Good place to start when there are no users available to help brainstorm

The Noun Technique: Steps

1. Using the use cases, actors, and other information about the system— including inputs and outputs—identify all nouns.
   - For the RMO CSMS, the nouns might include customer, product item, sale, confirmation, transaction, shipping, bank, change request, summary report, management, transaction report, accounting, back order, back order notification, return, return confirmation…
2. Using other information from existing systems, current procedures, and current reports or forms, add items or categories of information needed.
   - For the RMO CSMS, these might include price, size, color, style, season, inventory quantity, payment method, and shipping address.

The Noun Technique: Steps (continued)

3. As this list of nouns builds, refine it. Ask these questions about each noun to help you decide whether you should include it:
   - Is it a unique thing the system needs to know about?
   - Is it inside the scope of the system I am working on?
   - Does the system need to remember more than one of these items?

   Ask these questions to decide to exclude it:
   - Is it really a synonym for some other thing I have identified?
   - Is it really just an output of the system produced from other information I have identified?
   - Is it really just an input that results in recording some other information I have identified?

   Ask these questions to research it:
   - Is it likely to be a specific piece of information (attribute) about some other thing I have identified?
   - Is it something I might need if assumptions change?

The Noun Technique: Steps (continued)

4. Create a master list of all nouns identified and then note whether each one should be included, excluded, or researched further.

5. Review the list with users, stakeholders, and team members and then define the list of things in the problem domain.

Partial List of Nouns for RMO

- With notes on whether to include as domain class

Identified noun	Notes on including noun as a thing to store
Accounting	We know who they are. No need to store it.
Back order	A special type of order? Or a value of order status? Research.
Back-order information	An output that can be produced from other information.
Bank	Only one of them. No need to store.
Catalog	Yes, need to recall them, for different seasons and years. Include.
Catalog activity reports	An output that can be produced from other information. Not stored.
Catalog details	Same as catalog? Or the same as product items in the catalog? Research.
Change request	An input resulting in remembering changes to an order.
Charge adjustment	An input resulting in a transaction.
Color	One piece of information about a product item.
Confirmation	An output produced from other information. Not stored.
Credit card information	Part of an order? Or part of customer information? Research.
Customer	Yes, a key thing with lots of details required. Include.
Customer account	Possibly required if an RMO payment plan is included. Research.
Fulfillment reports	An output produced from information about shipments. Not stored.
Inventory quantity	One piece of information about a product item. Research.
Management	We know who they are. No need to store.
Marketing	We know who they are. No need to store.
Merchandising	We know who they are. No need to store.

Details about Domain Classes

- Attribute— describes one piece of information about each instance of the class
  - Customer has first name, last name, phone number
- Identifier or key
  - One attribute uniquely identifies an instance of the class. Required for data entities, optional for domain classes. Customer ID identifies a customer
- Compound attribute
  - Two or more attributes combined into one structure to simplify the model. (E.g., address rather than including number, street, city, state, zip separately). Sometimes an identifier or key is a compound attribute.

Attributes and Values

- Class is a type of thing. Object is a specific instance of the class. Each instance has its own values for an attribute

All customers have these attributes:	Each customer has a value for each attribute:		
Customer ID	101	102	103
First name	John	Mary	Bill
Last name	Smith	Jones	Casper
Home phone	555-9182	423-1298	874-1297
Work phone	555-3425	423-3419	874-8546

Domain Modeling

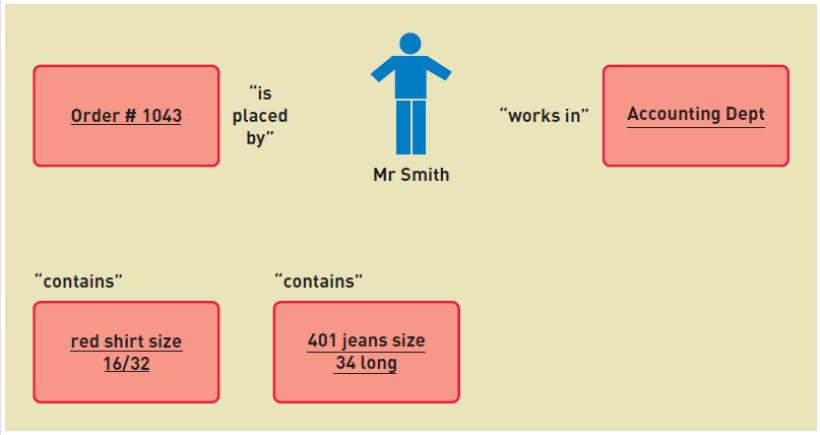
Chapter 4
Lecture 2

Systems Analysis and Design in a Changing World 7th Ed

Satzinger, Jackson & Burd

Associations Among Things

- Association— a naturally occurring relationship between classes (UML term)



Just to Clarify…

- Called ***association*** on class diagram in UML
  - **Multiplicity** is term for the number of associations between classes: 1 to 1 or 1 to many (synonym to cardinality)
  - UML is the primary emphasis of this text
- Called ***relationship*** on ERD in database class
  - **Cardinality** is term for number of relationships in entity relationship diagrams: 1 to 1 or 1 to many (synonym to multiplicity)
- Associations and Relationships apply in two directions
  - Read them separately each way
  - A customer places an order
  - An order is placed by a customer

Minimum and Maximum Multiplicity

- Associations have minimum and maximum constraints
  - minimum is zero, the association is optional
  - If minimum is at least one, the association is mandatory

Mr. Jones has placed no order yet, but there might be many placed over time. (Direction: Mr. Jones to Order)	→	multiplicity/cardinality is zero or more—optional relationship
A particular order is placed by Mr. Smith. There can't be an order without stating who the customer is. (Reverse direction: Order to Mr. Smith)	→	multiplicity/cardinality is one and only one—mandatory relationship
An order contains at least one item, but it could contain many items. (Direction: Order to OrderItem)	→	multiplicity/cardinality is one or more—mandatory relationship

Types of Associations

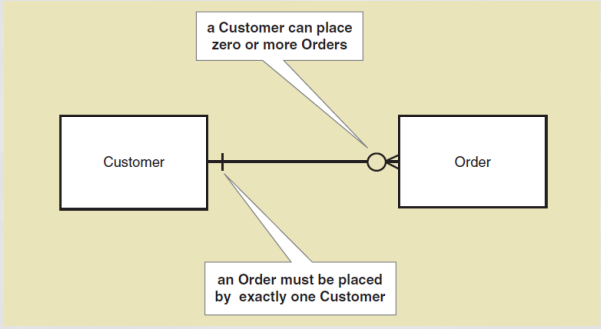
- Binary Association
  - Associations between exactly two different classes
    - Course Section includes Students
    - Members join Club
- Unary Association (recursive)
  - Associations between two instances of the same class
    - Person married to person
    - Part is made using parts
- Ternary Association (three)
- N-ary Association (between n)

Entity-Relationship Diagrams ERD

- ERDs have been used for many years to develop data models that are used in database development
- The term for "things" in ERD models is **data entities**
- ERD models are not UML models and do not use standard UML notation
- ERD models are not as expressive as UML models
  - They do not model generalization/specialization well
  - They do not model whole/part well

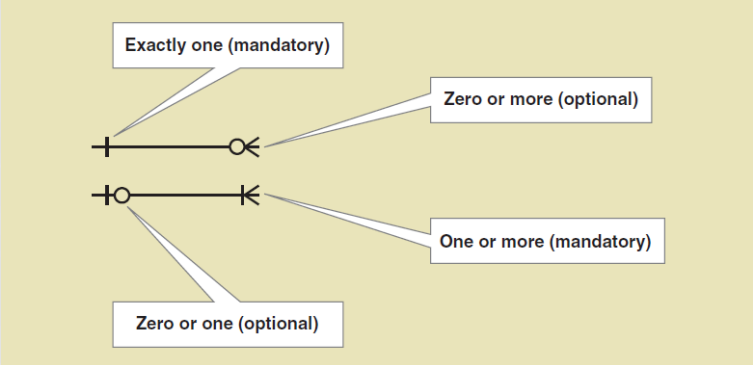
Example of ERD Notation

- ERD Models normally use “crows feet” notation to show cardinality



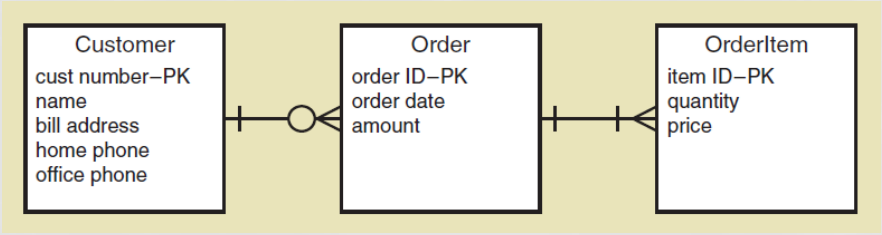
ERD Cardinality Symbols

- Examples of crows feet notation for various cardinalities



Expanded ERD with Attributes

- ERD with cardinalities and attributes
- There are several different notation methods for attributes in ERD models
- This notation places attributes within data entities

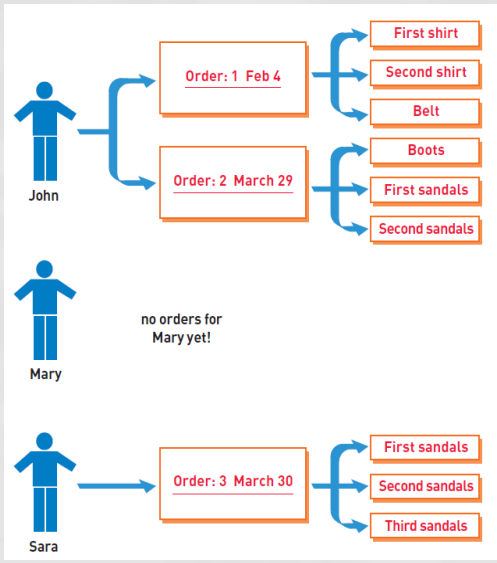


Semantic Net

- A **semantic net** is a graphical representation of an individual data entity and its relationship with other individual entities
- It is often used to help understand and then develop an ERD model

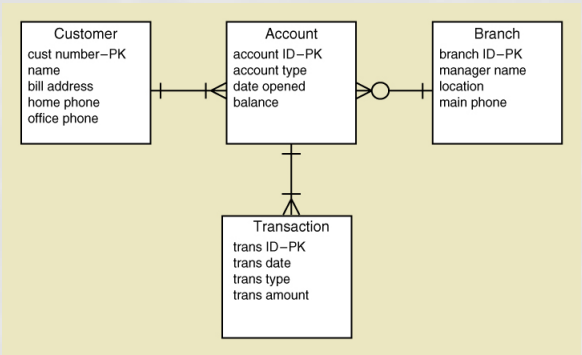
Semantic Net

- This example shows three classes.
- Quick quiz
  - What are the classes?
  - How many relationships?
  - What are min and max cardinalities?
  - What type of relationships are they?



An ERD for a Bank

- Quick Quiz
  - What are the key fields?
  - How many accounts can a customer have?
  - How many branches can a customer be assigned to?
  - How many customers can a branch have?

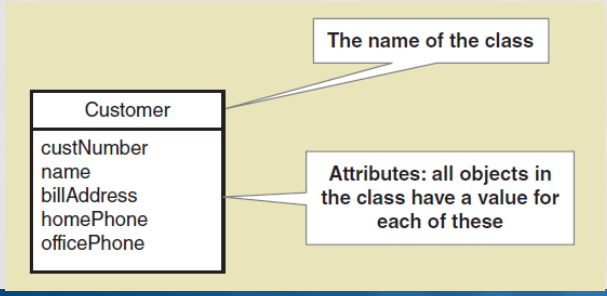


The Domain Model Class Diagram

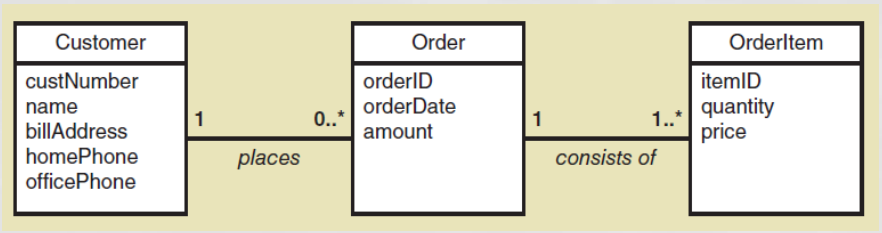
- Class
  - A type of classification used to describe a collection of objects
- Domain Class
  - Classes that describe objects in the problem domain
- Class Diagram
  - A UML diagram that shows classes with attributes and associations (plus methods if it models software classes)
- Domain Model Class Diagram
  - A class diagram that only includes classes from the problem domain, not software classes so no methods

UML Domain Class Notation

- Domain class a name and attributes (no methods)
- Class name is always capitalized
- Attribute names are not capitalized and use **camelback notation** (words run together and second word is capitalized)
- Compound class names also use camelback notation

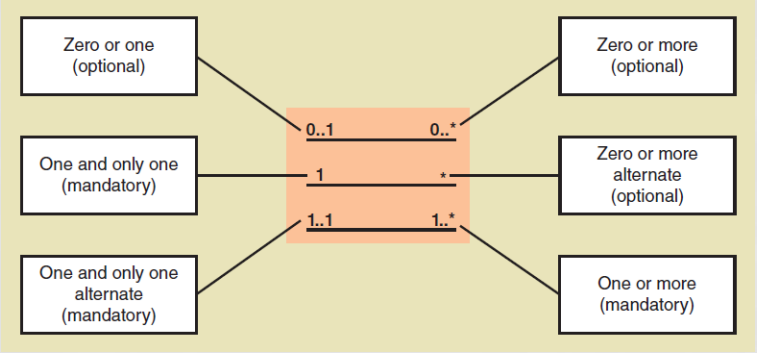


A Simple Domain Model Class Diagram



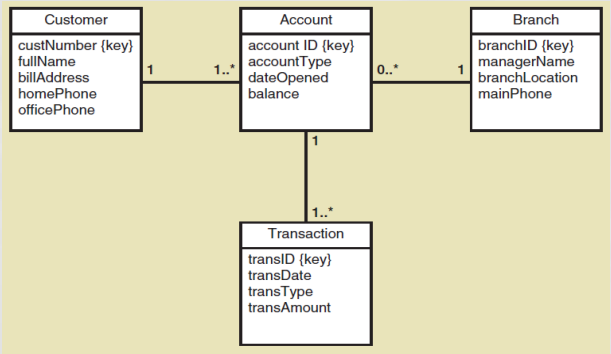
- Note: This diagram matches the semantic net shown previously
  - A customer places zero or more orders
  - An order is placed by exactly one customer
  - An order consists of one or more order items
  - An order item is part of exactly one order

UML Notation for Multiplicity



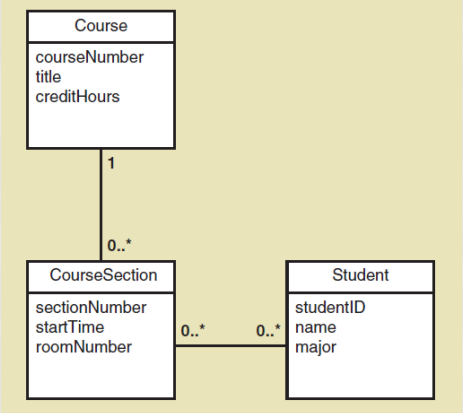
Domain Model Class Diagram

- Bank with many branches as show previously in ERD
  - Note notation for the key
  - Note the precise notation for the attributes (camelback)
  - Note the multiplicity notation



Domain Model Class Diagram

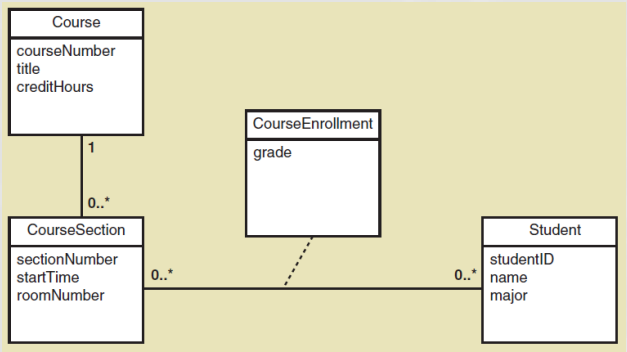
- Course Enrollment at a University
- A Course has many CourseSections
- A CourseSection has many Students and a Student is registered in many CourseSections
- Problem
  - How/where to capture student grades?



Refined Course Enrollment Model

with an Association Class CourseEnrollment

- **Association class**— an association that is treated as a class in a many to many association because it has attributes that need to be remembered (such as grade)

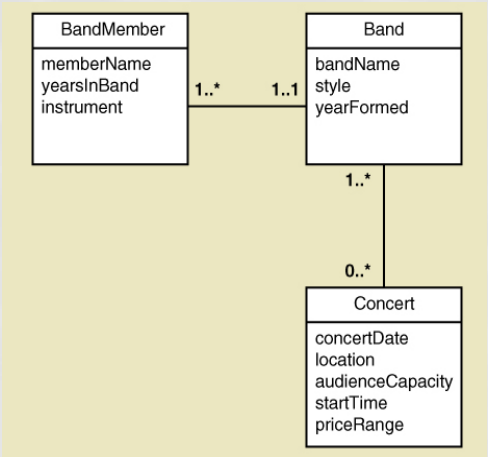


Association Class Properties

- The association class **is** the same "thing" as the association itself
- The unique identifier (key) for the association class is the concatenation of the keys of the attached classes
  - In the previous example the key for CourseSection is CourseNumber+SectionNumber
  - Hence the key for CourseEnrollment is CourseNumber+SectionNumber+StudentID
  - Note: If more information is required to uniquely identify instances of the association class, then the model is incorrect, i.e., if the key cannot be formed by the concatenation of the endpoint keys, it is in error.

Band with members and concerts

- Quick Quiz
  - How many bands can a person play in?
  - For a band, how many concerts can it play in?
  - For a concert, how many bands may be playing?
  - What attributes can you use for keys? Do you need to add "key" attributes?



Band with Concert Booking Information

- Note: The association class (Booking) also provides a name and meaning for the association
- Given the keys you identified, what is the key for the Booking class? Does it uniquely identify instances?