

EXTENDING THE REQUIREMENTS MODELS

Lecture 7

1.11.2014, Saturday

Outline of This Lecture

- Use Case Descriptions
- Activity Diagrams for Use Cases
- The System Sequence Diagram—
Identifying Inputs and Outputs
- The State Machine Diagram—Identifying
Object Behavior
- Integrating Requirements Models

Use Case Descriptions

- Write a *brief description* as shown in our discussion for most use cases so far.

Use case	Brief use case description
<i>Create customer account</i>	User/actor enters new customer account data, and the system assigns account number, creates a customer record, and creates an account record.
<i>Look up customer</i>	User/actor enters customer account number, and the system retrieves and displays customer and account data.
<i>Process account adjustment</i>	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
 - Post conditions
 - Flow of activities
 - Exception conditions

Fully Developed Use Case Description

Use case: *Create customer account*

Use case name:	<i>Create customer account.</i>	
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 <i>Check out shopping cart</i> 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:	<i>Create customer account.</i>
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 <i>Check out shopping cart</i> 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. 2. Customer enters one or more addresses. 3. Customer enters credit/debit card information.	1.1 System creates a new customer. 1.2 System prompts for customer addresses. 2.1 System creates addresses. 2.2 System prompts for credit/debit card. 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

- 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

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

Another Fully Developed Use Case Description Example

Use case *Ship items*

Use case name:	<i>Ship items.</i>	
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:	<i>Ship items.</i>
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.

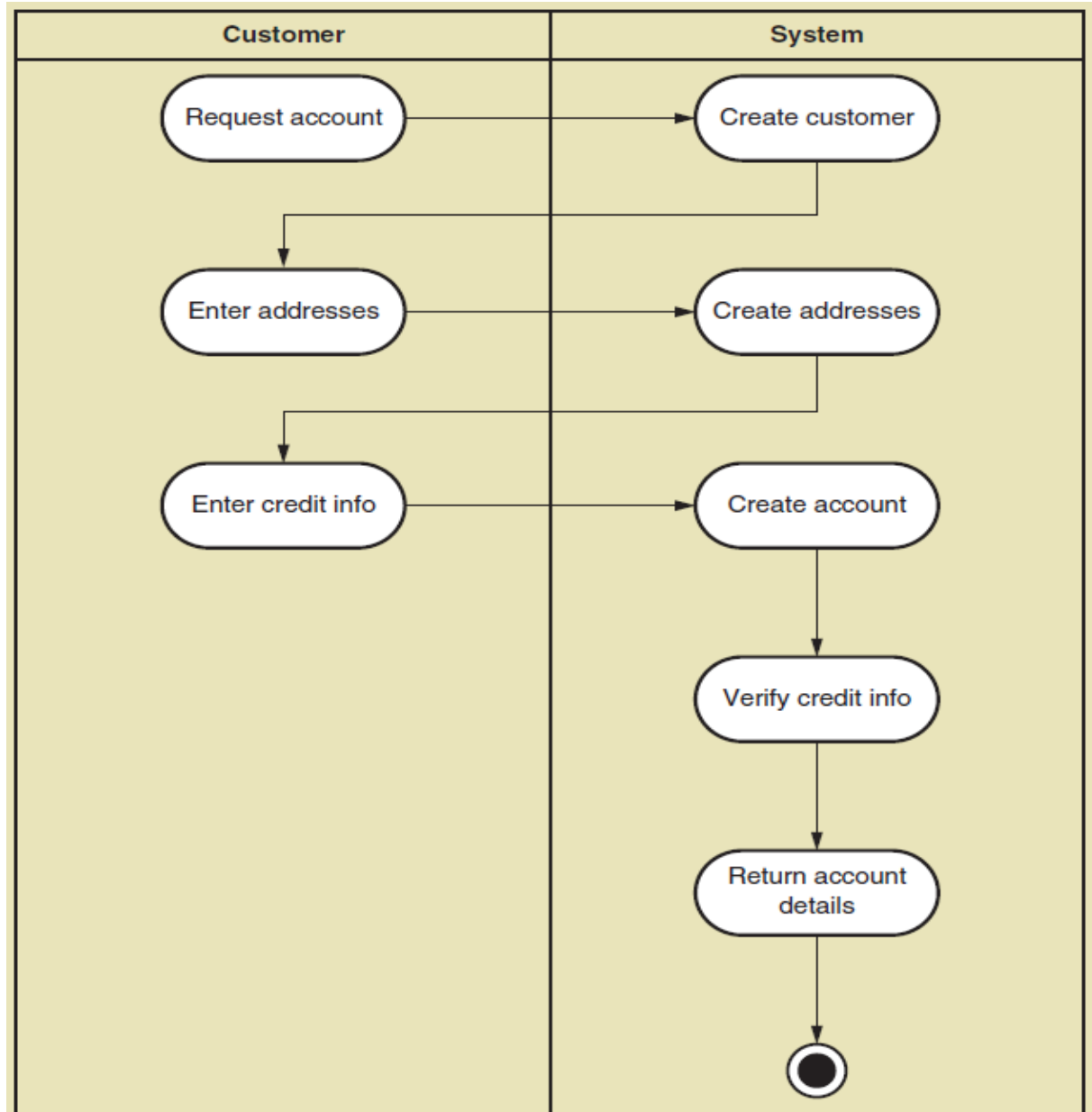
Fully Developed Use Case Description *Ship items (part 2)*

Flow of activities:	Actor	System
	<ol style="list-style-type: none"> 1. Shipping requests sale and sale item information. 2. Shipping assigns shipper. 3. For each available item, shipping records item is shipped. 4. For each unavailable item, shipping records back order. 5. Shipping requests shipping label supplying package size and weight. 	<ol style="list-style-type: none"> 1.1 System looks up sale and returns customer, address, sale, and sales item information. 2.1 System creates shipment and associates it with the shipper. 3.1 System updates sale item as shipped and associates it with shipment. 4.1 System updates sale item as on back order. 5.1 System produces shipping label for shipment. 5.2 System records shipment cost.
Exception conditions:	<ol style="list-style-type: none"> 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

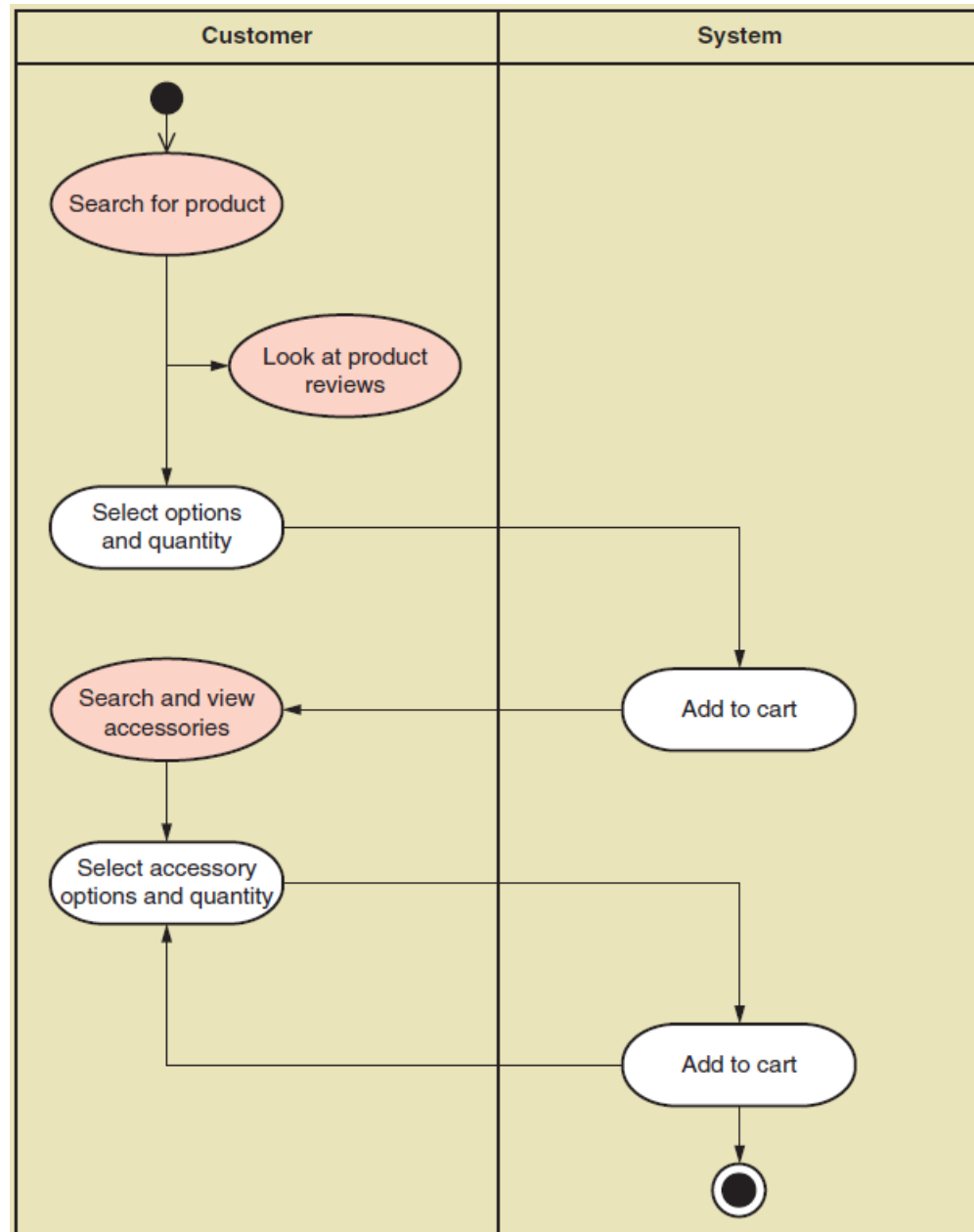
Note: this shows flow of activities only



UML Activity Diagram for Use Case

Fill shopping cart

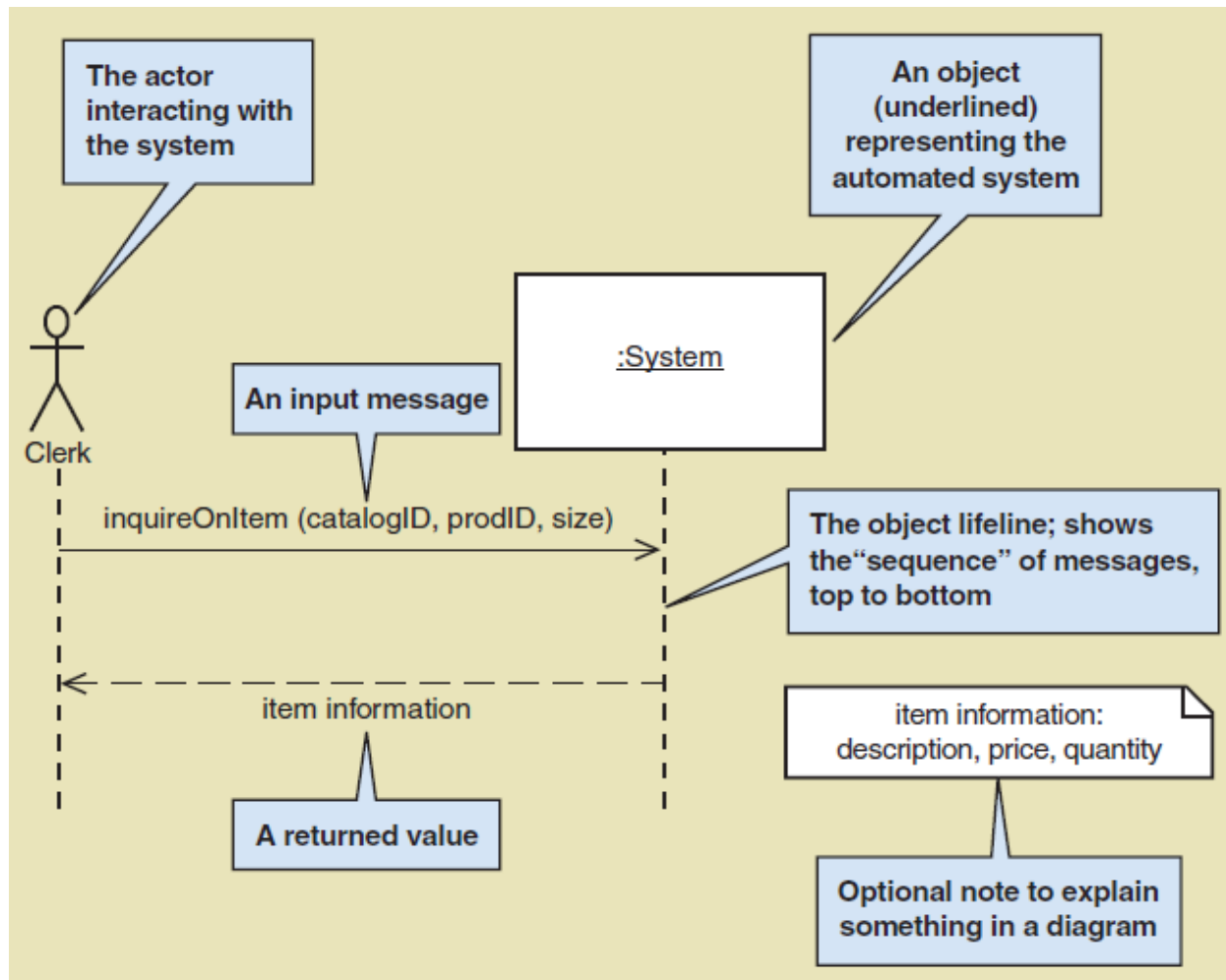
Note: this shows use case with <<includes>> relationship



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

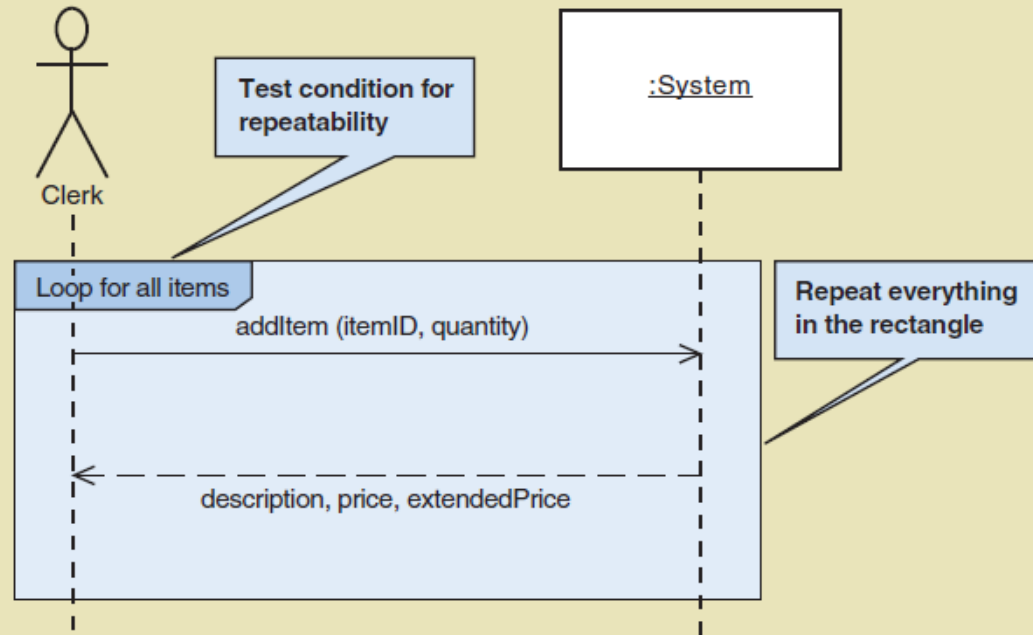


Message Notation

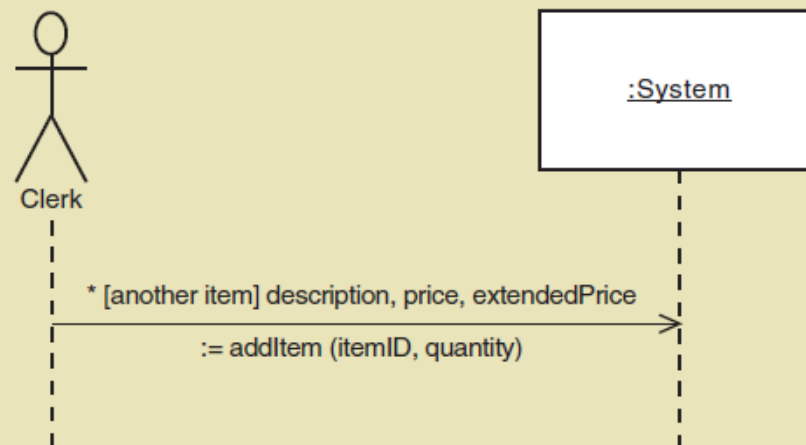
[true/false condition] return-value := message-name (parameter-list)

- 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. It is omitted on dashed-line return messages, which only show the return data parameters.
- 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 with Loop Frame



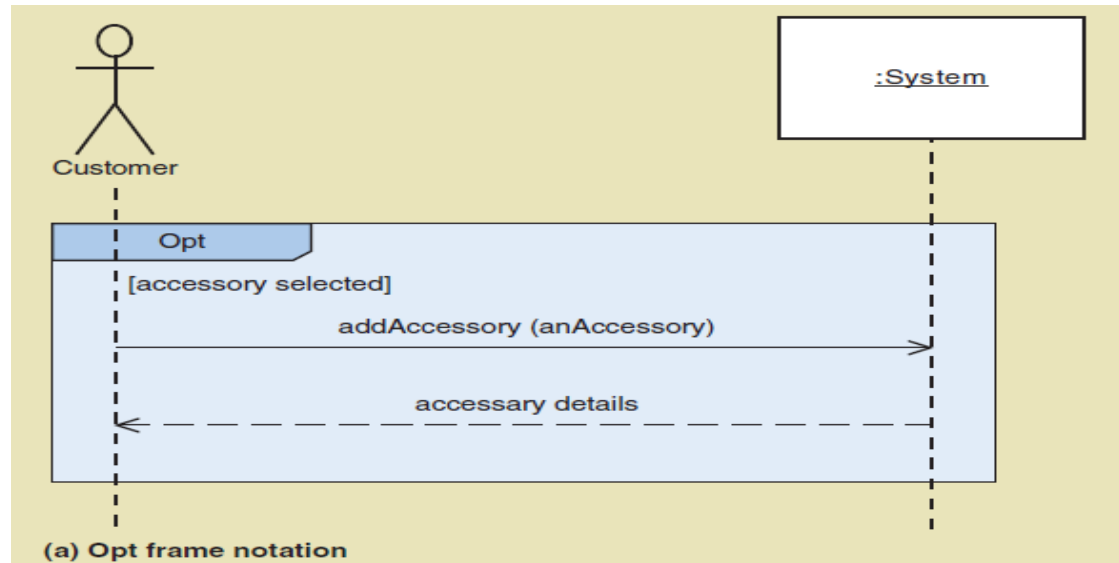
(a) Detailed notation



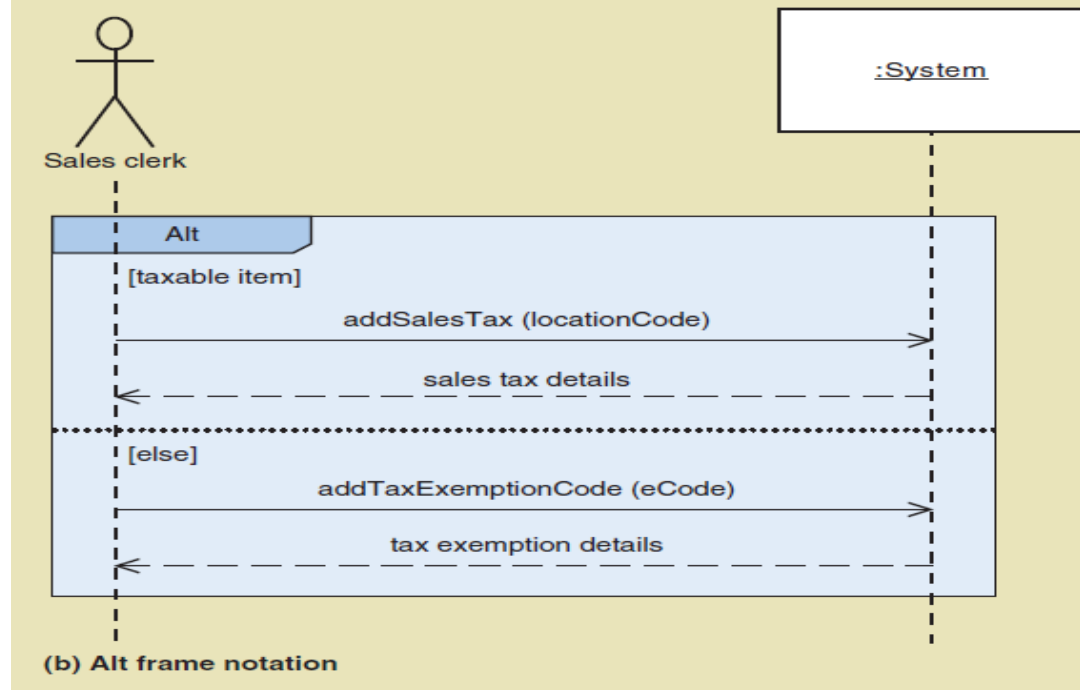
(b) Alternate notation

SSD Message Examples

Opt Frame (optional)



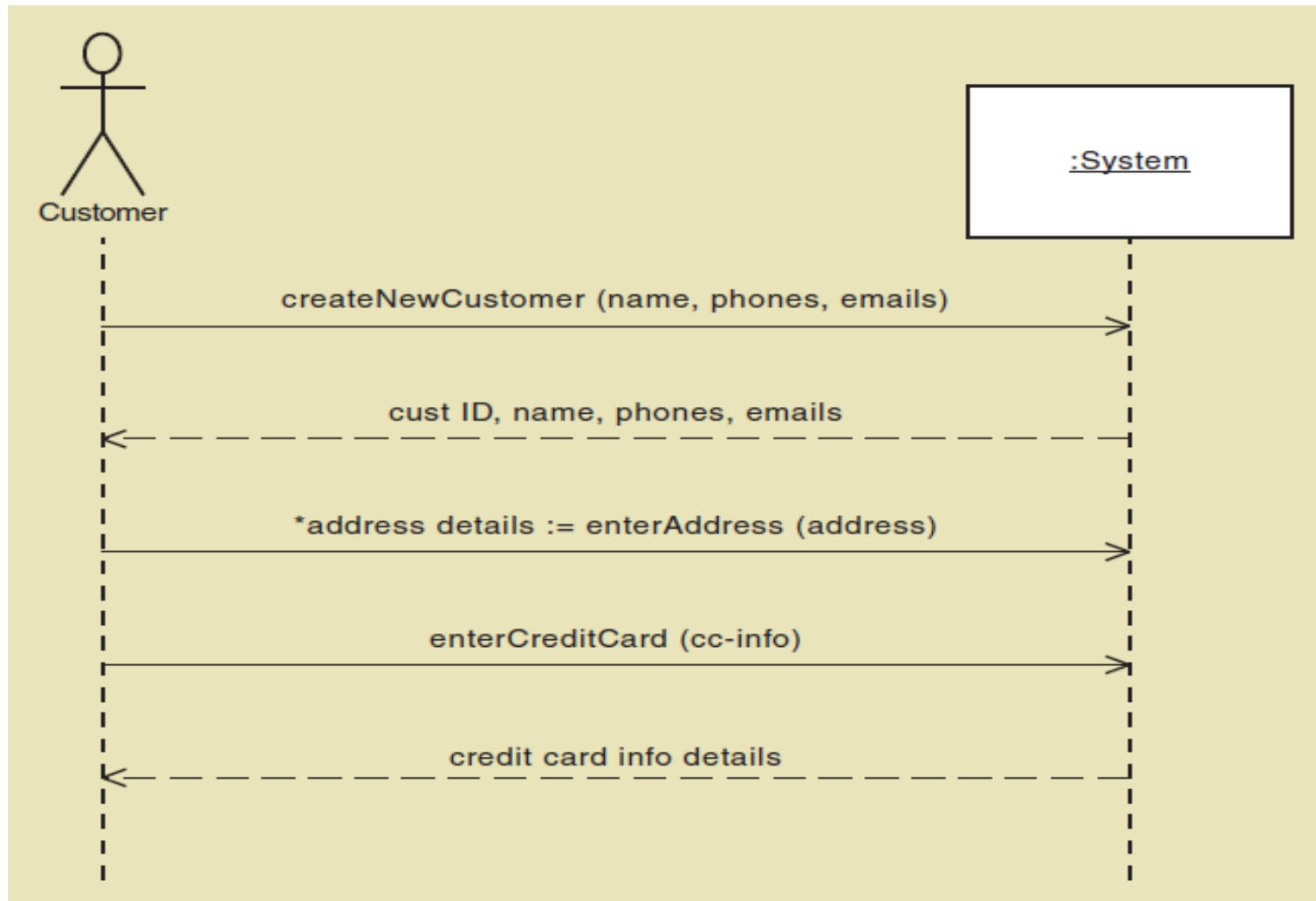
Alt Frame (if-else)



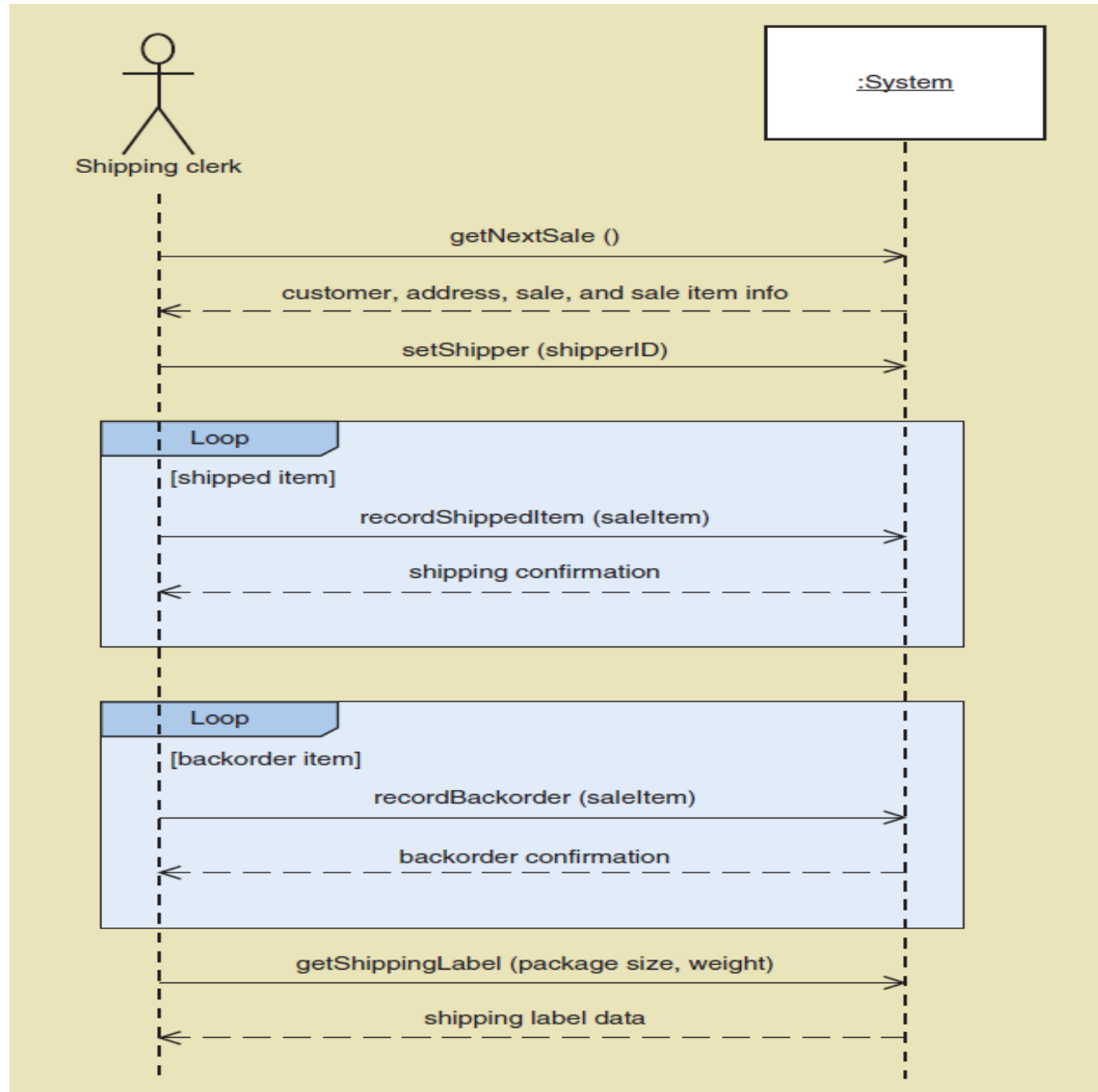
Steps for Developing SSD

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



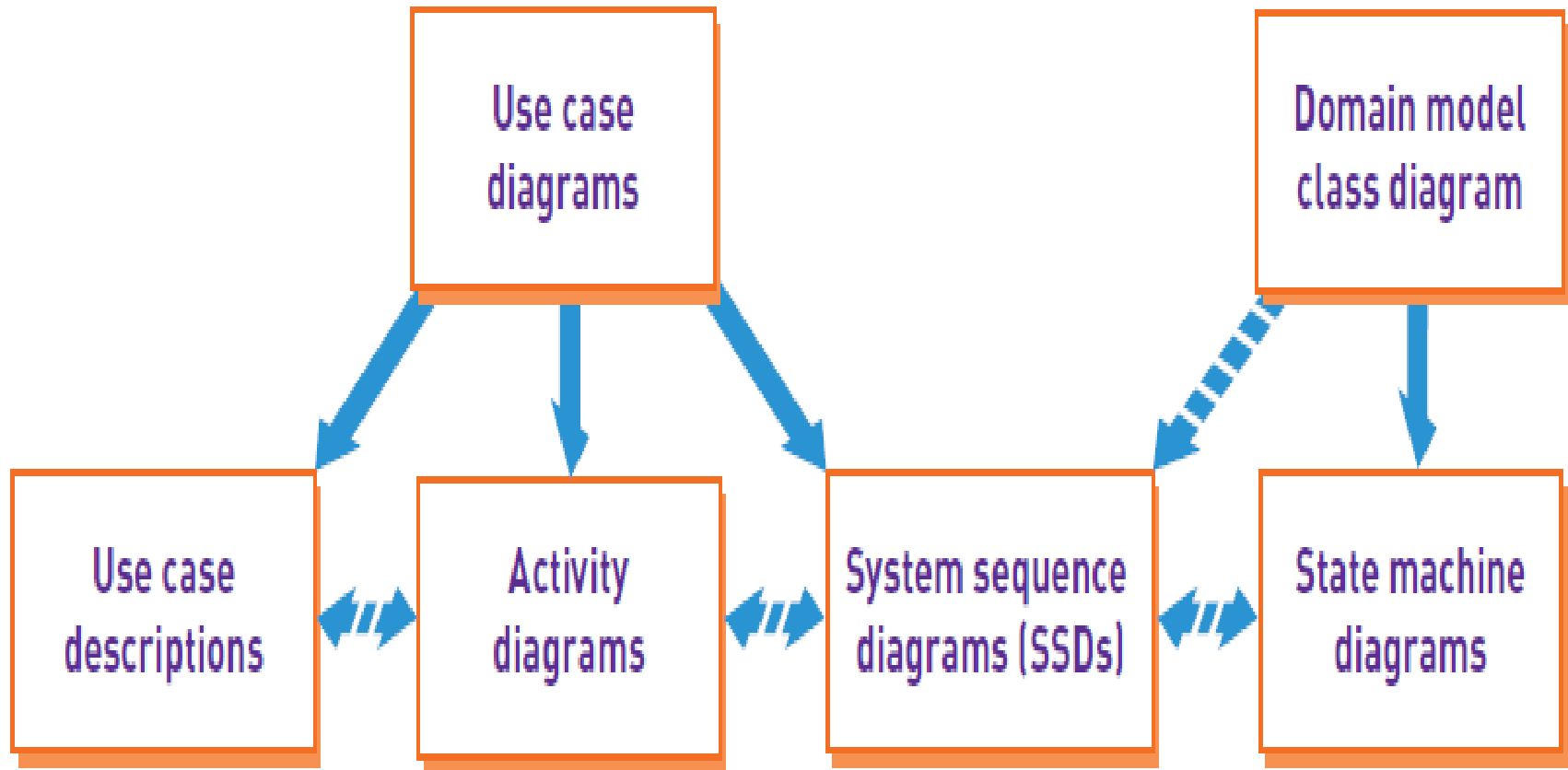
SSD for *Ship items* Use Case



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



Summary

- In the last two lectures, we identified and modelled the two primary aspects of functional requirements: *use cases* and *domain classes*
- This lecture focuses on additional techniques and models to extend the requirements models to show more detail
- 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 the initial lecture) 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
- Use cases are modelled in more detail using fully developed use case descriptions, activity diagrams, and system sequence diagrams
- Domain classes are modelled in more detail using state machine diagrams
- Not all use cases and domain classes are modelled at this level of detail. Only model when there is complexity and a need to communicate details