Object Oriented Modeling

Using Modified Modeling Language

UML

Overview

- The objective of requirements definition is understanding – understanding the users' needs, the business processes, and the systems to support business processes
- Understand and define requirements for a new system using object-oriented analysis models and techniques
- Line between object-oriented analysis and object-oriented design is somewhat fuzzy غامض
  - Iterative approach to development
  - Models built in analysis are refined during design

Object-Oriented Requirements

- Object-oriented modeling notation is Unified Modeling Language (UML 2.0)
- UML was accepted by Object Management Group (OMG) as standard modeling technique
- Purpose of Object Management Group
  - Promote theory and practice of object-oriented technology for development of distributed systems
  - Provide common architectural framework for OO

Object-Oriented Requirements (continued)

- Object-oriented system requirements are specified and documented through process of building models
- Modeling process starts with identification of use cases and problem domain classes (things in users' work environment)
- Business events trigger elementary business processes (EBP) that new system must address as use cases
- Use cases define functional requirements

Object-Oriented Requirements Models

- Use case model – a collection of models to capture system requirements
- Use case diagram – identify actors and their roles and how the actor roles utilize the system
- Systems sequence diagrams (SSDs) – define inputs and outputs and sequence of interactions between user and system for a use case
- Activity Diagram – Used to document workflow of business processes within a use case
- Domain model – describes the classes of objects and their states
- State machine diagrams – describe states of each object

Requirements Models—Traditional vs OO

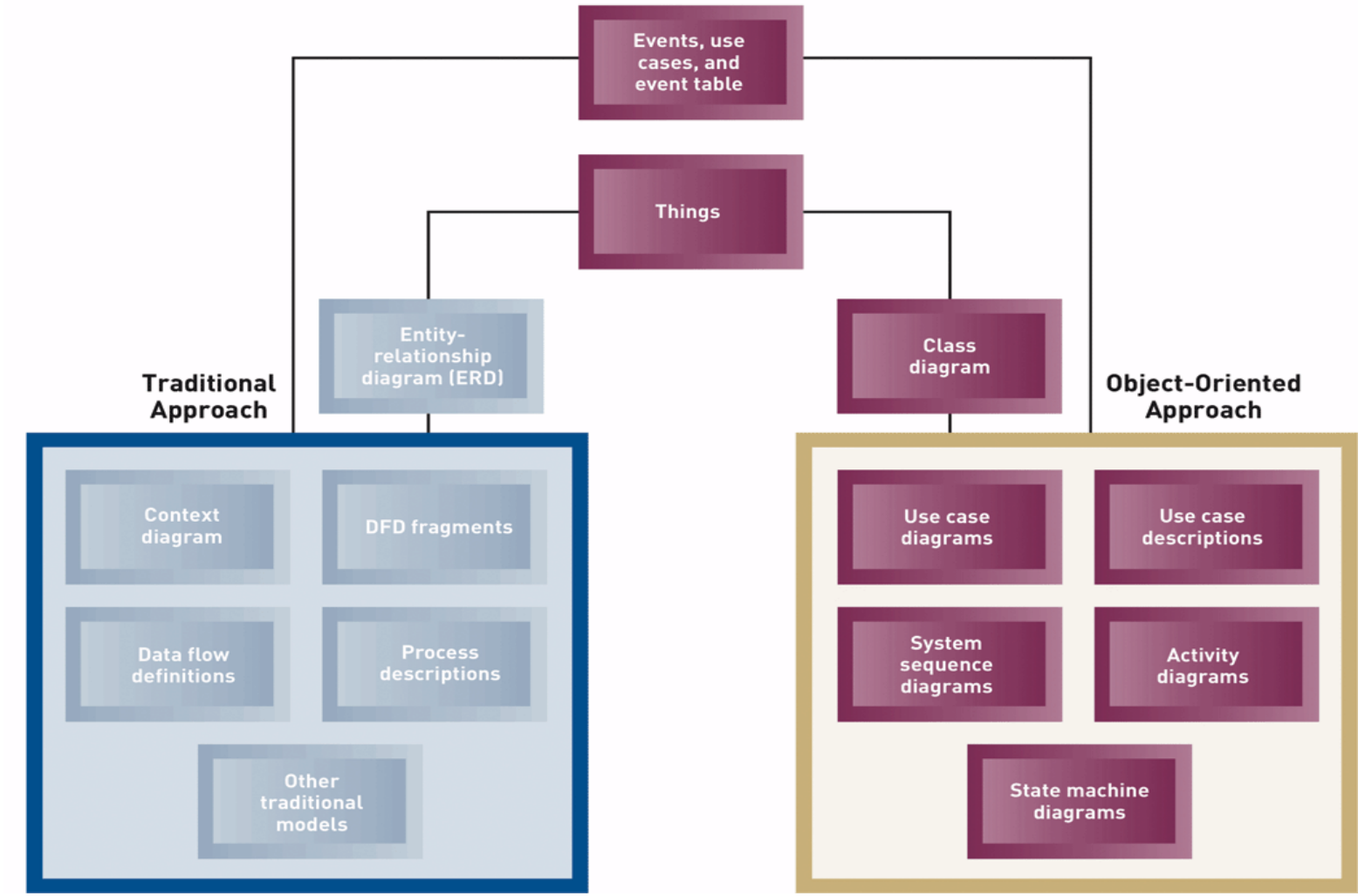


Figure 7-1

The System Activities—
A Use Case/Scenario View

- Use case analysis used to identify and define all business processes that system must support
- Use case – an activity a system carried out, usually in response to a user request
- Actor
  - Role played by user
  - Outside automation boundary

Techniques for Identifying Use Cases

- Identify user goals
  - Each goal at the elementary business process (EBP) level is a ***use case***
  - EBP – task performed by one user in one place and in response to business event that adds measurable business value, and leaves system and data in consistent state
- Event decomposition technique (event table)
- CRUD analysis technique (create, read/report, update, delete) to ensure coverage

Events and Use Cases

- Event Table – a catalog of use cases listed by event. Contains detailed information
  - Trigger – a signal that indicates an event has occurred
  - Source – an external agent that initiates event and supplies data for the event
  - Response – an output produced by the system
  - Destination – an external agent that receives the response

Information about Each Event in an Event Table

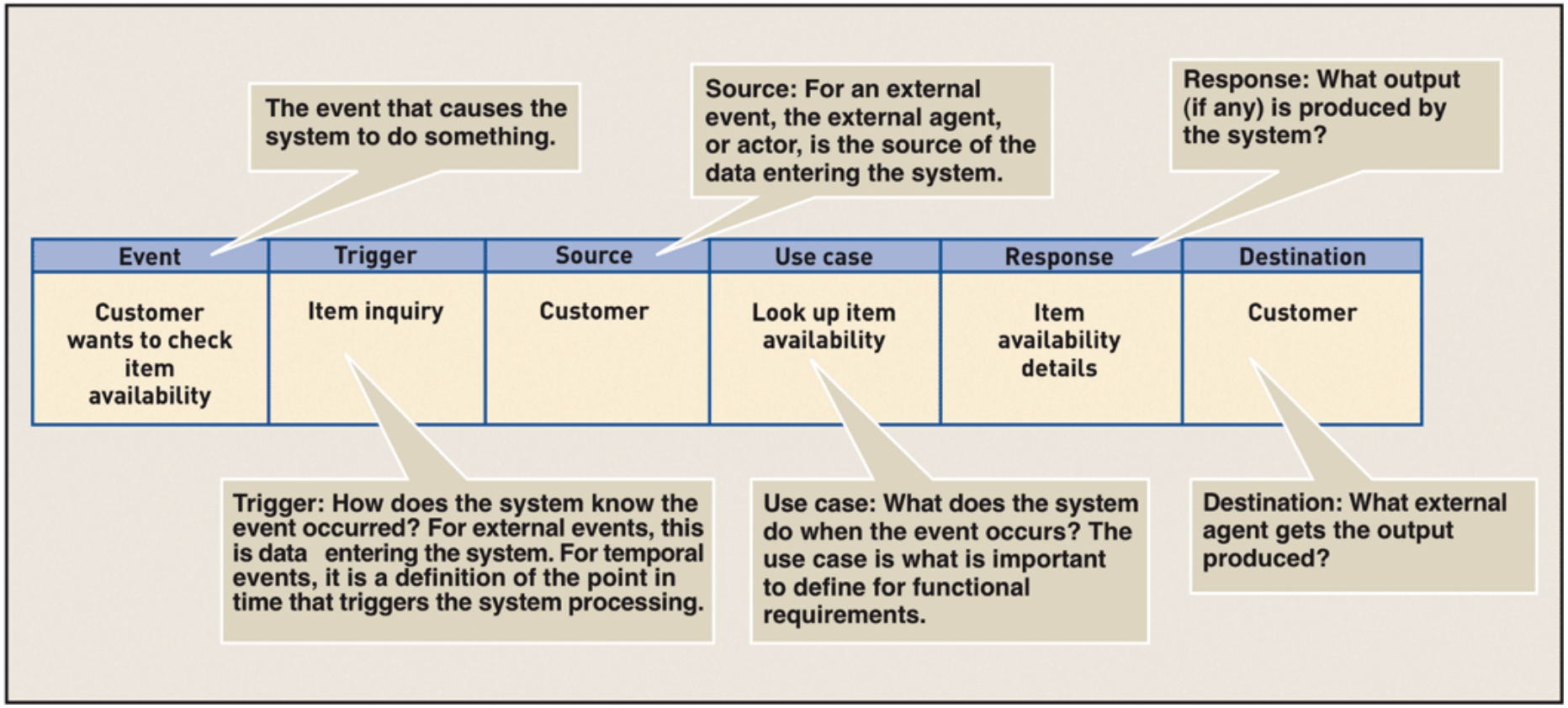


Figure 5-11

RMO Event Table

Customer support system event table					
Event	**Trigger**	**Source**	**Use case**	**Response**	**Destination**
1. Customer wants to check item availability	Item inquiry	Customer	Look up item availability	Item availability details	Customer
2. Customer places an order	New order	Customer	Create new order	Real-time link Order confirmation Order details Transaction	Credit bureau Customer Shipping Bank
3. Customer changes or cancels order	Order change request	Customer	Update order	Change confirmation Order change details Transaction	Customer Shipping Bank
4. Time to produce order summary reports	"End of week, month, quarter and year"		Produce order summary reports	Order summary reports	Management
5. Time to produce transaction summary reports	"End of day"		Produce transaction summary reports	Transaction summary reports	Accounting
6. Customer or management wants to check order status	Order status inquiry	Customer or management	Look up order status	Order status details	Customer or management

Figure 5-12

DFD Integrates Event Table and ERD

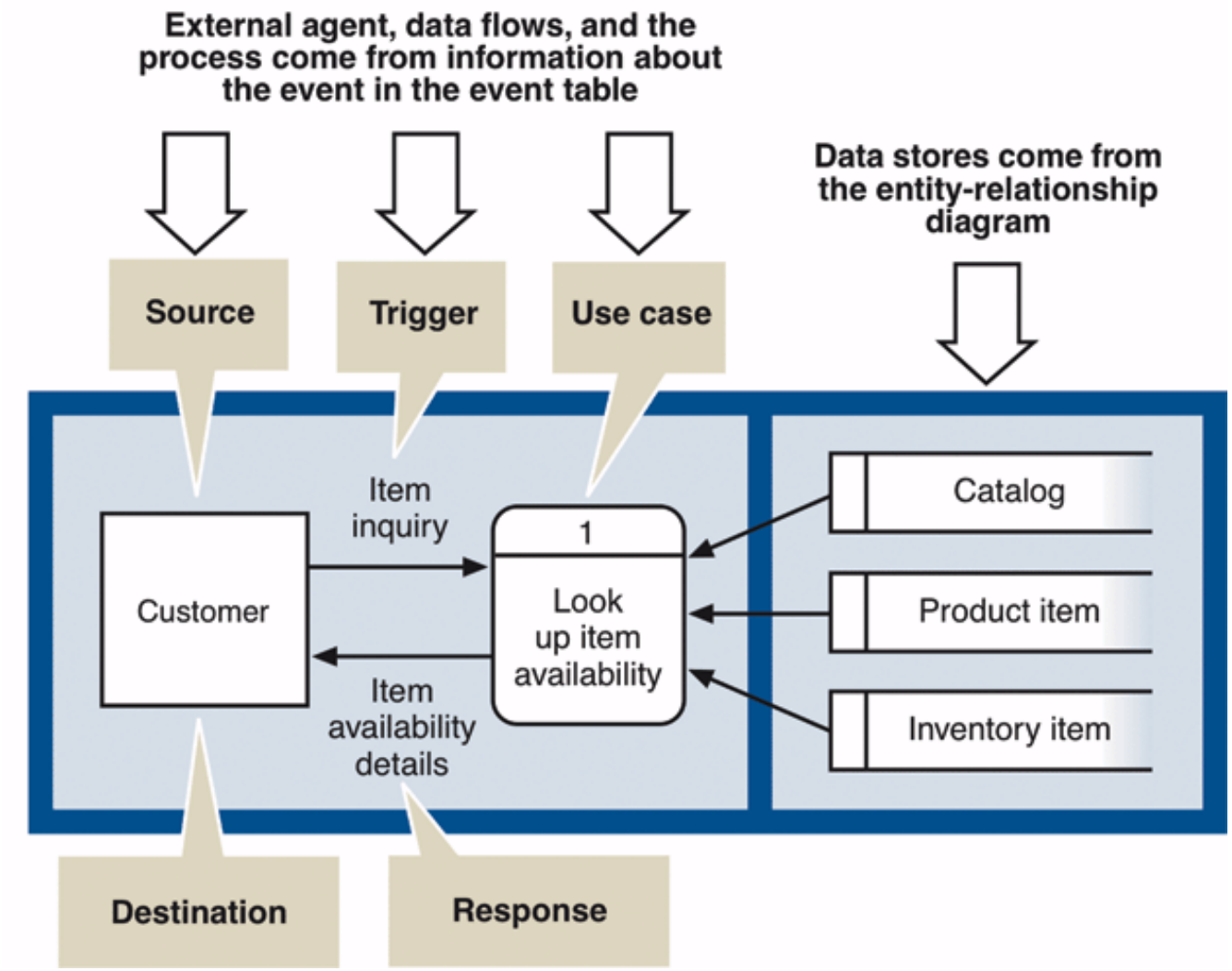


Figure 6-5

Use Case Diagram with Automation Boundary and Alternate Actor Notation

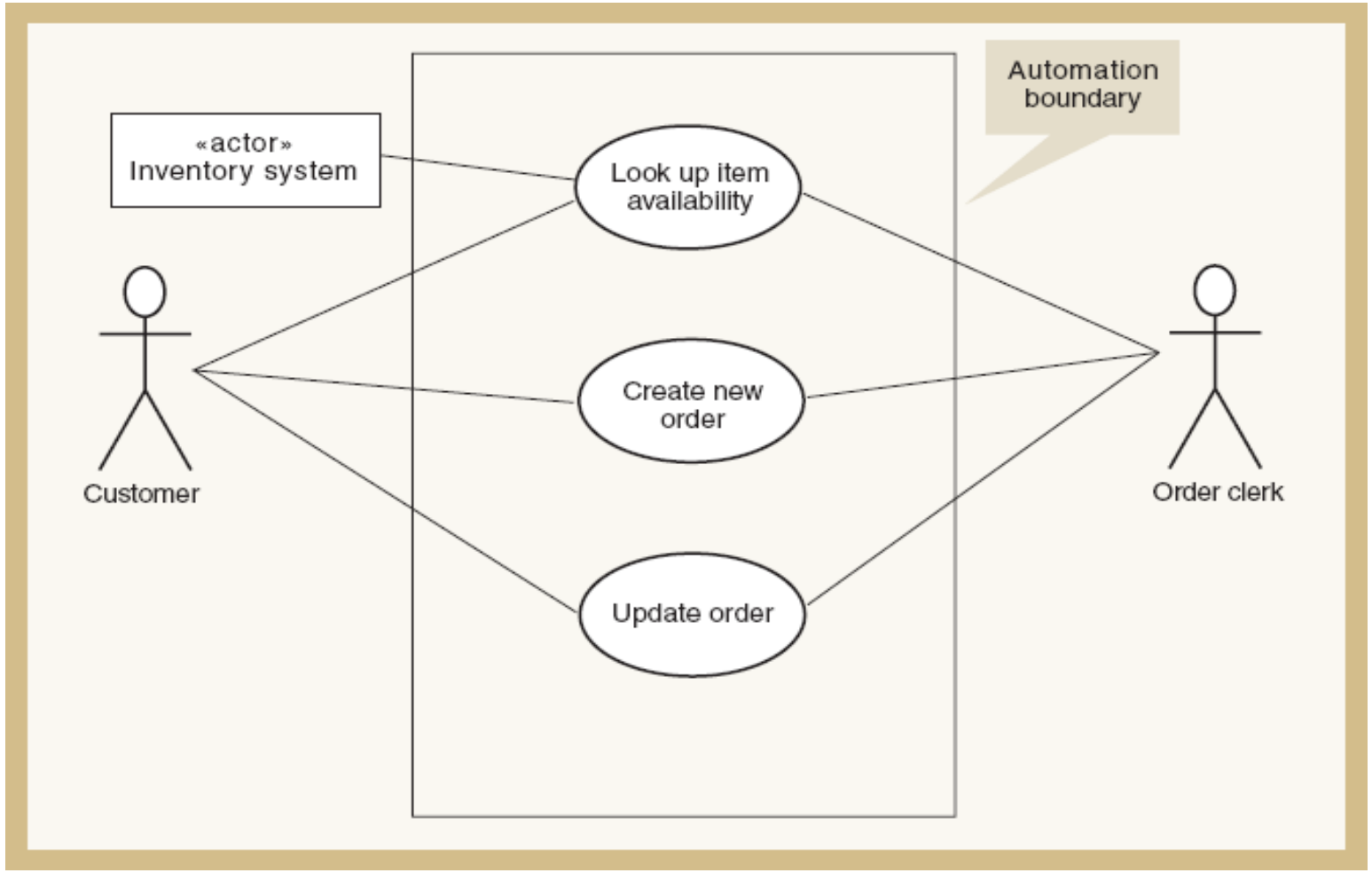


Figure 7-3

Event Decomposition Technique: Specific Steps

1. Consider the external events in the system environment that require a response from the system by using the checklist shown in Figure 3-3
2. For each external event, identify and name the use case that the system requires
3. Consider the temporal events that require a response from the system by using the checklist shown in Figure 3-4
4. For each temporal event, identify and name the use case that the system requires and then establish the point of time that will trigger the use case

Event Decomposition Technique: Specific Steps (continued)

5. Consider the state events that the system might respond to, particularly if it is a real-time system in which devices or internal state changes trigger use cases.
6. For each state event, identify and name the use case that the system requires and then define the state change.
7. When events and use cases are defined, check to see if they are required by using the perfect technology assumption. Do not include events that involve such system controls as login, logout, change password, and backup or restore the database, as these are put in later.

Event Decomposition Technique: Benefits

- Events are broader than user goal: Capture temporal and state events
- Help decompose at the right level of analysis: an elementary business process (EBP)
- EBP is a fundamental business process performed by one person, in one place, in response to a business event
- Uses perfect technology assumption to make sure functions that support the users work are identified and not additional functions for security and system controls

Use Case Diagram

- Graphical UML diagram that summarizes information about actors and use cases
- Simple diagram shows overview of functional requirements
- Can have multiple use case diagrams
  - By subsystem
  - By actor

Simple Use Case with an Actor

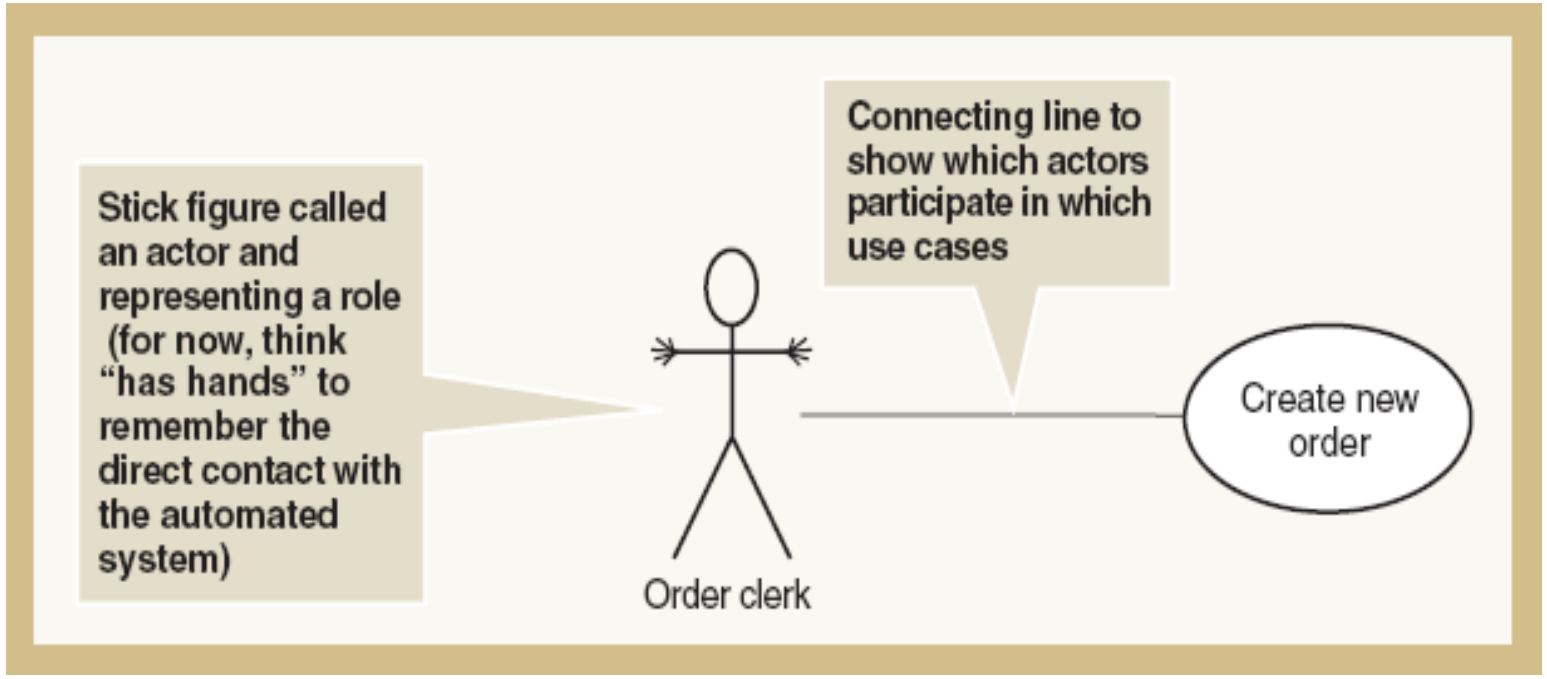


Figure 7-2

Use Case Diagram with Automation Boundary and Alternate Actor Notation

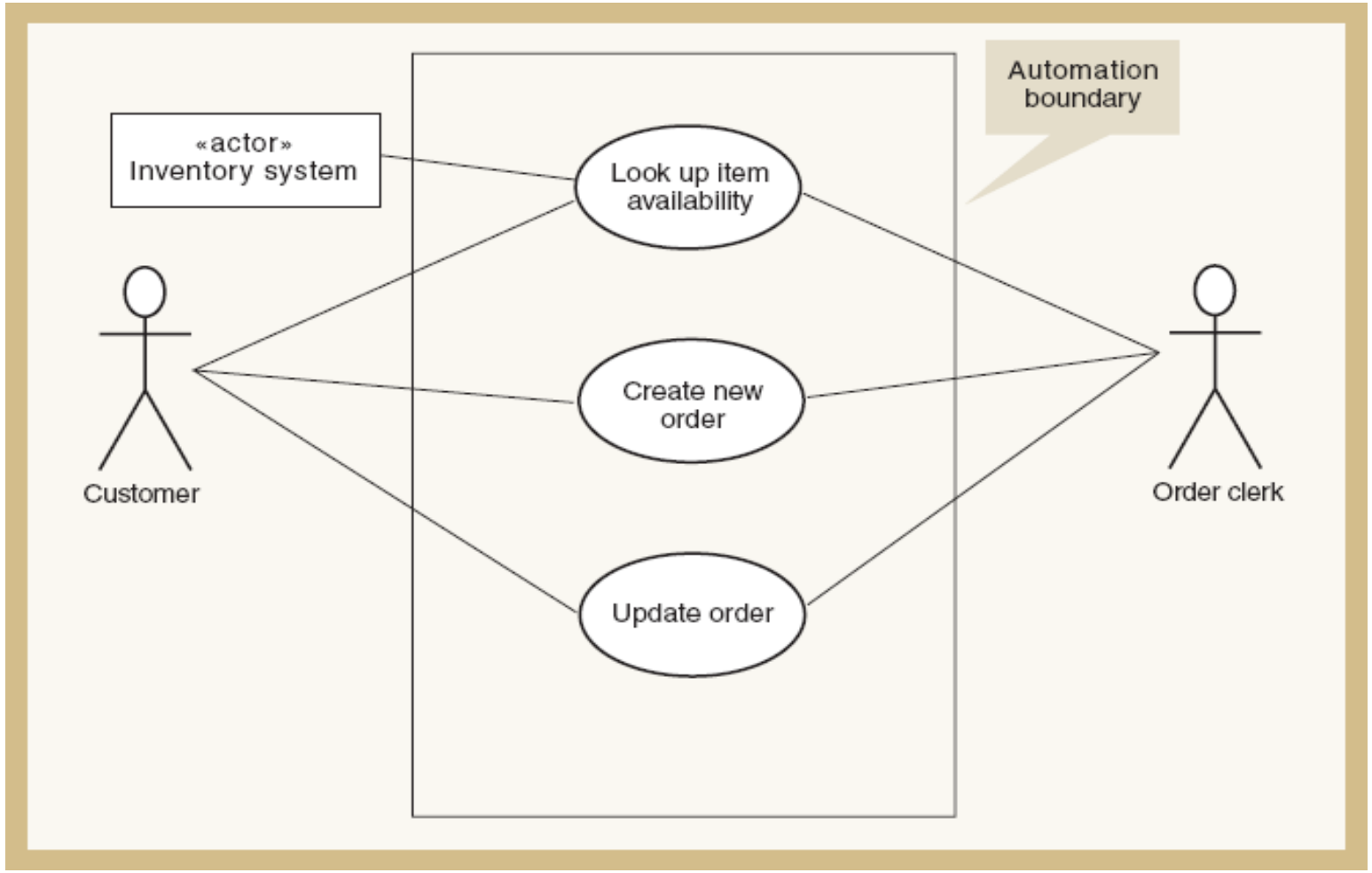


Figure 7-3

Use Cases of RMO Order Entry Subsystem

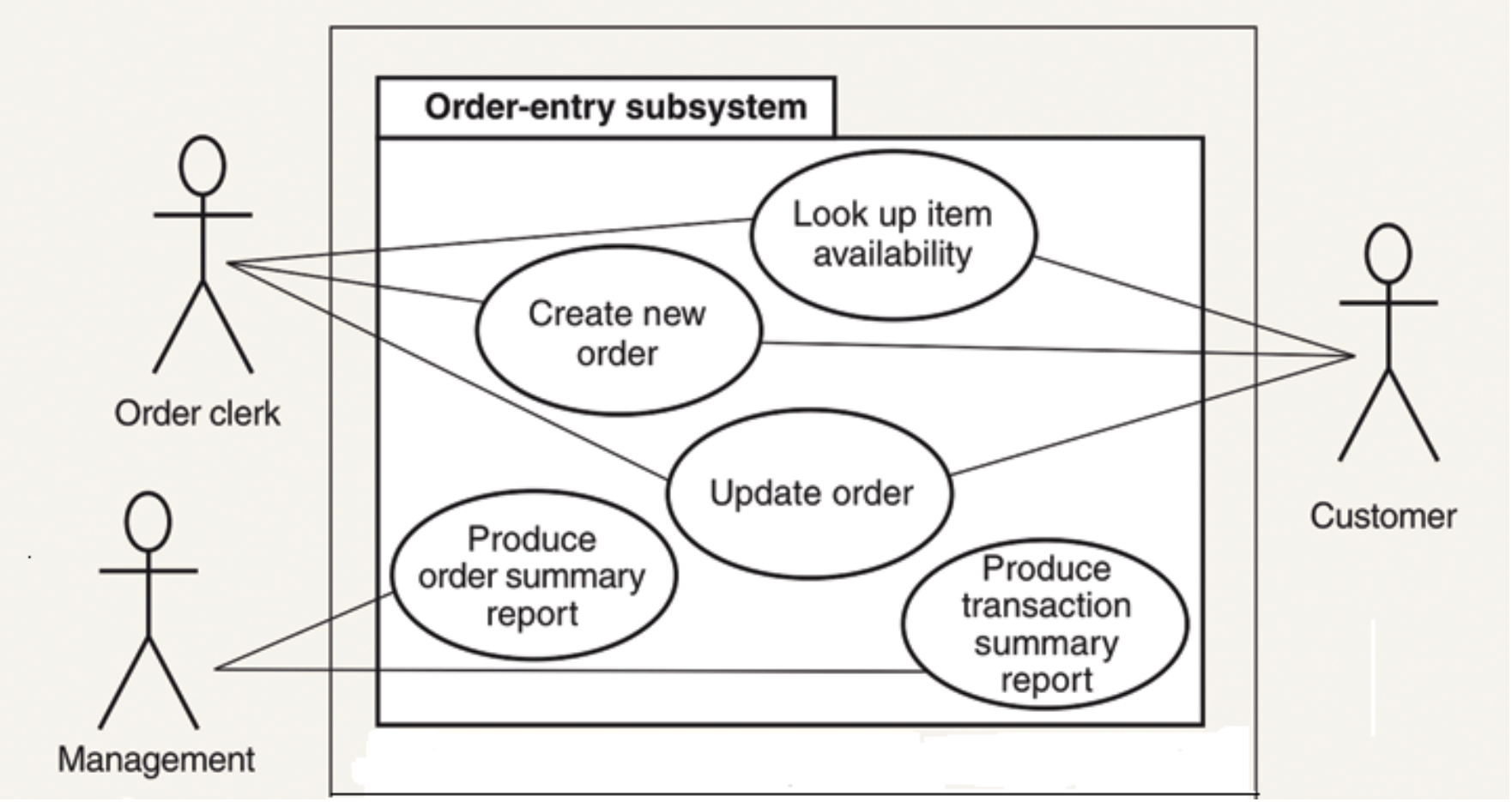
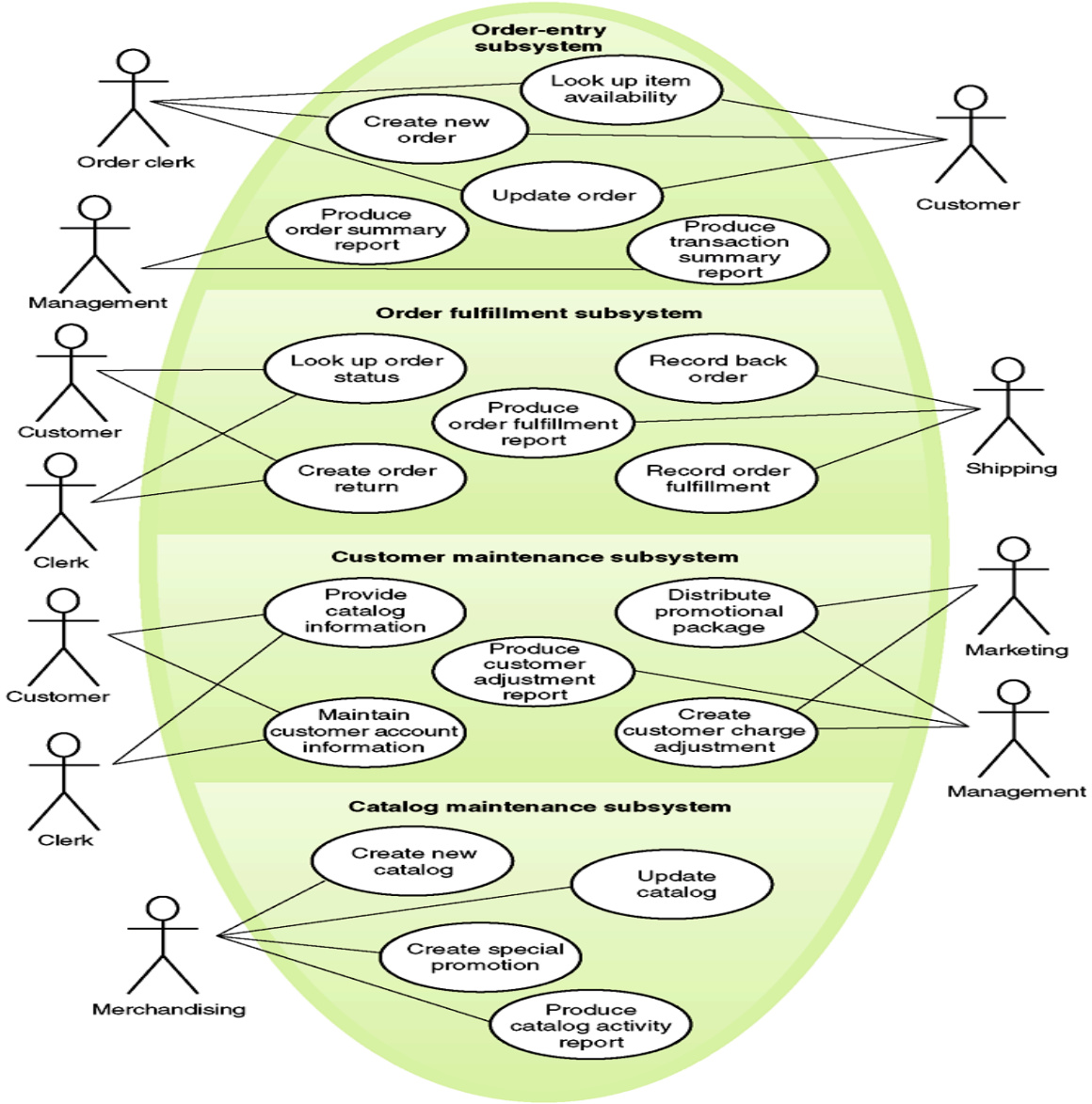


Figure 7-5 (partial figure)

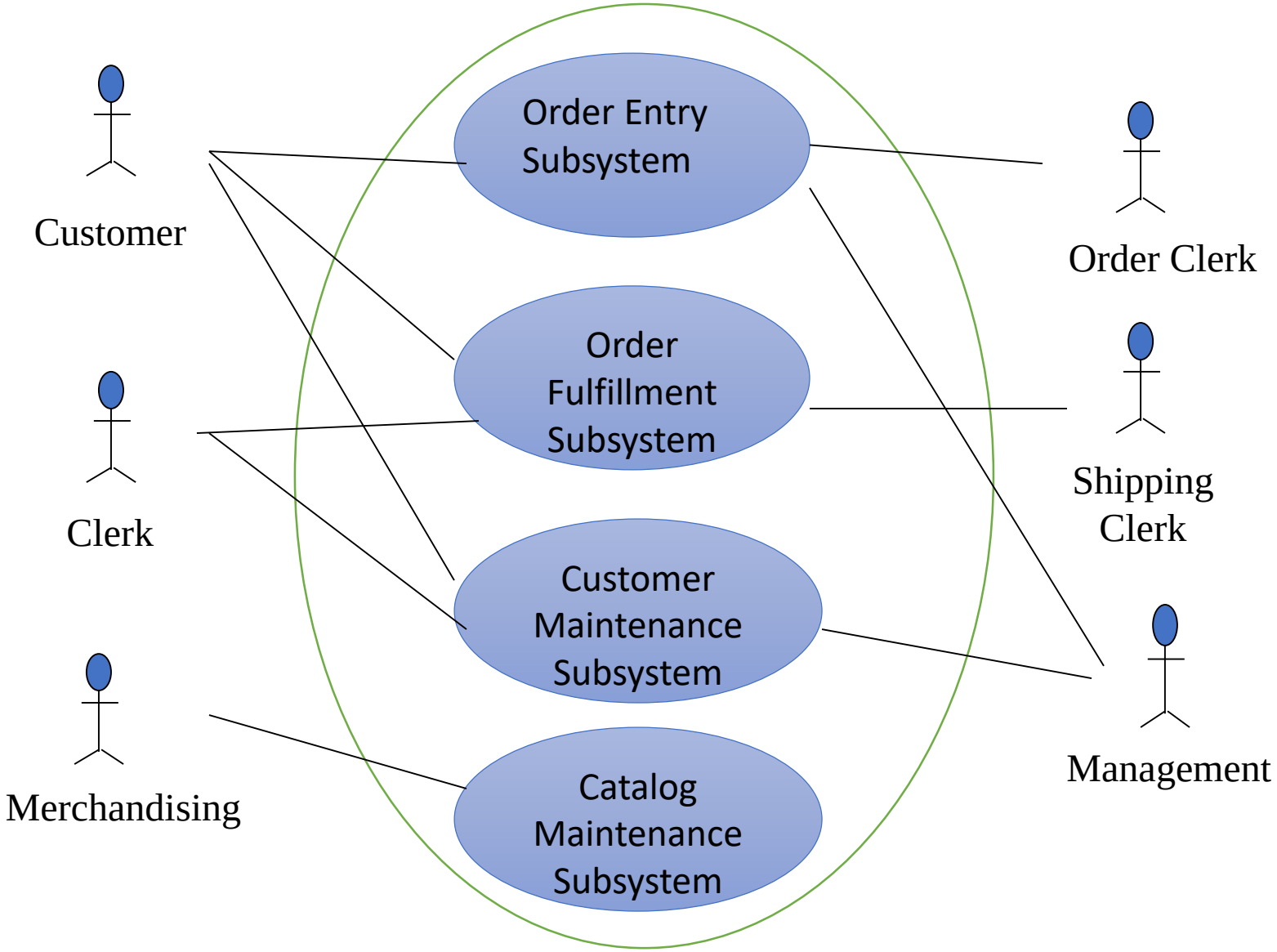
Use Case of Customer Support System

FIGURE 7-4

A use case diagram of the customer support system (by subsystem).



Use Case of Customer Support System



All Use Cases Involving Customer as Actor

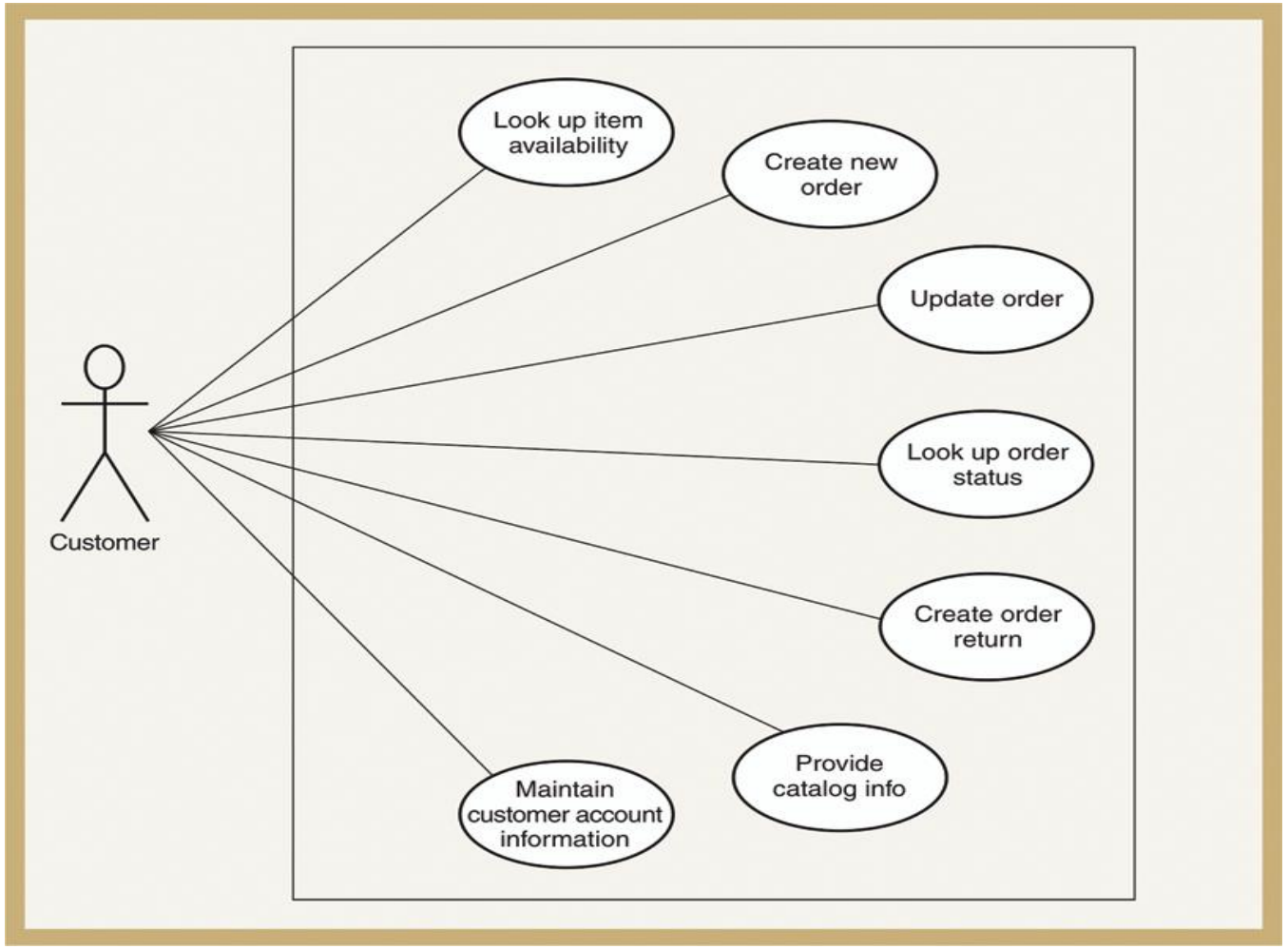
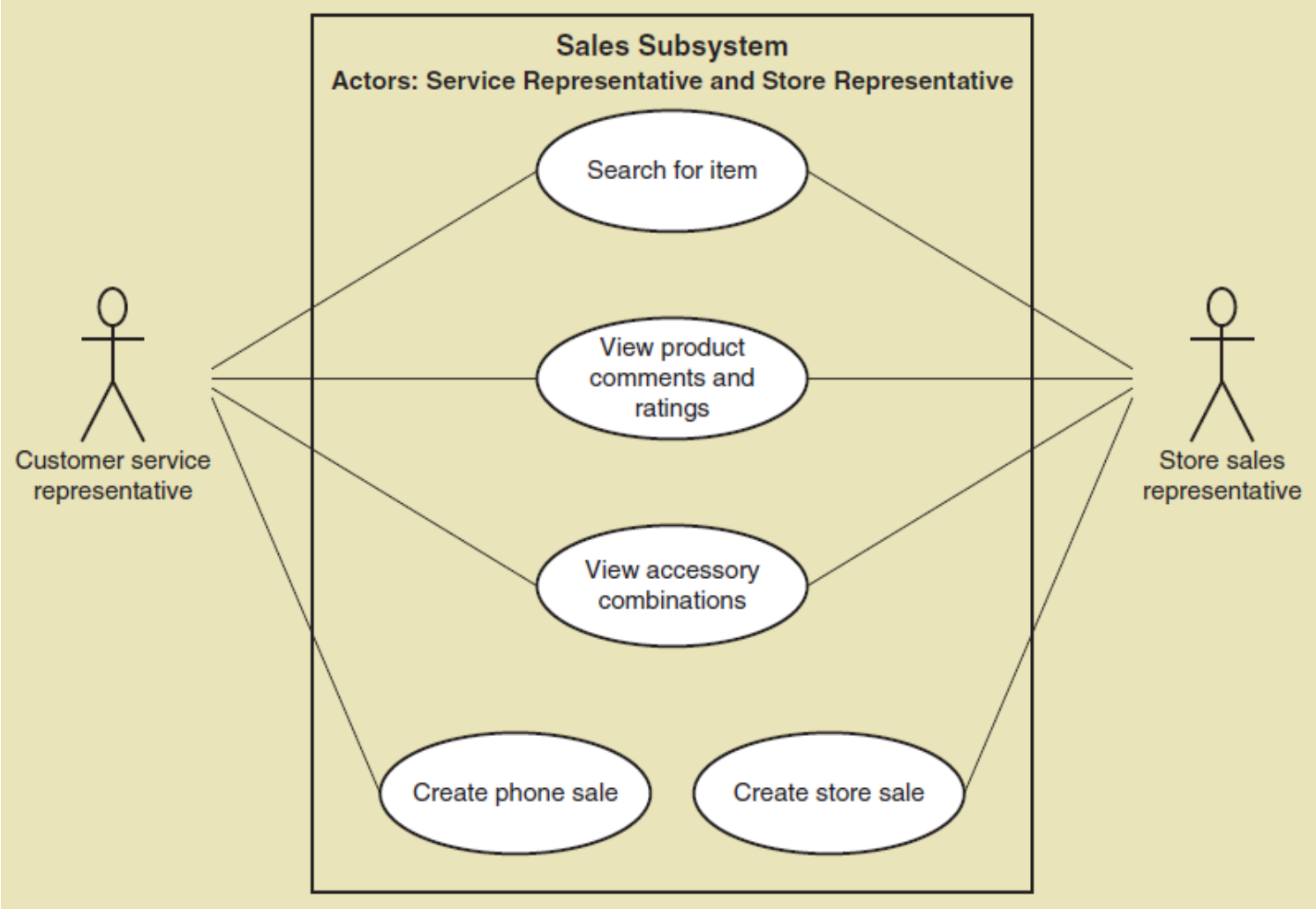


Figure 7-4

Use Case Diagrams

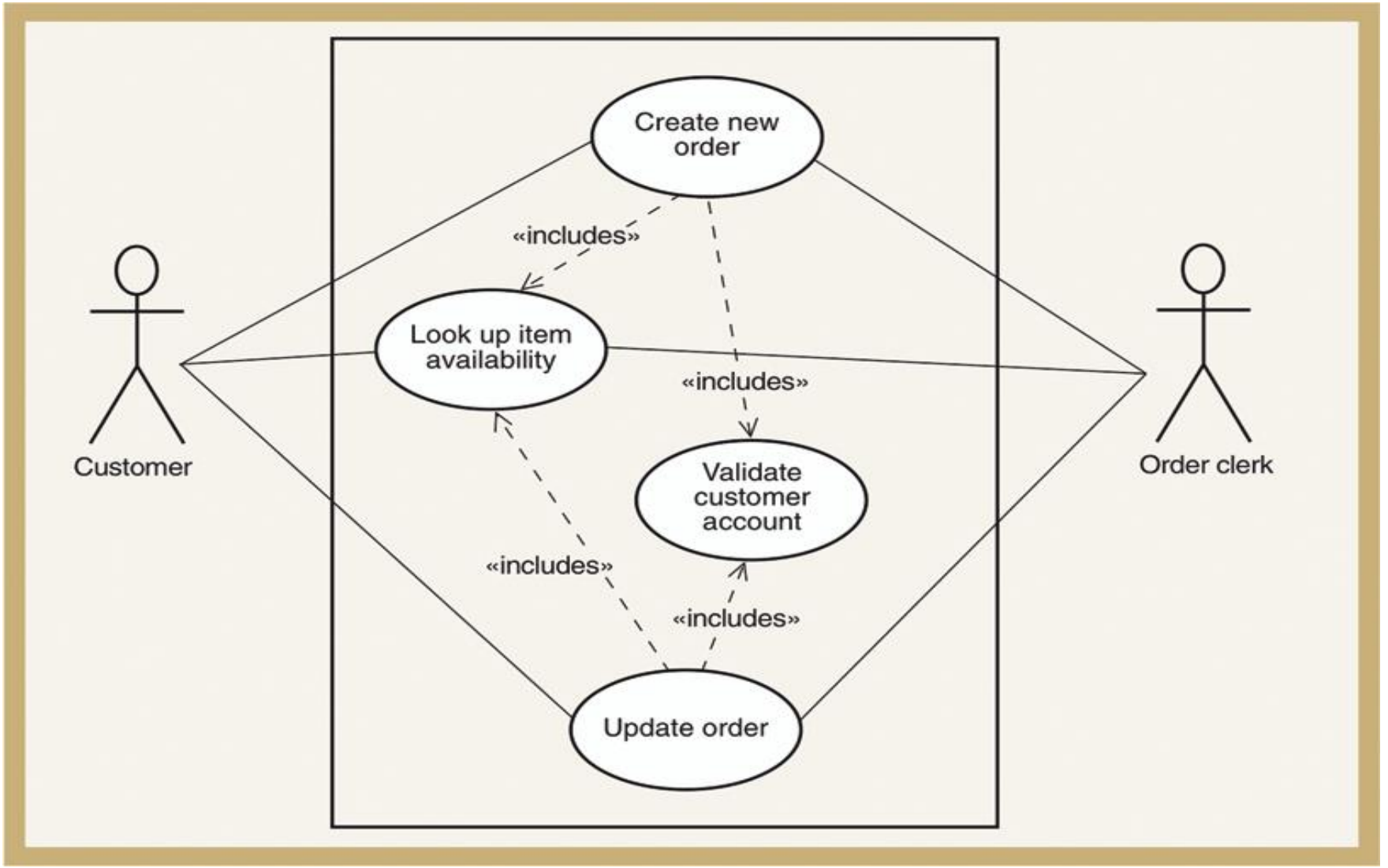
Draw for internal RMO actors



<<Includes>> Relationship

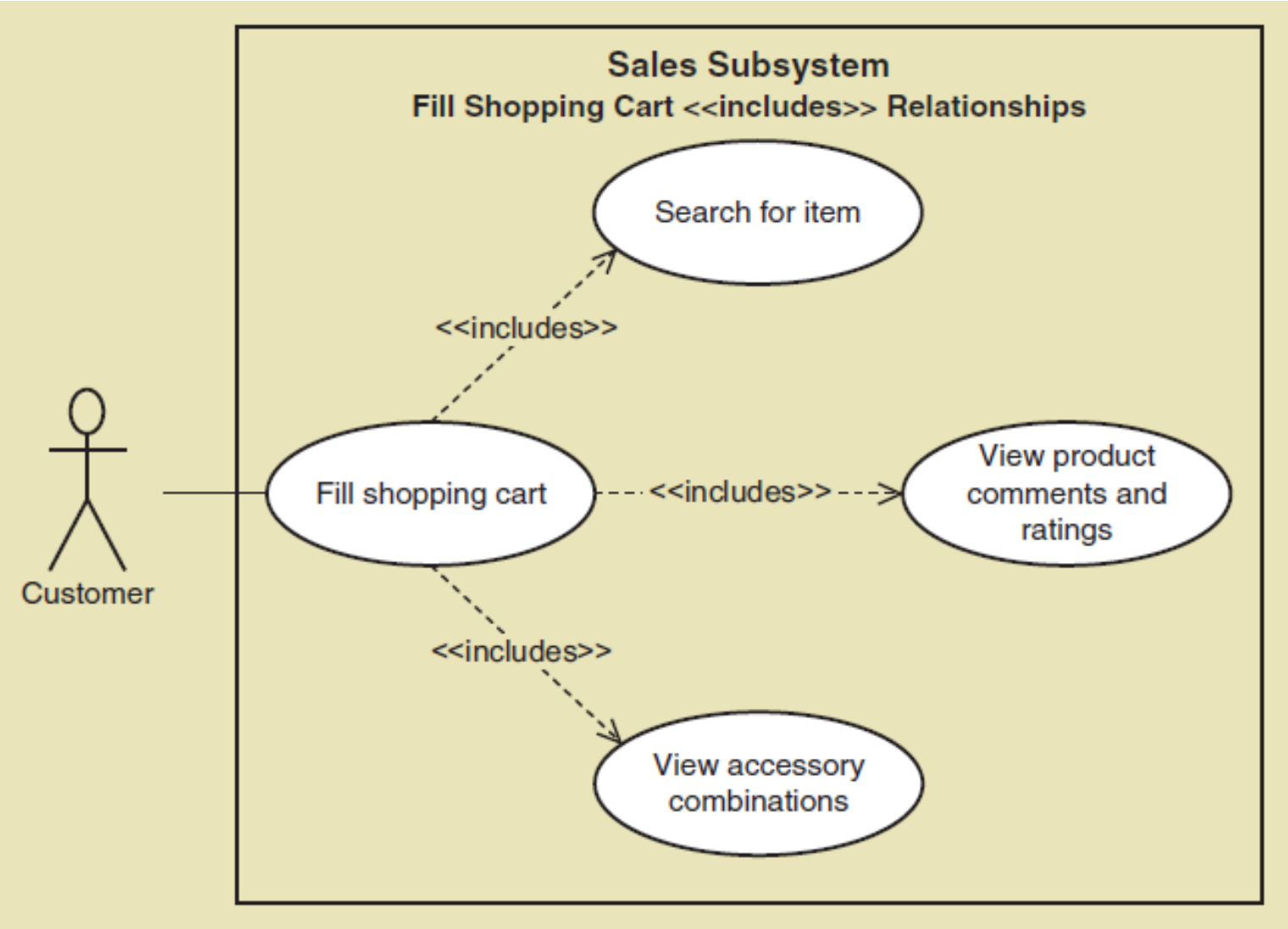
- Documents situation in which one use case requires the services of a common subroutine
- Another use case is developed for this common subroutine
- A common use case can be reused by multiple use cases

Example of Order-Entry Subsystem with <<Includes>> Use Cases (Ex. 1)



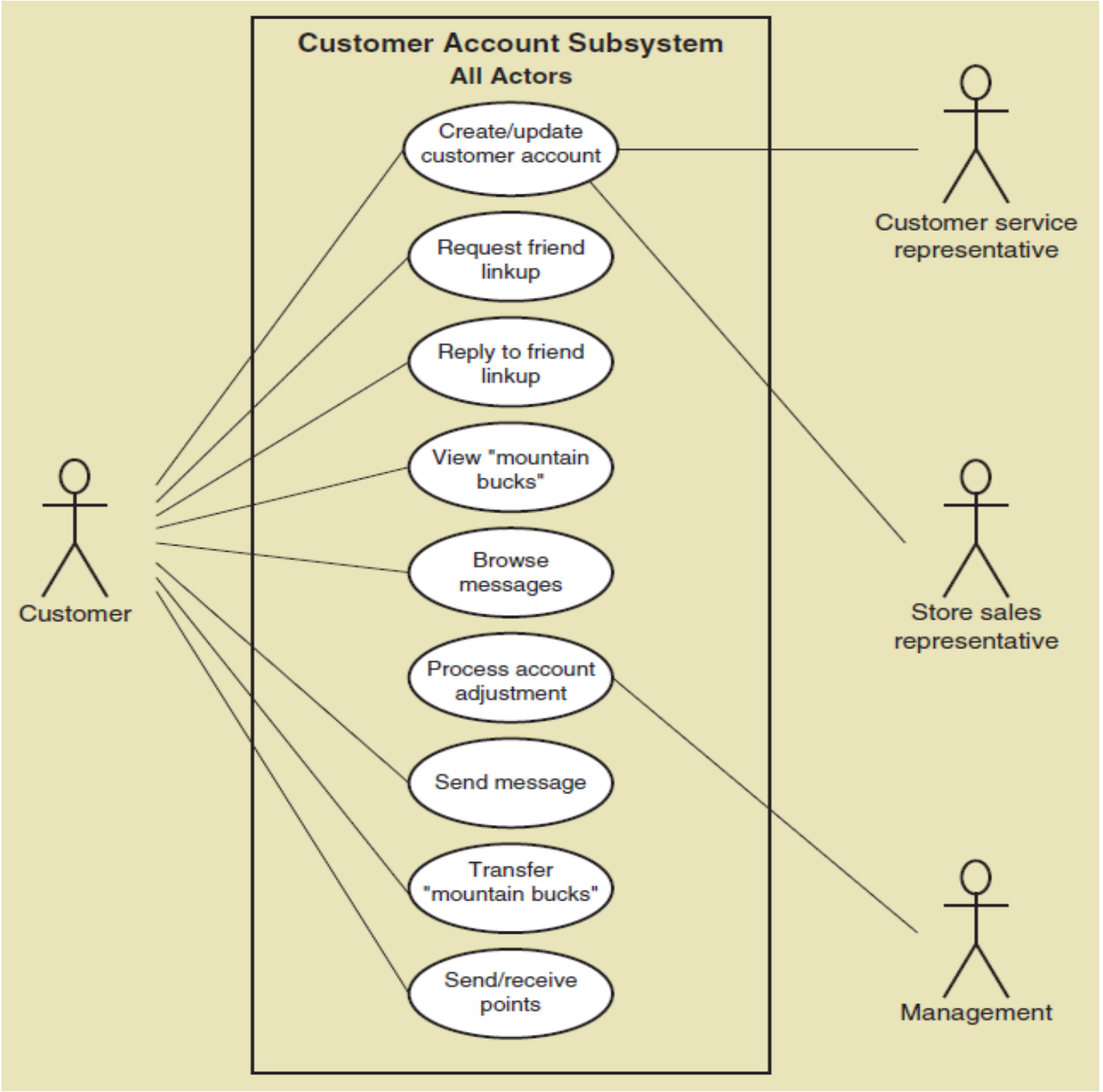
Use Case Diagrams

The <<Includes>> relationship (Ex. 2)



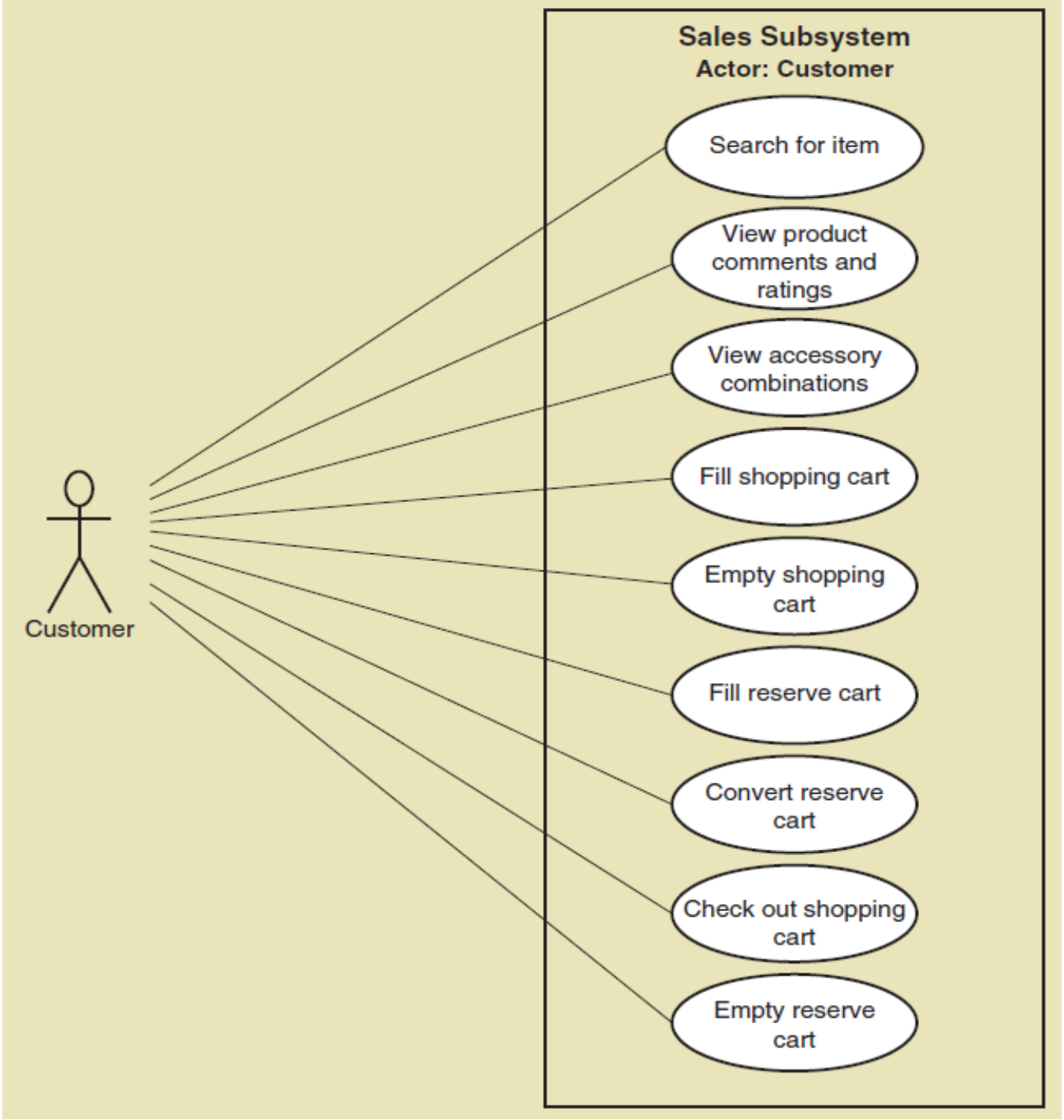
Use Case Diagrams

Draw for each subsystem



Use Case Diagrams

Draw for actor, such as customer



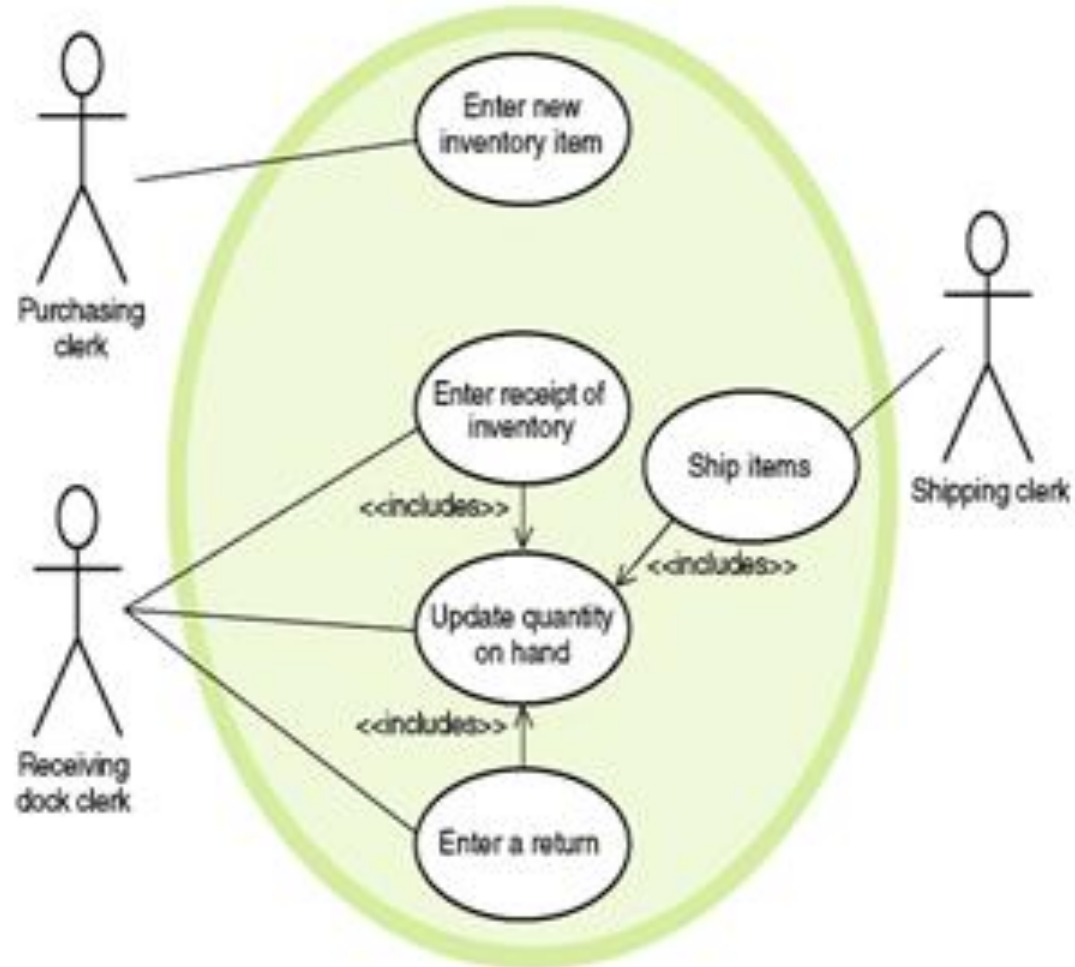
Developing a Use Case Diagram

- Underlying conditions for describing use cases
  - Based on automated system, e.g. users "touch" the system
  - Assume perfect technology condition
- Iterate through these two steps
  - Identify actors as roles
  - List goals, e.g. use cases, for each actor. A goal is a unit of work.
- Finalize with a CRUD analysis to ensure completeness

Use Case Diagram for Inventory System

FIGURE 7-22

A use case diagram for the inventory system.



Use Cases and CRUD Technique

- CRUD is Create, Read/Report, Update, and Delete (archive)
- Often introduced in database context
- Technique to validate, refine or cross-check use cases
- NOT for primarily identifying use cases

Use Cases and CRUD Technique

- For Customer domain class, verify that there are use cases that create, read/report, update, and delete (archive) the domain class

Data entity/domain class	CRUD	Verified use case
Customer	Create	Create customer account
	Read/report	Look up customer Produce customer usage report
	Update	Process account adjustment Update customer account
	Delete	Update customer account (to archive)

CRUD Technique Steps

1. Identify all the data entities or domain classes involved in the new system. (more in Chapter 4)
2. For each type of data (data entity or domain class), verify that a use case has been identified that creates a new instance, updates existing instances, reads or reports values of instances, and deletes (archives) an instance.
3. If a needed use case has been overlooked, add a new use case and then identify the stakeholders.
4. With integrated applications, make sure it is clear which application is responsible for adding and maintaining the data and which system merely uses the data.

CRUD Technique
Use Case vs. Domain Class Table

- To summarize CRUD analysis results, create a matrix of use cases and domain classes indicating which use case C, R, U, or D a domain class

Use case vs. entity/domain class	Customer	Account	Sale	Adjustment
Create customer account	C	C		
Look up customer	R	R		
Produce customer usage report	R	R	R	
Process account adjustment	R	U	R	C
Update customer account	UD (archive)	UD (archive)		