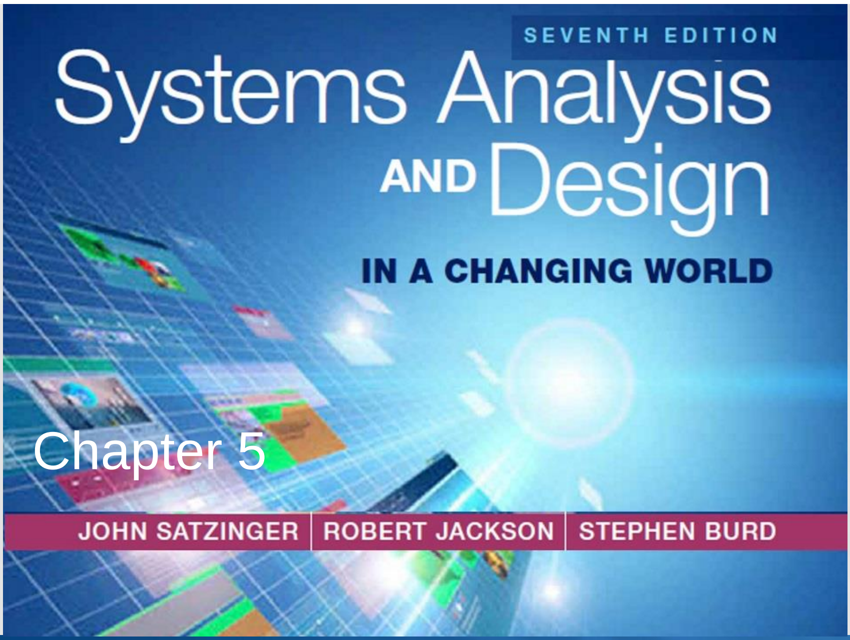


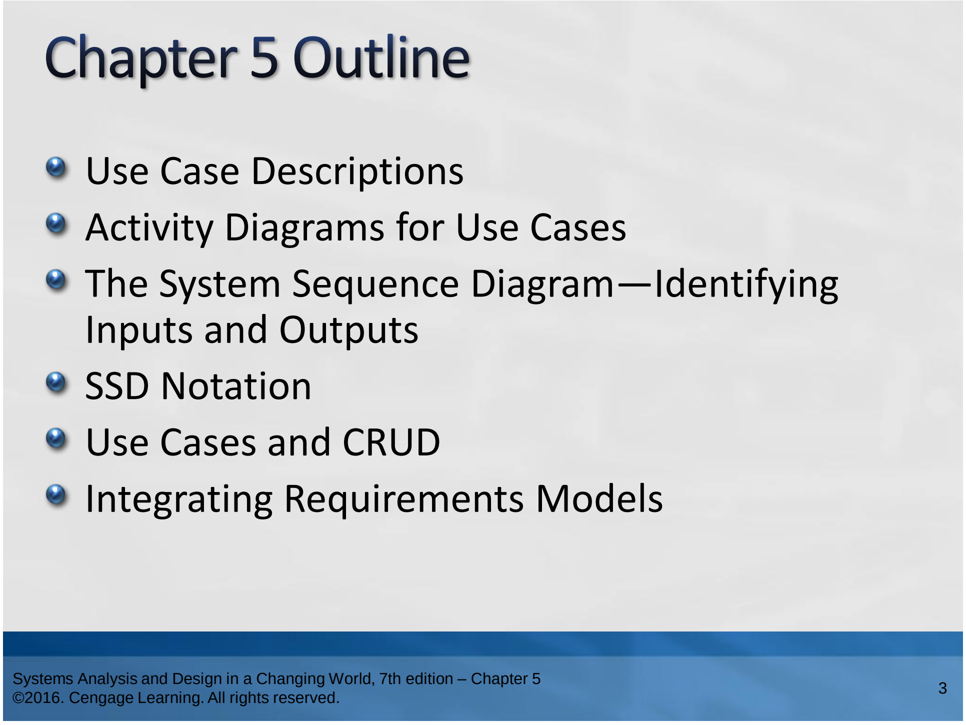
Chapter 5

Use Case Modeling

Chapter 5

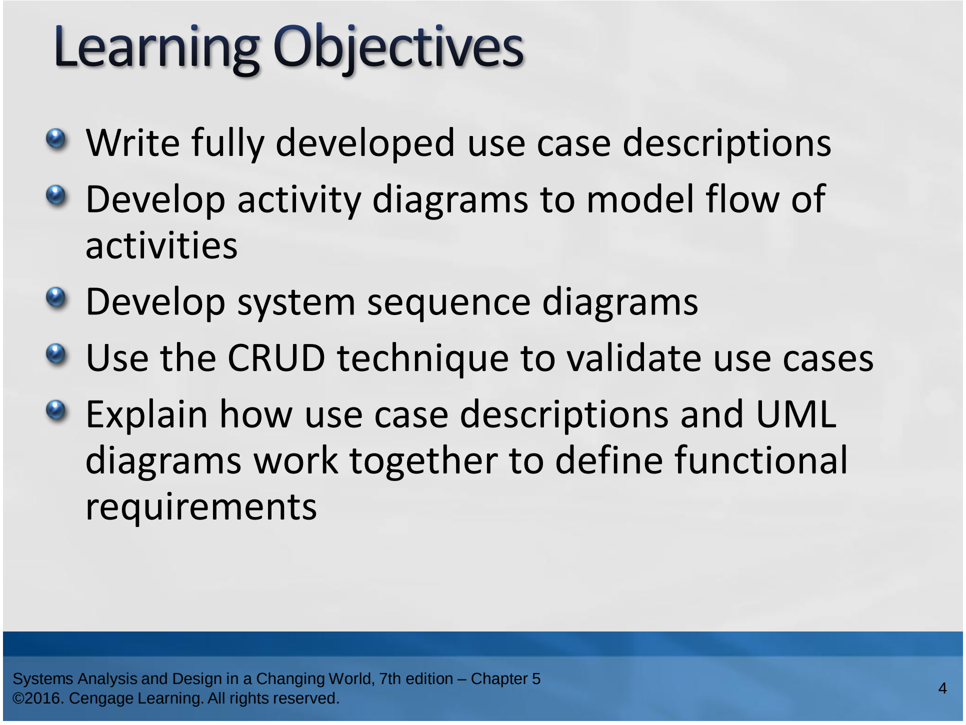
Systems Analysis and Design in a Changing World 7th Ed

Satzinger, Jackson & Burd



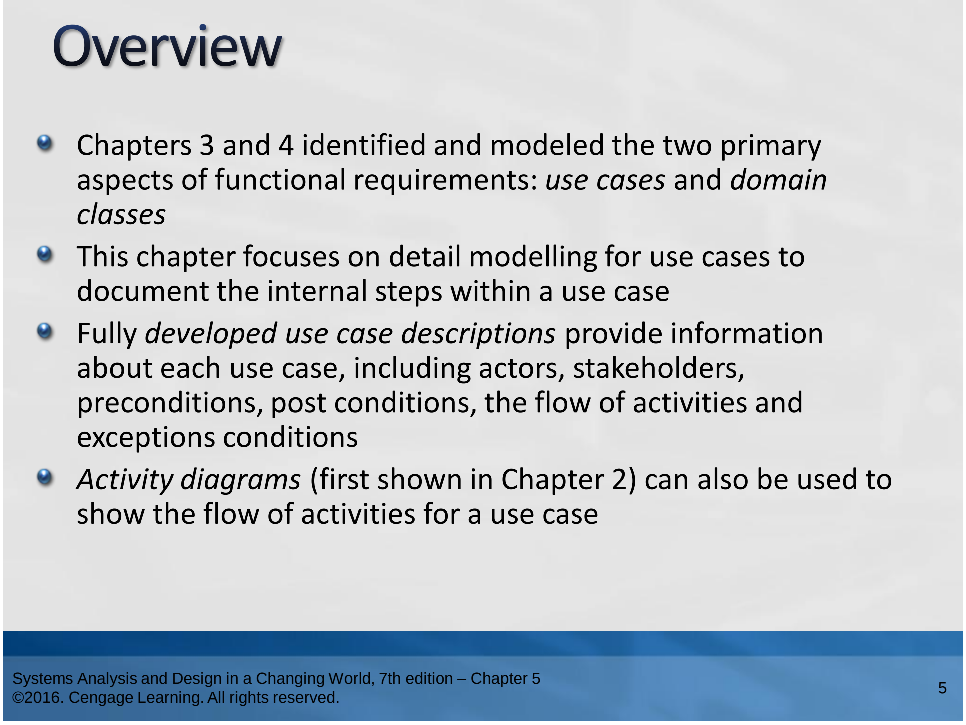
Chapter 5 Outline

- Use Case Descriptions
- Activity Diagrams for Use Cases
- The System Sequence Diagram—Identifying Inputs and Outputs
- SSD Notation
- Use Cases and CRUD
- Integrating Requirements Models



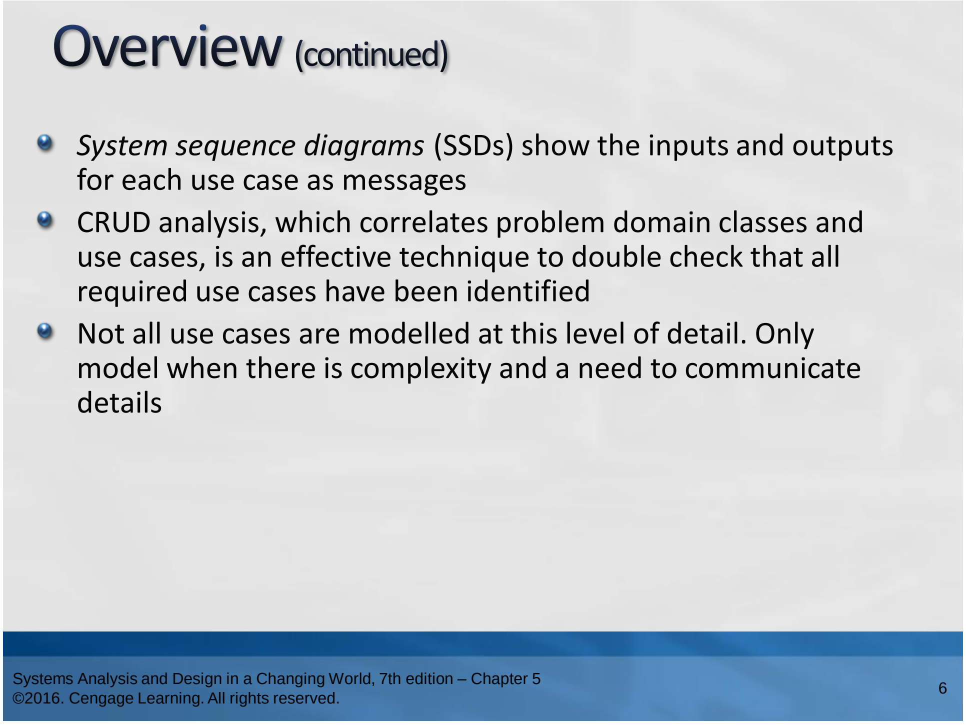
Learning Objectives

- Write fully developed use case descriptions
- Develop activity diagrams to model flow of activities
- Develop system sequence diagrams
- Use the CRUD technique to validate use cases
- Explain how use case descriptions and UML diagrams work together to define functional requirements



Overview

- Chapters 3 and 4 identified and modeled the two primary aspects of functional requirements: *use cases* and *domain classes*
- This chapter focuses on detail modelling for use cases to document the internal steps within a use case
- Fully *developed use case descriptions* provide information about each use case, including actors, stakeholders, preconditions, post conditions, the flow of activities and exceptions conditions
- *Activity diagrams* (first shown in Chapter 2) can also be used to show the flow of activities for a use case

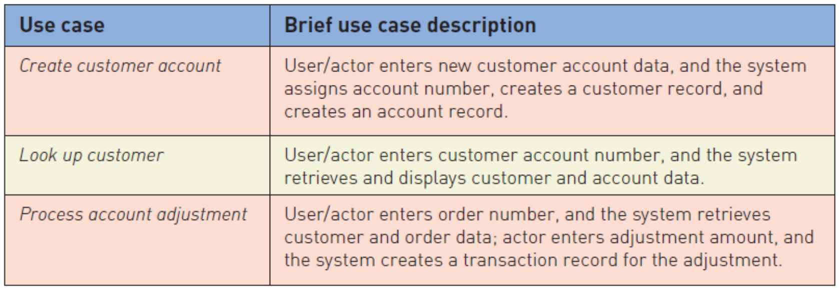


Overview (continued)

- *System sequence diagrams* (SSDs) show the inputs and outputs for each use case as messages
- CRUD analysis, which correlates problem domain classes and use cases, is an effective technique to double check that all required use cases have been identified
- Not all use cases are modelled at this level of detail. Only model when there is complexity and a need to communicate details

Use Case Descriptions

- Write a *brief description* as shown in Chapter 3 for most use cases.



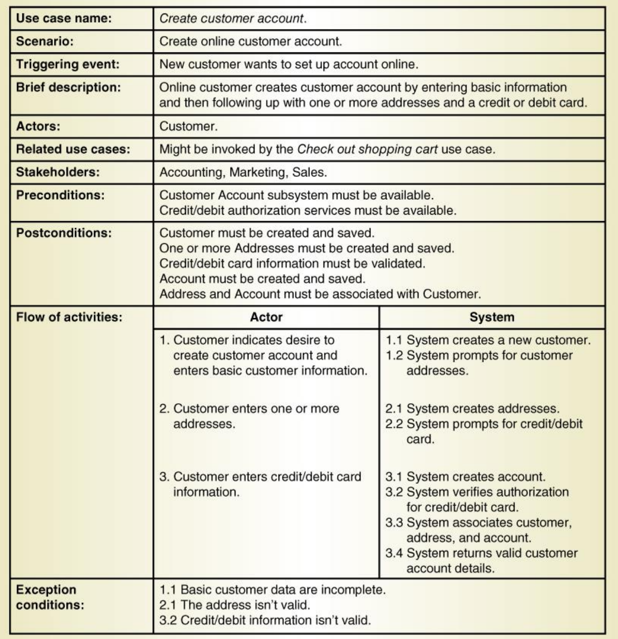
Use case	Brief use case description
Create customer account	User/actor enters new customer account data, and the system assigns account number, creates a customer record, and creates an account record.
Look up customer	User/actor enters customer account number, and the system retrieves and displays customer and account data.
Process account adjustment	User/actor enters order number, and the system retrieves customer and order data; actor enters adjustment amount, and the system creates a transaction record for the adjustment.

Use Case Descriptions

- Write a *fully developed use case description* for more complex use cases
- Typical use case description templates include:
    - Use case name
    - Scenario (if needed)
    - Triggering event
    - Brief description
    - Actors
    - Related use cases (<<includes>>)
    - Stakeholders
    - Preconditions
    - Postconditions
    - Flow of activities
    - Exception conditions

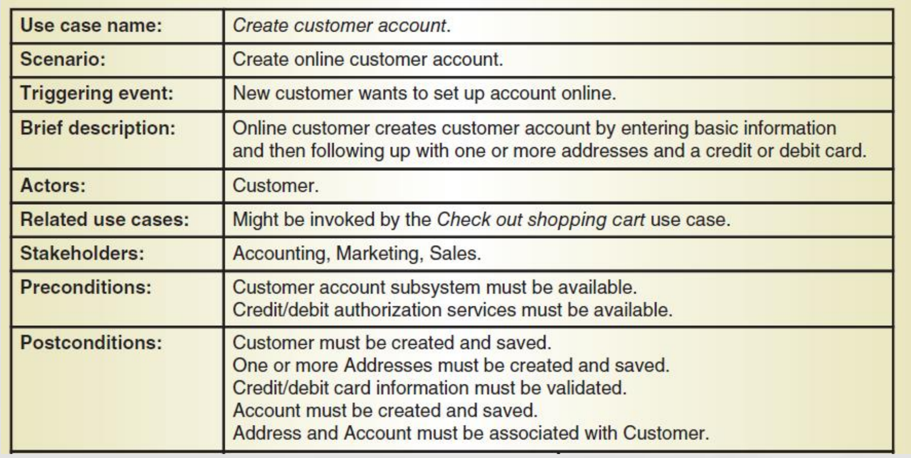
Fully Developed Use Case Description

Use case: *Create customer account*



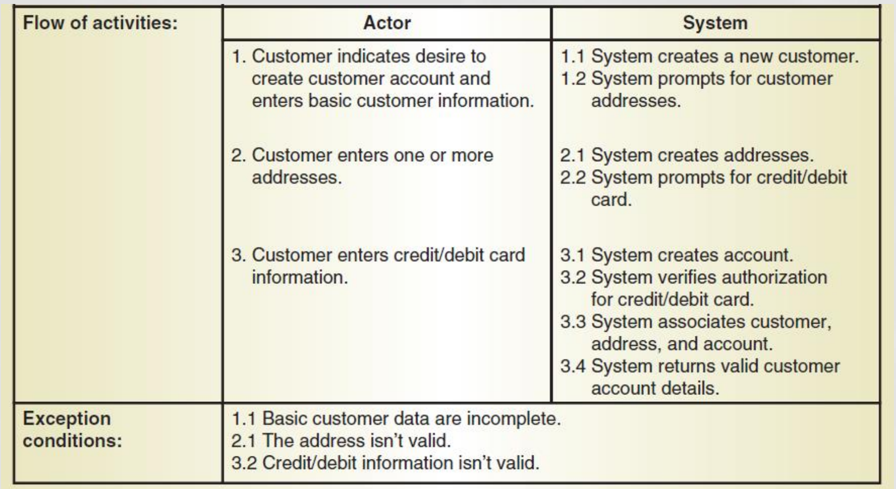
Use case name:	*Create customer account.*	
Scenario:	Create online customer account.	
Triggering event:	New customer wants to set up account online.	
Brief description:	Online customer creates customer account by entering basic information and then following up with one or more addresses and a credit or debit card.	
Actors:	Customer.	
Related use cases:	Might be invoked by the *Check out shopping cart* use case.	
Stakeholders:	Accounting, Marketing, Sales.	
Preconditions:	Customer Account subsystem must be available. Credit/debit authorization services must be available.	
Postconditions:	Customer must be created and saved. One or more Addresses must be created and saved. Credit/debit card information must be validated. Account must be created and saved. Address and Account must be associated with Customer.	
Flow of activities:	**Actor**	**System**
	1. Customer indicates desire to create customer account and enters basic customer information.	1.1 System creates a new customer. 1.2 System prompts for customer addresses.
	2. Customer enters one or more addresses.	2.1 System creates addresses. 2.2 System prompts for credit/debit card.
	3. Customer enters credit/debit card information.	3.1 System creates account. 3.2 System verifies authorization for credit/debit card. 3.3 System associates customer, address, and account. 3.4 System returns valid customer account details.
Exception conditions:	1.1 Basic customer data are incomplete. 2.1 The address isn't valid. 3.2 Credit/debit information isn't valid.	

Fully Developed Use Case Description *Create customer account* (part 1)



Use case name:	*Create customer account.*
Scenario:	Create online customer account.
Triggering event:	New customer wants to set up account online.
Brief description:	Online customer creates customer account by entering basic information and then following up with one or more addresses and a credit or debit card.
Actors:	Customer.
Related use cases:	Might be invoked by the *Check out shopping cart* use case.
Stakeholders:	Accounting, Marketing, Sales.
Preconditions:	Customer account subsystem must be available. Credit/debit authorization services must be available.
Postconditions:	Customer must be created and saved. One or more Addresses must be created and saved. Credit/debit card information must be validated. Account must be created and saved. Address and Account must be associated with Customer.

Fully Developed Use Case Description *Create customer account* (part 2)



Flow of activities:	Actor	System
	1. Customer indicates desire to create customer account and enters basic customer information.	1.1 System creates a new customer. 1.2 System prompts for customer addresses.
	2. Customer enters one or more addresses.	2.1 System creates addresses. 2.2 System prompts for credit/debit card.
	3. Customer enters credit/debit card information.	3.1 System creates account. 3.2 System verifies authorization for credit/debit card. 3.3 System associates customer, address, and account. 3.4 System returns valid customer account details.
Exception conditions:	1.1 Basic customer data are incomplete. 2.1 The address isn't valid. 3.2 Credit/debit information isn't valid.	

Use Case Description Details

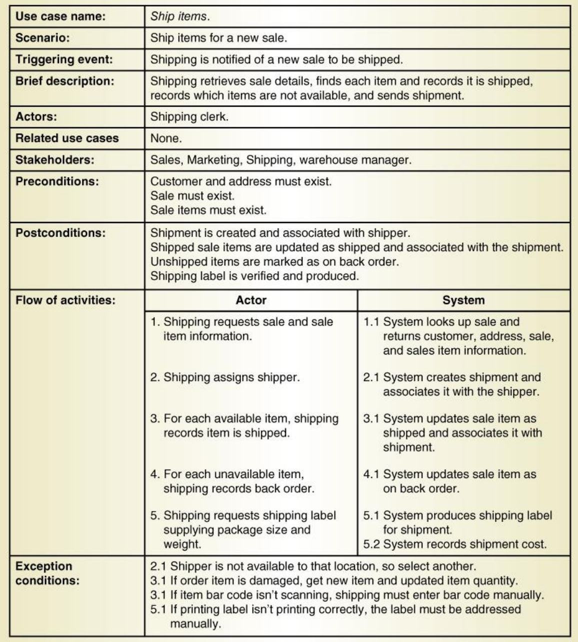
- Use case name
  - Verb-noun
- Scenario (if needed)
  - A use case can have more than one scenario (special case or more specific path)
- Triggering event
  - Based on event decomposition technique
- Brief description
  - Written previously when use case was identified
- Actors
  - One or more users from use case diagrams

Use Case Description Details

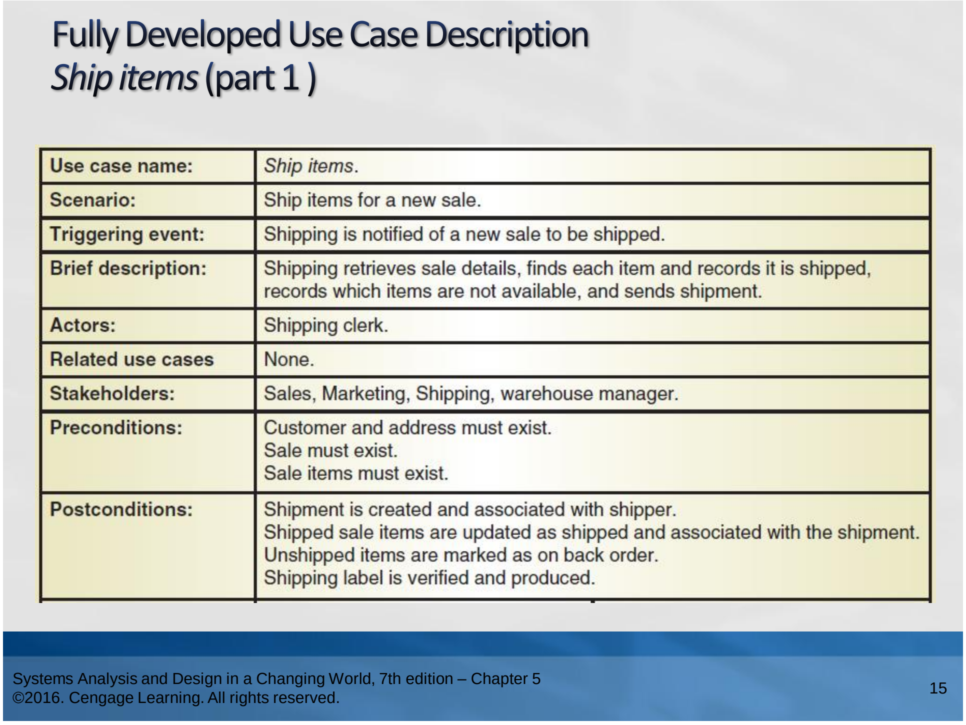
- Related use cases <<includes>>
  - If one use case invokes or includes another
- Stakeholders
  - Anyone with an interest in the use case
- Preconditions
  - What must be true before the use case begins
- Post conditions
  - What must be true when the use case is completed
  - Use for planning test case expected results
- Flow of activities
  - The activities that go on between actor and the system
- Exception conditions
  - Where and what can go wrong

Another Fully Developed Use Case Description Example

Use case
Ship items

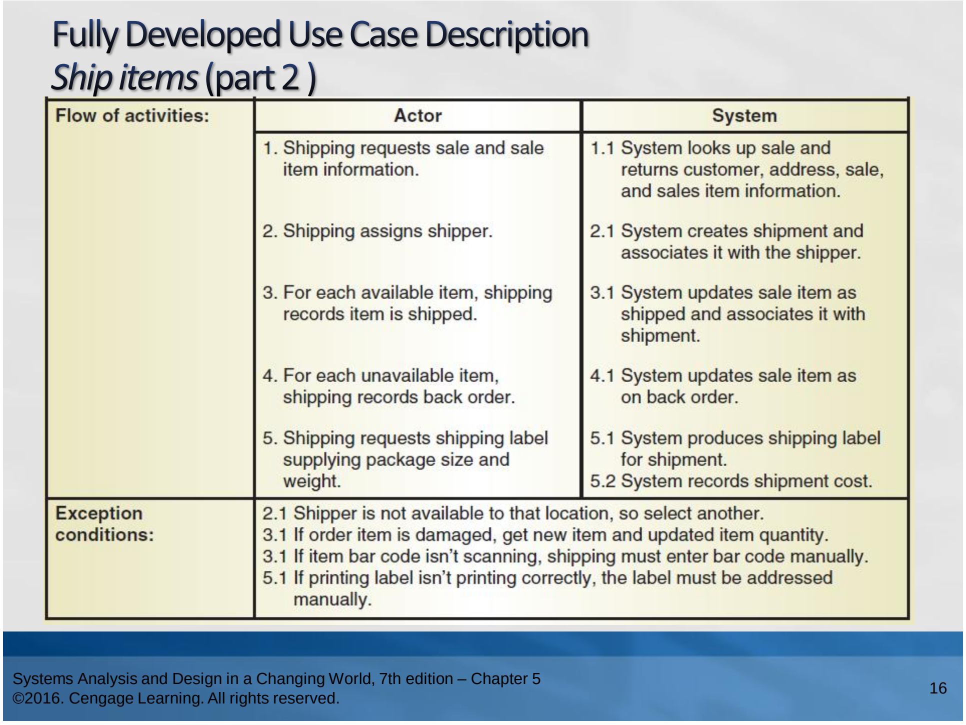


Use case name:	*Ship items*.	
Scenario:	Ship items for a new sale.	
Triggering event:	Shipping is notified of a new sale to be shipped.	
Brief description:	Shipping retrieves sale details, finds each item and records it is shipped, records which items are not available, and sends shipment.	
Actors:	Shipping clerk.	
Related use cases	None.	
Stakeholders:	Sales, Marketing, Shipping, warehouse manager.	
Preconditions:	Customer and address must exist. Sale must exist. Sale items must exist.	
Postconditions:	Shipment is created and associated with shipper. Shipped sale items are updated as shipped and associated with the shipment. Unshipped items are marked as on back order. Shipping label is verified and produced.	
Flow of activities:	**Actor**	**System**
	1. Shipping requests sale and sale item information.	1.1 System looks up sale and returns customer, address, sale, and sales item information.
	2. Shipping assigns shipper.	2.1 System creates shipment and associates it with the shipper.
	3. For each available item, shipping records item is shipped.	3.1 System updates sale item as shipped and associates it with shipment.
	4. For each unavailable item, shipping records back order.	4.1 System updates sale item as on back order.
	5. Shipping requests shipping label supplying package size and weight.	5.1 System produces shipping label for shipment. 5.2 System records shipment cost.
Exception conditions:	2.1 Shipper is not available to that location, so select another. 3.1 If order item is damaged, get new item and updated item quantity. 3.1 If item bar code isn't scanning, shipping must enter bar code manually. 5.1 If printing label isn't printing correctly, the label must be addressed manually.	



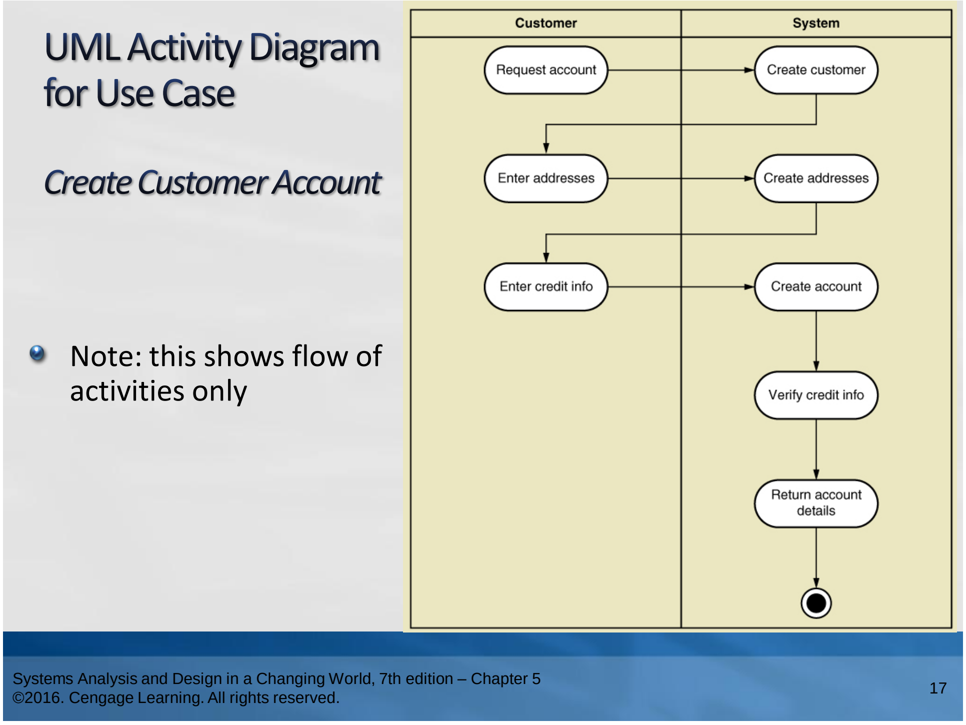
Fully Developed Use Case Description *Ship items* (part 1)

Use case name:	*Ship items*.
Scenario:	Ship items for a new sale.
Triggering event:	Shipping is notified of a new sale to be shipped.
Brief description:	Shipping retrieves sale details, finds each item and records it is shipped, records which items are not available, and sends shipment.
Actors:	Shipping clerk.
Related use cases	None.
Stakeholders:	Sales, Marketing, Shipping, warehouse manager.
Preconditions:	Customer and address must exist. Sale must exist. Sale items must exist.
Postconditions:	Shipment is created and associated with shipper. Shipped sale items are updated as shipped and associated with the shipment. Unshipped items are marked as on back order. Shipping label is verified and produced.



Fully Developed Use Case Description *Ship items* (part 2)

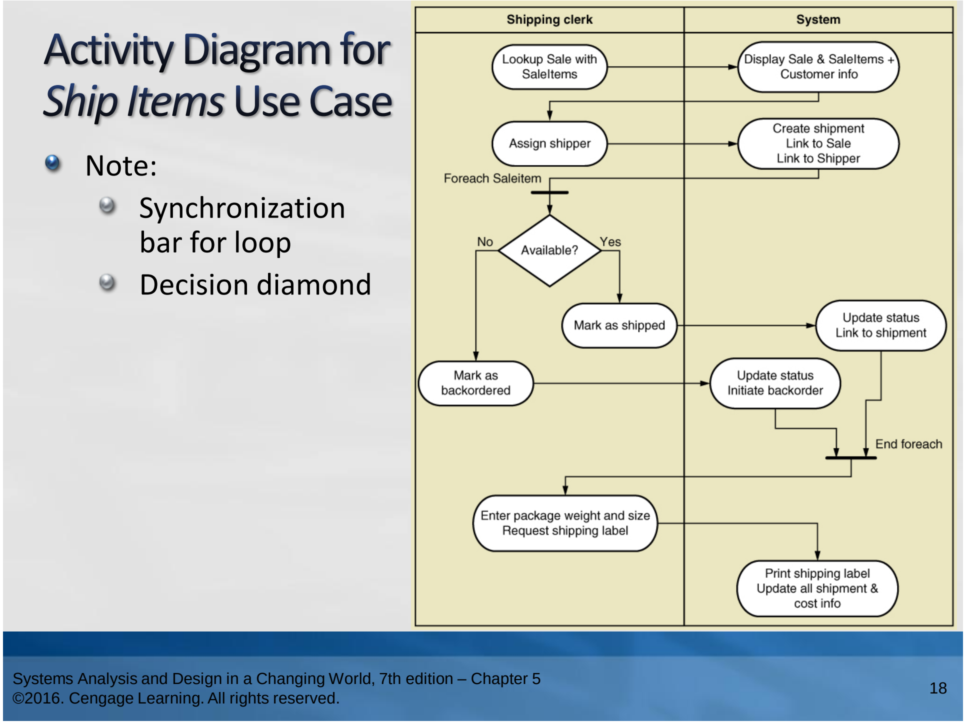
Flow of activities:	**Actor**	**System**
	1. Shipping requests sale and sale item information.	1.1 System looks up sale and returns customer, address, sale, and sales item information.
	2. Shipping assigns shipper.	2.1 System creates shipment and associates it with the shipper.
	3. For each available item, shipping records item is shipped.	3.1 System updates sale item as shipped and associates it with shipment.
	4. For each unavailable item, shipping records back order.	4.1 System updates sale item as on back order.
	5. Shipping requests shipping label supplying package size and weight.	5.1 System produces shipping label for shipment. 5.2 System records shipment cost.
Exception conditions:	2.1 Shipper is not available to that location, so select another. 3.1 If order item is damaged, get new item and updated item quantity. 3.1 If item bar code isn't scanning, shipping must enter bar code manually. 5.1 If printing label isn't printing correctly, the label must be addressed manually.	



UML Activity Diagram for Use Case

Create Customer Account

- Note: this shows flow of activities only



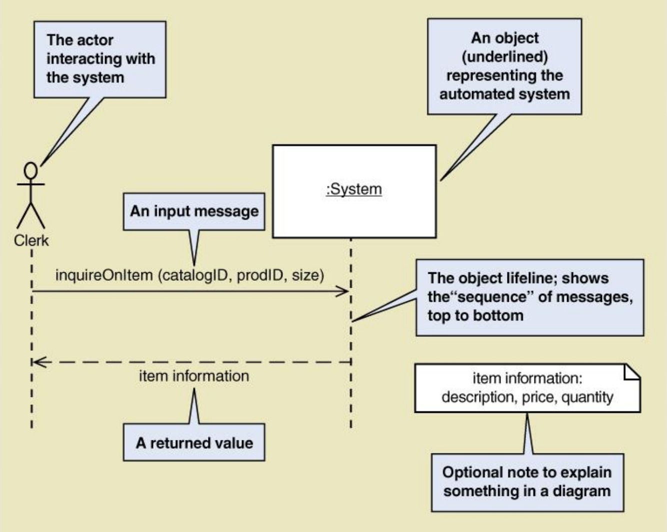
Activity Diagram for *Ship Items* Use Case

- Note:
  - Synchronization bar for loop
  - Decision diamond

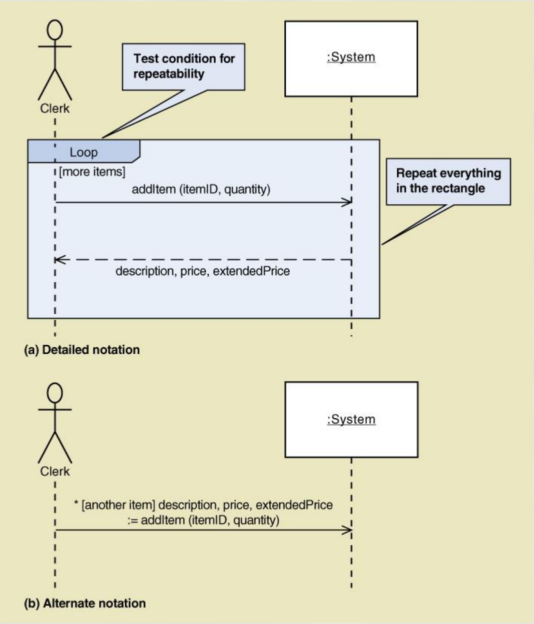
System Sequence Diagram (SSD)

- A UML sequence diagram
- Special case for a sequence diagram
  - Only shows actor and one object
  - The one object represents the complete system
  - Shows input & output messaging requirements for a use case
- Actor, :System, object lifeline
- Messages

System Sequence Diagram (SSD) Notation


The actor interacting with the system
An object (underlined) representing the automated system
:System
An input message
Clerk
inquireOnItem (catalogID, prodID, size)
The object lifeline; shows the"sequence" of messages, top to bottom
item information
item information: description, price, quantity
A returned value
Optional note to explain something in a diagram

SSD Message Examples with Loop Frame


Clerk
Test condition for repeatability
:System
Loop
[more items]
addItem (itemID, quantity)
Repeat everything in the rectangle
description, price, extendedPrice
Clerk
:System
* [another item] description, price, extendedPrice := addItem (itemID, quantity)

(a) Detailed notation

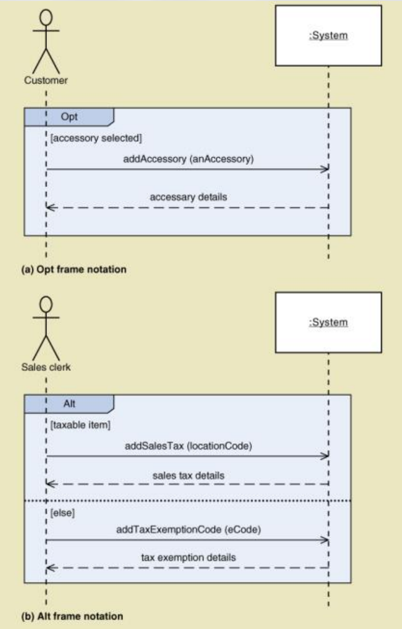
(b) Alternate notation

Message Notation for SSD

- *[true/false condition] return-value := message-name (parameter-list)*
  - An asterisk (*) indicates repeating or looping of the message
  - Brackets [ ] indicate a true/false condition. This is a test for that message only. If it evaluates to true, the message is sent. If it evaluates to false, the message isn't sent.
  - Message-name is the description of the requested service written as a verb-noun.
  - Parameter-list (with parentheses on initiating messages and without parentheses on return messages) shows the data that are passed with the message.
  - Return-value on the same line as the message (requires :=) is used to describe data being returned from the destination object to the source object in response to the message.

SSD Message Examples

- Opt Frame (optional)
- Alt Frame
  (if-else)



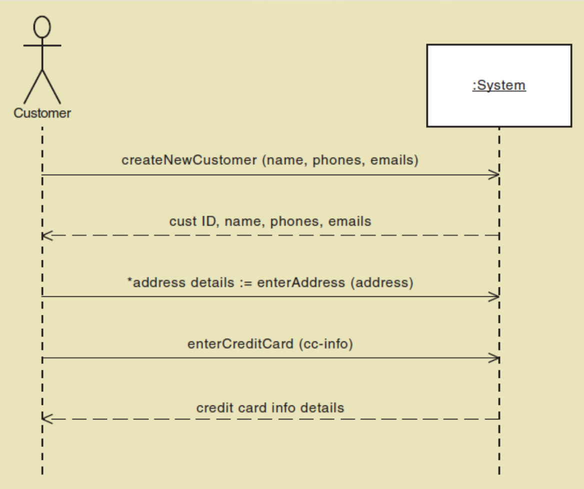
(a) Opt frame notation

(b) Alt frame notation

Steps for Developing SSD

1. Identify input message
   - See use case flow of activities or activity diagram
2. Describe the message from the external actor to the system using the message notation
   - Name it verb-noun: what the system is asked to do
   - Consider parameters the system will need
3. Identify any special conditions on input messages
   - Iteration/loop frame
   - Opt or Alt frame
4. Identify and add output return values
   - On message itself: aValue:= getValue(valueID)
   - As explicit return on separate dashed line

SSD for *Create customer account* Use case



Customer

:System

createNewCustomer (name, phones, emails)

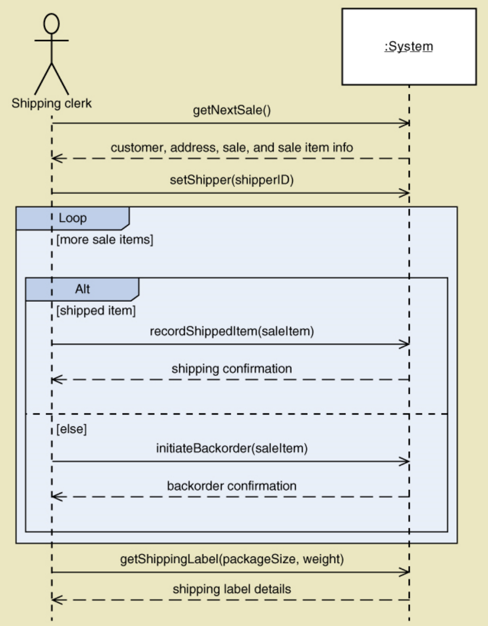
cust ID, name, phones, emails

*address details := enterAddress (address)

enterCreditCard (cc-info)

credit card info details

SSD for *Ship items* Use Case



Shipping clerk

:System

getNextSale()

customer, address, sale, and sale item info

setShipper(shipperID)

Loop

[more sale items]

Alt

[shipped item]

recordShippedItem(saleItem)

shipping confirmation

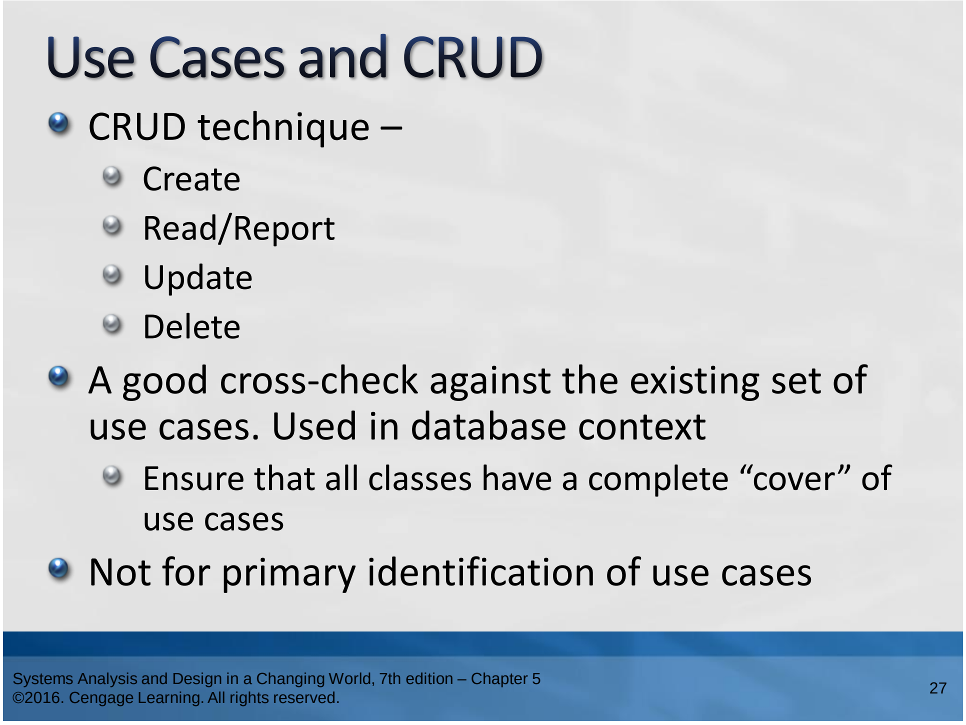
[else]

initiateBackorder(saleItem)

backorder confirmation

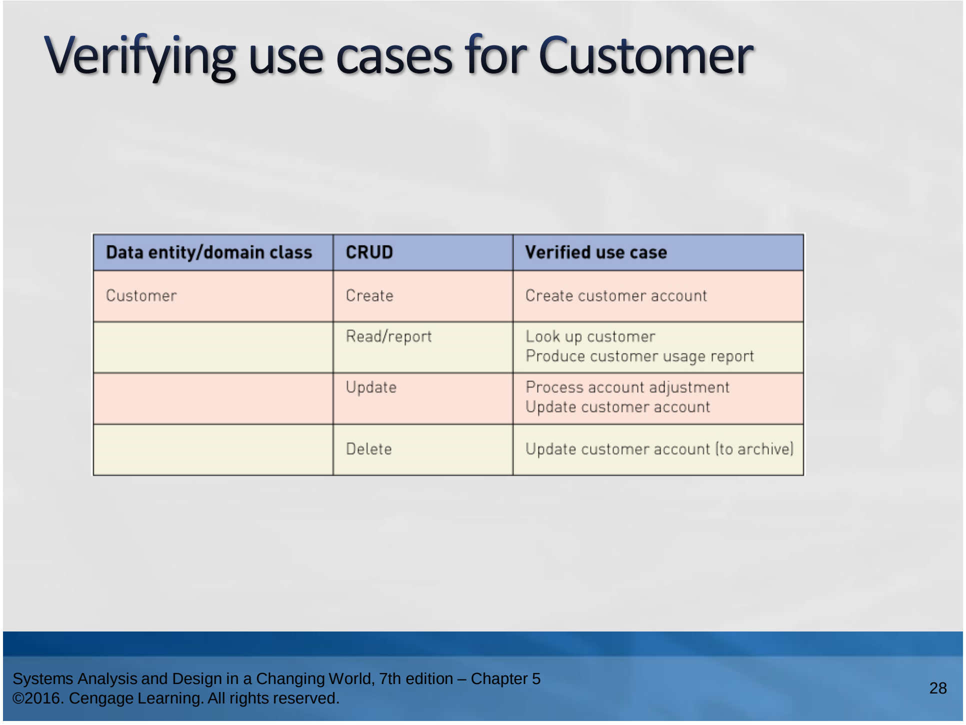
getShippingLabel(packageSize, weight)

shipping label details



Use Cases and CRUD

- CRUD technique –
  - Create
  - Read/Report
  - Update
  - Delete
- A good cross-check against the existing set of use cases. Used in database context
  - Ensure that all classes have a complete "cover" of use cases
- Not for primary identification of use cases



Verifying use cases for Customer

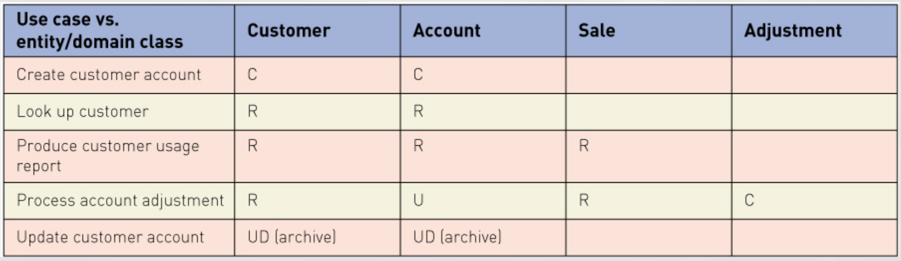
Data entity/domain class	CRUD	Verified use case
Customer	Create	Create customer account
	Read/report	Look up customer Produce customer usage report
	Update	Process account adjustment Update customer account
	Delete	Update customer account (to archive)

CRUD Analysis

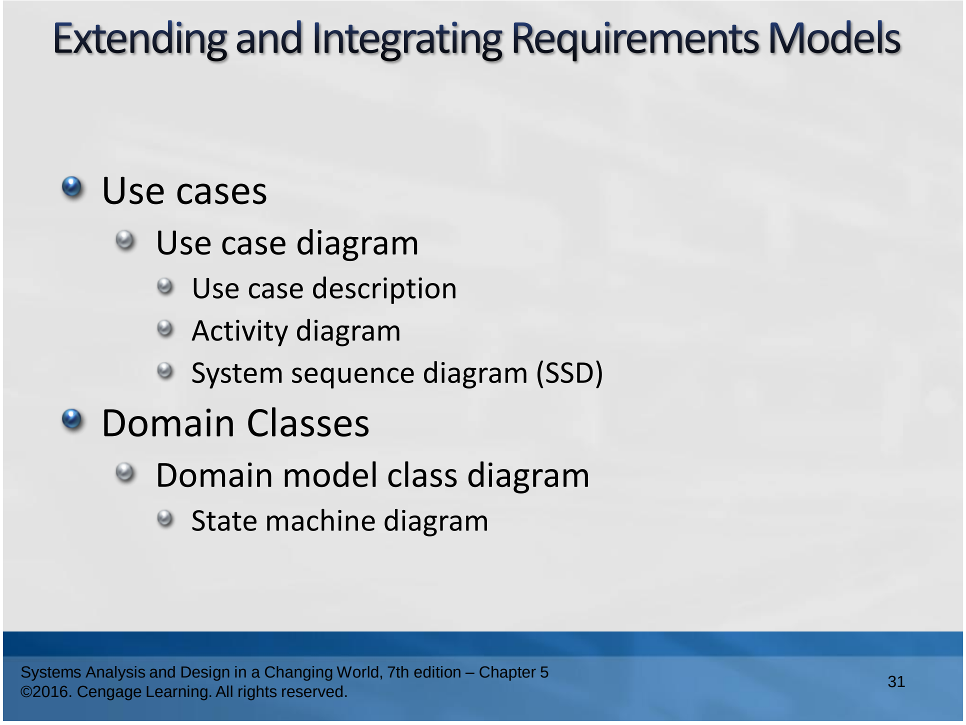
Steps

1. Identify all domain classes
2. For each class verify that use cases exist to
   - Create a new instance
   - Update existing instances
   - Reads or reports on information in the class
   - Deletes or archives inactive instances
3. Add new use cases as required. Identify responsible stakeholders
4. Identify which application has responsibility for each action: which to create, which to update, which to use

Sample CRUD Matrix

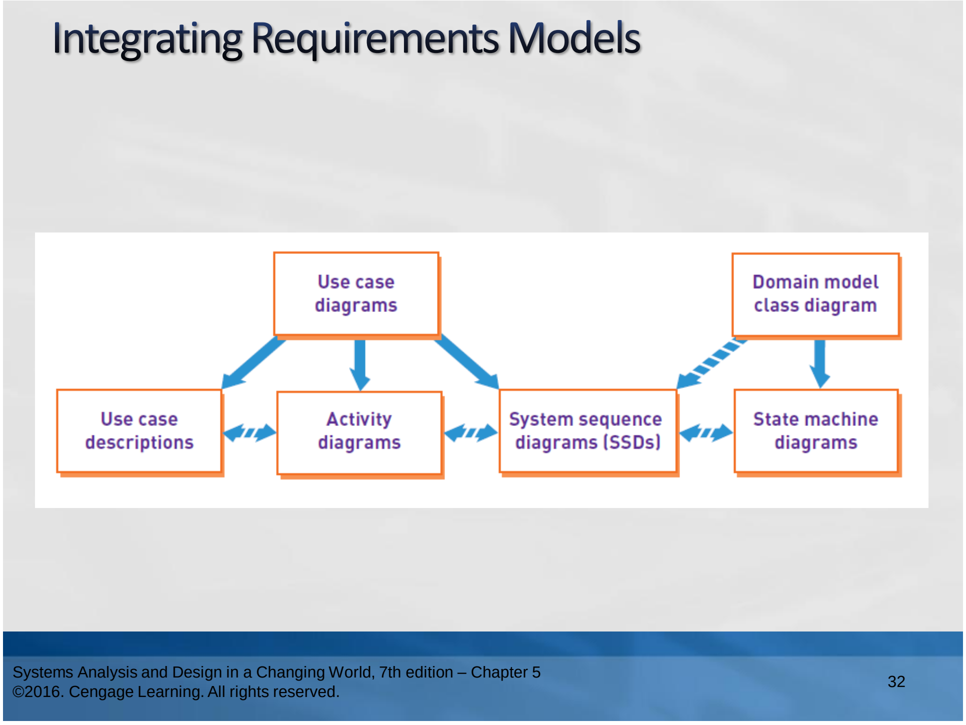


Use case vs. entity/domain class	Customer	Account	Sale	Adjustment
Create customer account	C	C		
Look up customer	R	R		
Produce customer usage report	R	R	R	
Process account adjustment	R	U	R	C
Update customer account	UD (archive)	UD (archive)		



Extending and Integrating Requirements Models

- Use cases
  - Use case diagram
    - Use case description
    - Activity diagram
    - System sequence diagram (SSD)
- Domain Classes
  - Domain model class diagram
    - State machine diagram



Integrating Requirements Models

Use case diagrams

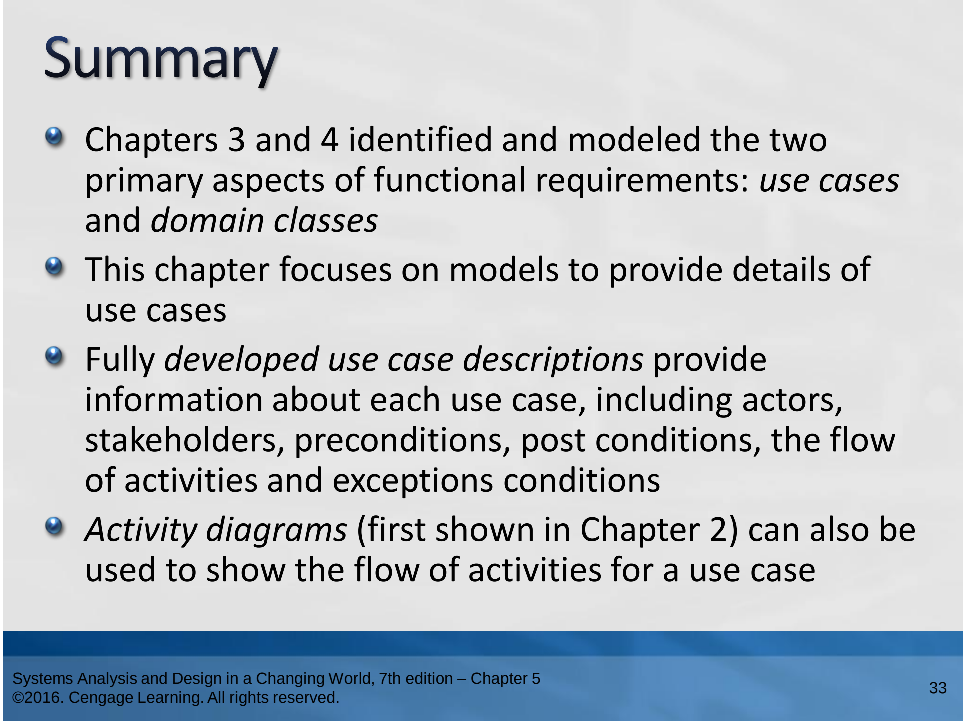
Domain model class diagram

Use case descriptions

Activity diagrams

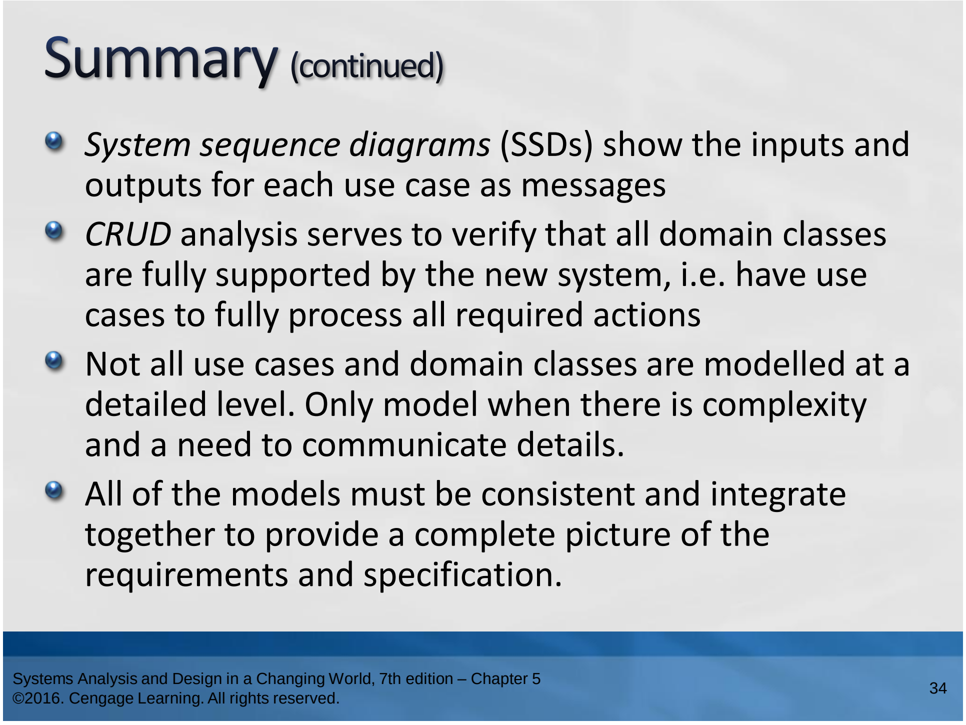
System sequence diagrams (SSDs)

State machine diagrams



Summary

- Chapters 3 and 4 identified and modeled the two primary aspects of functional requirements: *use cases* and *domain classes*
- This chapter focuses on models to provide details of use cases
- Fully *developed use case descriptions* provide information about each use case, including actors, stakeholders, preconditions, post conditions, the flow of activities and exceptions conditions
- *Activity diagrams* (first shown in Chapter 2) can also be used to show the flow of activities for a use case



Summary (continued)

- *System sequence diagrams* (SSDs) show the inputs and outputs for each use case as messages
- *CRUD* analysis serves to verify that all domain classes are fully supported by the new system, i.e. have use cases to fully process all required actions
- Not all use cases and domain classes are modelled at a detailed level. Only model when there is complexity and a need to communicate details.
- All of the models must be consistent and integrate together to provide a complete picture of the requirements and specification.