Murdoch
UNIVERSITY

Extending the requirements models

Topic 5

ICT284 Systems Analysis and Design

About this topic

We have already covered two primary models of functional requirements: use cases and domain class models. The next step is to review these models for consistency and to document parts of them in more depth, as we begin to move from analysis towards design. In this topic we cover some additional techniques and models to extend the analysis models to show further information about the system. In particular, we focus on fully developed use case descriptions to document the internal steps within a use case. We'll also cover system sequence diagrams (SSDs), state machine diagrams (SMDs), and the CRUD technique for cross-checking the domain classes and use cases.

Unit learning outcomes addressed in this topic

1. Explain how information systems are used within organisations to fulfil organisational needs
2. **Describe the phases and activities typically involved in the systems development life cycle**
3. Describe the professional roles, skills and ethical issues involved in systems analysis and design work
4. Use a variety of techniques for analysing and defining business problems and opportunities and determining system requirements
5. **Model system requirements using UML, including use case diagrams and descriptions, activity diagrams and domain model class diagrams**
6. Explain the activities involved in systems design, including designing the system environment, application components, user interfaces, database and software
7. Represent early system design using UML, including sequence diagrams, architectural diagrams and design class diagrams
8. Describe tools and techniques for planning, managing and evaluating systems development projects
9. Describe the key features of several different systems development methodologies
10. **Present systems analysis and design documentation in an appropriate, consistent and professional manner**

Murdoch UNIVERSITY

Topic learning outcomes

After completing this topic you should be able to:

- Explain how additional information about use cases can be represented in detail
- Create a **CRUD** table (CRUD matrix) to verify use cases against the domain model
- Interpret and write **fully developed use case descriptions**
- Develop **activity diagrams** to document the flow of activities within a use case
- Develop **system sequence diagrams** to model the interaction between actors and the system
- Develop **state machine diagrams** to model object behavior

Murdoch UNIVERSITY

Resources for this topic

READING

- Satzinger, Jackson & Burd, Chapter 5
- Satzinger, Jackson & Burd, Chapter 2 p60-62 (activity diagrams)
- Satzinger, Jackson & Burd, Chapter 4 p114-122 (State Machine Diagrams)

Except where otherwise referenced, all images in these slides are from those provided with the textbook: Satzinger, J., Jackson, R. and Burd, S. (2016) *Systems Analysis and Design in a Changing World*, 7th edition, Course Technology, Cengage Learning: Boston. ISBN-13 9781305117204

Murdoch UNIVERSITY

Topic outline

- CRUD technique for checking model consistency
- Brief use case descriptions
- Fully developed use case descriptions
- Activity diagrams for use cases
- System sequence diagrams (SSD)
- State Machine Diagrams (SMD)
- Summary of requirements models

Introduction

Murdoch UNIVERSITY

Almost completed analysis activities …

- We have already covered two primary aspects of functional requirements: use cases and problem domain classes
- In this topic we cover some additional techniques and models to extend these models to show further information about the system

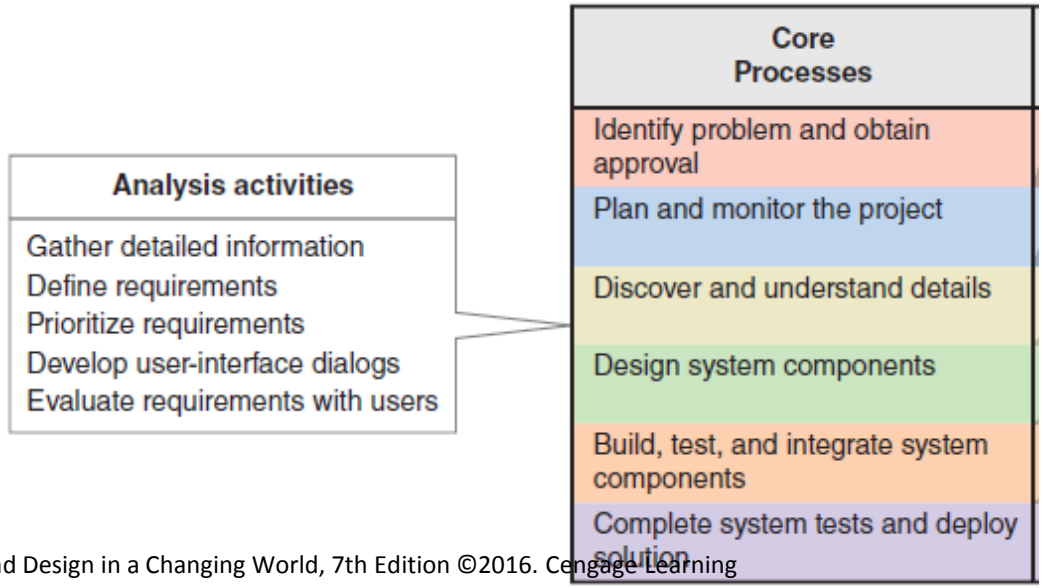


Image from: Systems Analysis and Design in a Changing World, 7th Edition ©2016. Cengage Learning

Revision

What is this diagram called?

What does it tell us?

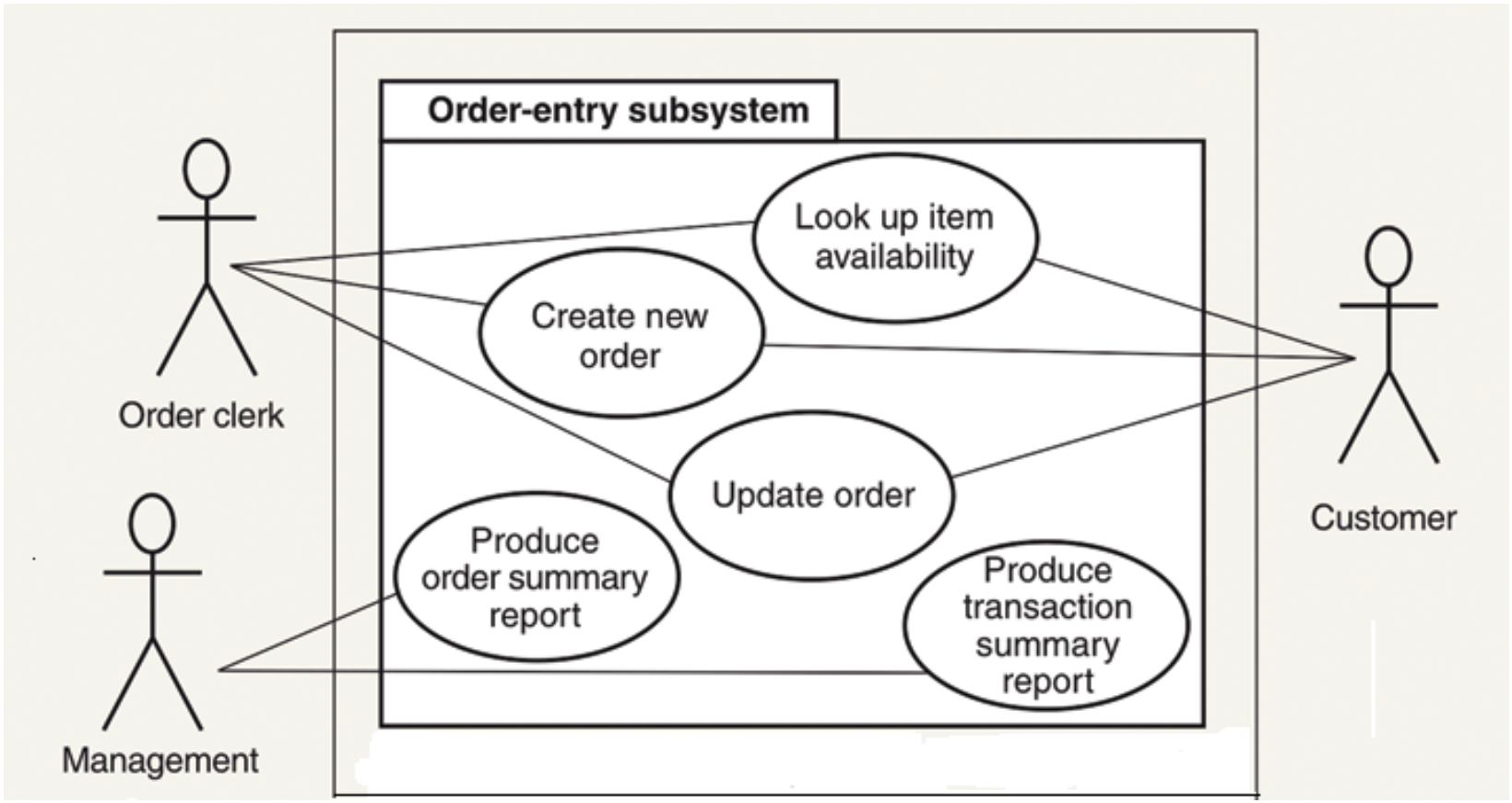
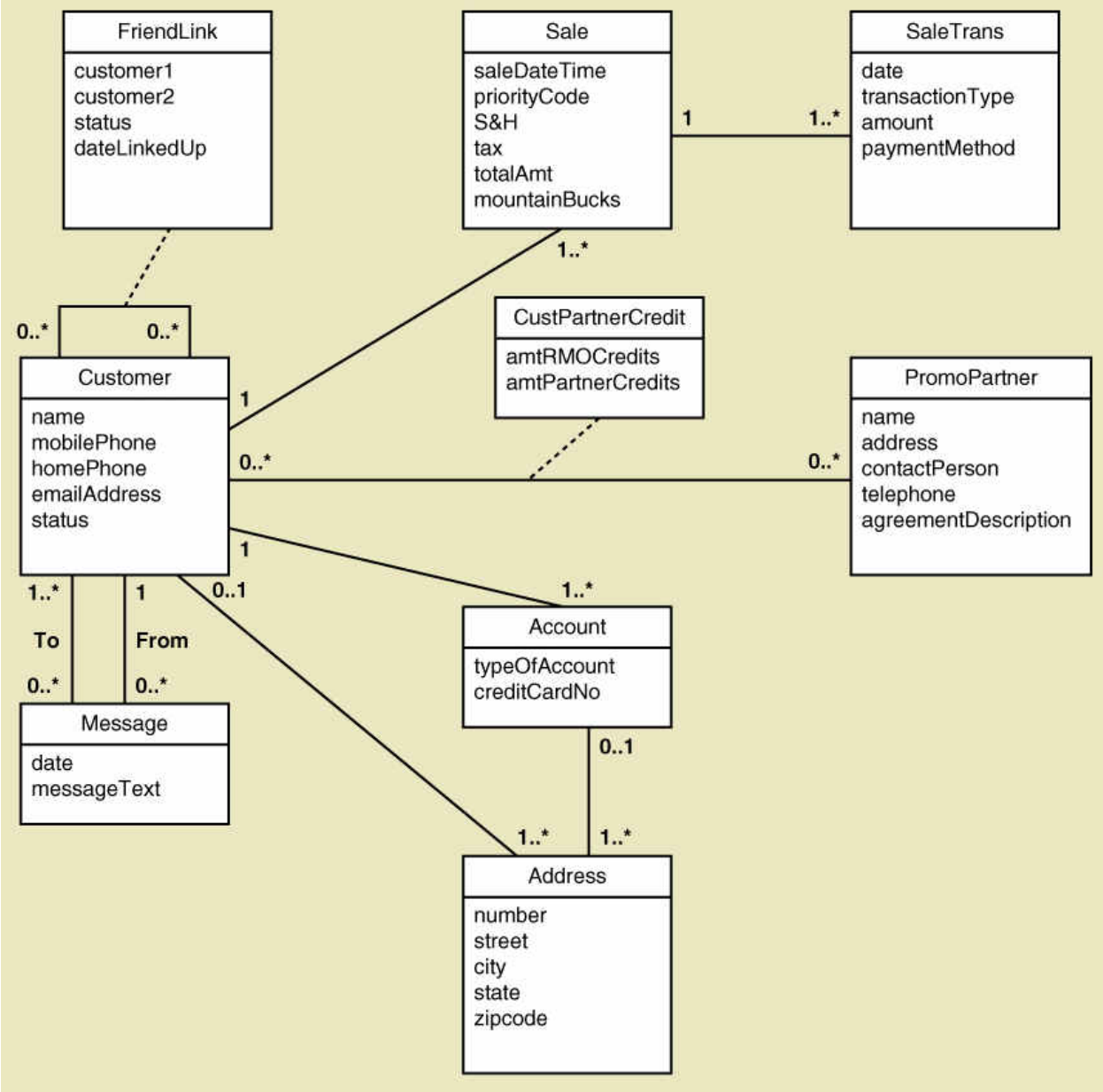


Image from Satzinger, J. Jackson, R. & Burd, S. Systems Analysis and Design in a Changing World, ? edition. Course Technology, Thomson Learning.

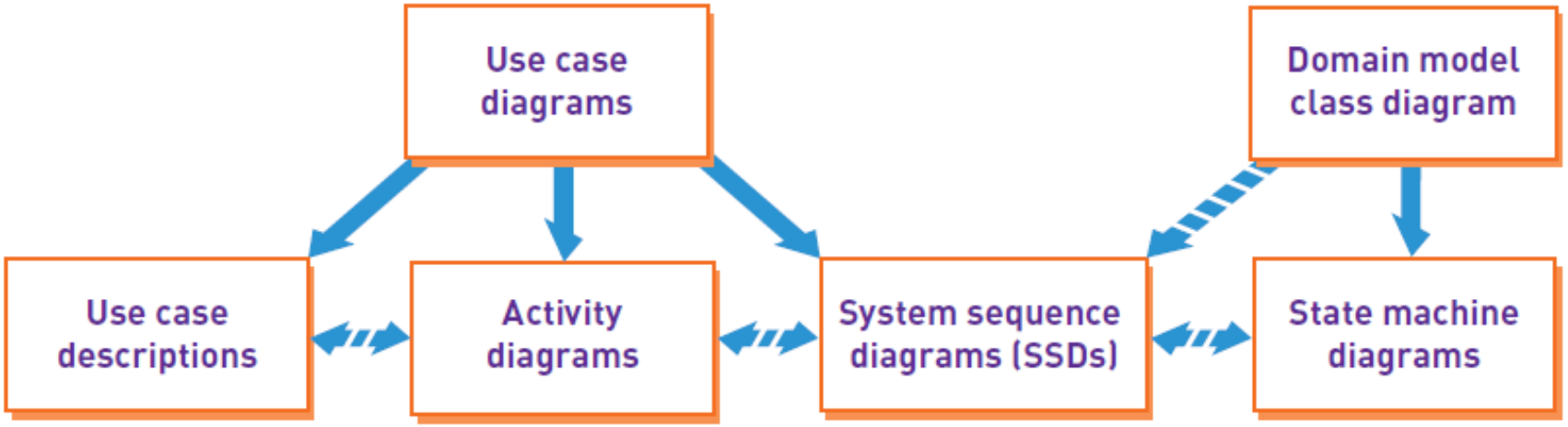
Revision

What is the name of this diagram?

What does it tell us?



Extending and integrating the requirements models



Overview

Murdoch UNIVERSITY

- Topics 3 and 4 identified and modeled the two primary aspects of functional requirements: *use cases* and *domain classes*
- This topic focuses on detailed modelling for use cases to document the internal steps within a complex 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

Overview (continued)

- *Activity diagrams* can be used to show the flow of activities for a use case
- *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
- The use case modelling can be complemented by extending the domain modelling by identifying object behaviour using *state machine diagrams*

Brief use case descriptions

Murdoch UNIVERSITY

Use case descriptions

Murdoch UNIVERSITY

- Write a *brief description* for every use case:

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.

- Complex use cases will also require a *fully developed use case description* (discussed later)

Image from: Systems Analysis and Design in a Changing World, 7th Edition ©2016. Cengage Learning

CRUD technique for verifying use cases

Murdoch UNIVERSITY

CRUD technique

- CRUD stands for –

  Create

  Read/Report

  Update

  Delete

- The CRUD technique provides a way of verifying that *all the required use cases have been identified*
- And that *all the domain classes are supported* by the set of defined use cases
- There are two main ways of using the technique (next slides)

CRUD: 1. Verifying use cases

- In this form of CRUD analysis each operation (C, R, U, D) is checked to verify there is a relevant use case. Done for each 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)

- This example shows that the identified use cases are sufficient to maintain Customer data

CRUD analysis - Steps

Murdoch UNIVERSITY

1. Identify all domain classes
2. For each class verify that use cases exist to:
   - Create a new instance
   - Update existing instances
   - Read or report on information in the class
   - Delete or archive inactive instances
3. Add new use cases as required. Identify responsible stakeholders/actors
4. If there are different subsystems/applications, identify which has responsibility for each action: which to create, which to update, which to use the data

CRUD: 2. Cross-checking use cases and domain classes

Murdoch UNIVERSITY

- Cross-match all of the domain classes and use cases with the operations they perform

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)		

- This example shows that the 'Sale' class is read but never updated. 'Adjustment' is created but never used – additional use cases will be required

Summing up…

- The **CRUD** technique is a way of ensuring consistency between the use case modelling and the domain modelling
- It documents whether there is a use case to **c**reate, **r**ead, **u**pdate and **d**elete each domain class
- And whether domain classes exist to support the requirements of each use case
- If any inconsistencies are found, the models can be questioned and corrected

Murdoch UNIVERSITY

Fully-developed use case descriptions

Murdoch UNIVERSITY

Fully developed use case descriptions

Murdoch UNIVERSITY

- Where a use case is more complex, we may need to write a more detailed *fully developed use case description*
- Typically, a template is completed that ensures all the required information is documented formally
- … the one used in the textbook is described here

Fully developed use case description

(Larger version on next slides)

- Use case name
- Scenario (if needed)
- Triggering event
- Brief description
- Actors
- Related use cases (<<includes>>)
- Stakeholders
- Preconditions
- Post conditions
- Flow of activities
- Exception conditions

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.	

Image from: Systems Analysis and Design in a Changing World, 7th Edition ©2016. Cengage Learning

Fully developed use case description *Create customer account* (part 1)

Murdoch UNIVERSITY

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.	

Image from: Systems Analysis and Design in a Changing World, 7th Edition ©2016. Cengage Learning

Use case description details 1

- **Use case name**

  Verb-noun
- **Scenario** (only if needed)

  A use case can have more than one scenario
  - e.g. ‘create customer account’ might have two scenarios, ‘create online’ and ‘create by phone’; or be invoked by different actors
  - Each scenario would have a slightly different flow of activities
- **Triggering event**

  Based on event decomposition technique

Use case description details 2

- **Brief description**

  Can use the original 'brief description' written when the use case was identified

- **Actors**

  From the use case diagram

  The person or role that interacts with the *automated* part of the system

  - by specifying 'automated' it ensures we can define the user interface dialogs precisely

Use case description details 3

- **Related use cases**

  If one use case invokes or <<includes>> another
- **Stakeholders**

  Anyone with an interest in the use case, other than the actors involved

Fully developed use case description *Create customer account* (part 1)

Murdoch UNIVERSITY

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.

Use case description details 4

Murdoch UNIVERSITY

- **Preconditions**

  What must be true before the use case begins

  - What objects already exist, what information must be available

- **Post conditions**

  What must be true when the use case is completed:

  - What new objects are created or updated

  - how objects are now associated (e.g. an Account is now associated with a Customer)

  - Use for planning test case expected results
  - For design stage - which objects will be involved in collaborating

Use case description details 5

- **Flow of activities**

  The activities that go on between actor and the system
  - Use a text description, using numbers to indicate flow sequence
  - or an *activity diagram*
- **Exception conditions**
  - Alternative conditions or unexpected conditions (e.g. credit information isn't valid)
  - Link to specific step in the flow of activities described above

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.	

Image from: Systems Analysis and Design in a Changing World, 7th Edition ©2016. Cengage Learning

Another fully developed use case description example: *Ship Items*

Go through this one later at your own pace to make sure you fully understand the technique

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)

Murdoch UNIVERSITY

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)

Murdoch UNIVERSITY

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.	

Summing up…

- Fully developed use case descriptions provide a comprehensive description of the context of a use case and the actions that occur in it
- Typically, a template is completed that ensures all the required information is documented formally
- The fully developed use case description is a basis for later documentation (such as system sequence diagrams and sequence diagrams)
- The preconditions and postconditions included in the fully developed use case descriptions form a basis for later software testing

Activity diagrams for use cases

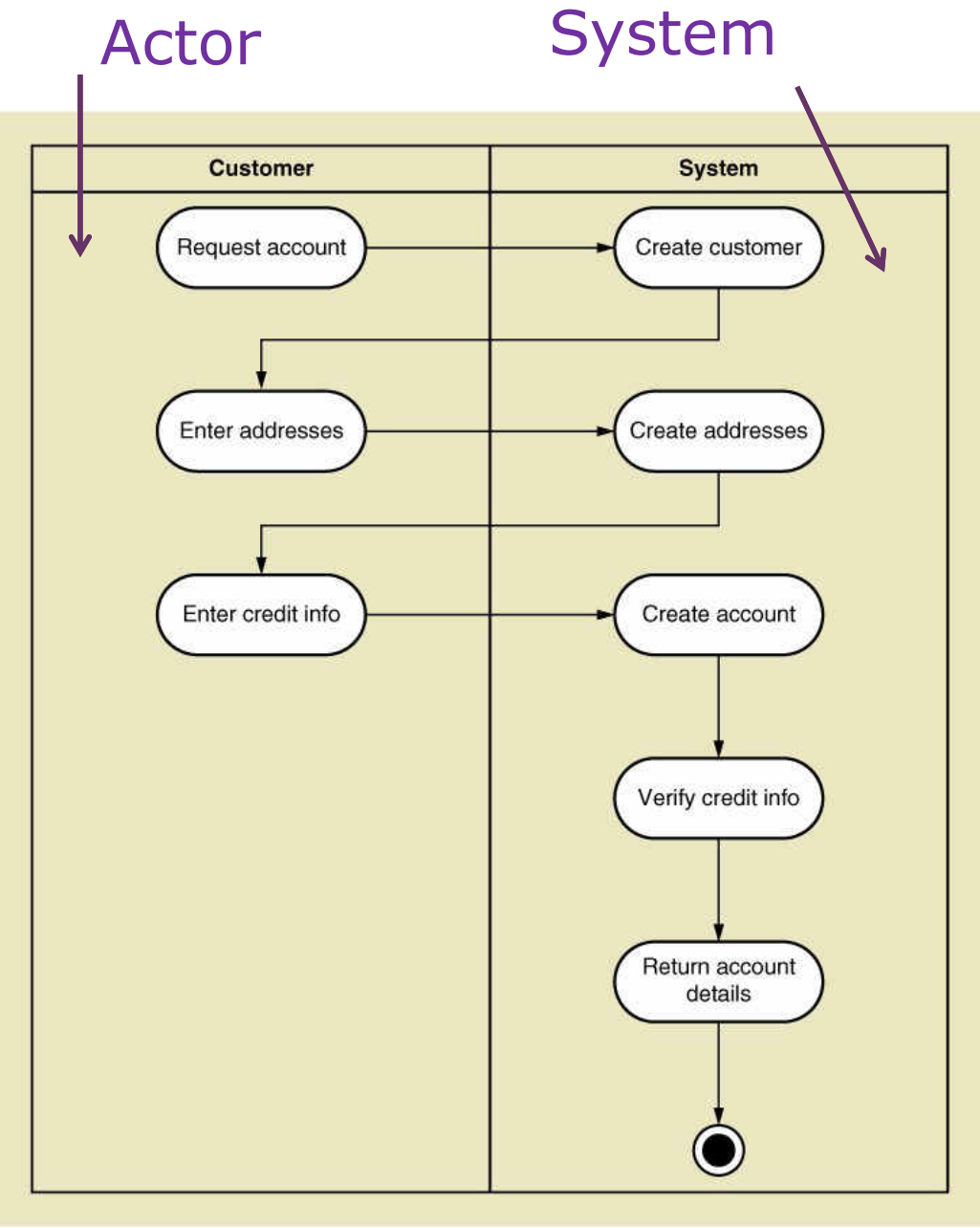
Murdoch UNIVERSITY

Activity diagrams for use case descriptions

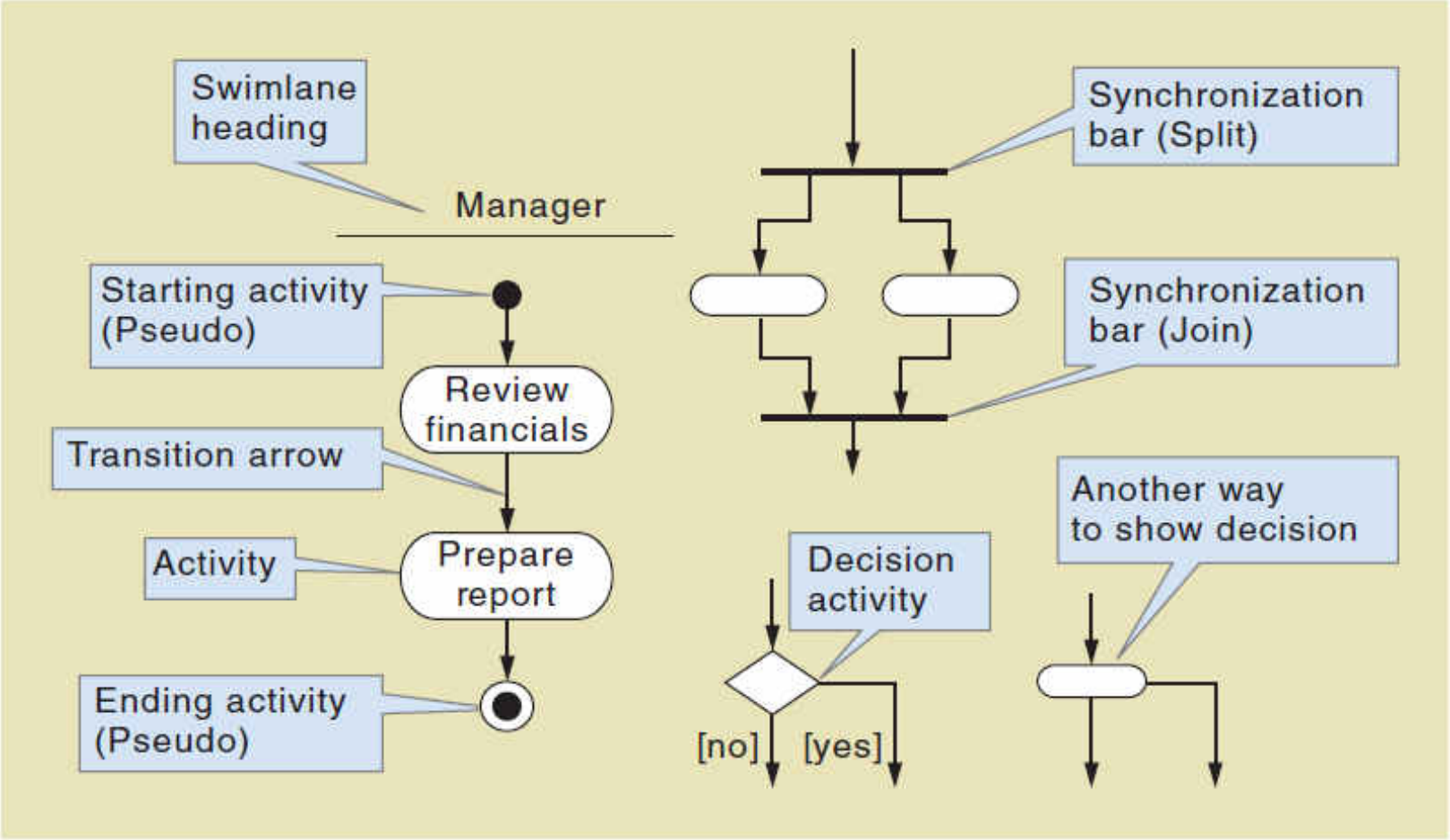
- We can use the **activity diagram** notation to model the flow of activities between the *Actor(s)* and the *System* within a single use case
- The activity diagram may replace the textual flow of activities, or supplement it

Activity diagram for use case *Create Customer Account*

shows the flow of activities between customer and system within this use case



Activity diagram notation



Activity diagram notation

1. **Initial node** - solid circle representing the start of the process.

2. **Actions** – rounded rectangles representing individual steps. The sequence of actions make up the total activity shown by the diagram.

3. **Flow** - arrows on the diagram indicating the progression through the actions. Most flows do not need words to identify them unless coming out of decisions.

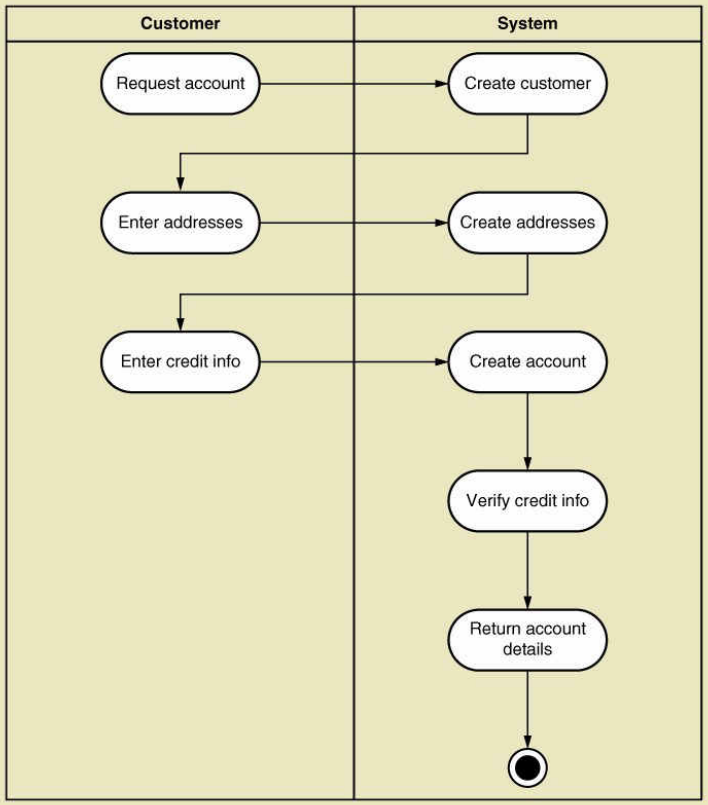
4. **Decision** - diamond shapes with one flow coming in and two or more flows going out. The flows coming out are marked to indicate the conditions.

Activity diagram notation (cont'd)

Murdoch UNIVERSITY

6. **Merge** - diamond shapes with multiple flows coming in and one flow going out. This combines flows previously separated by decisions. Processing continues with any one flow coming into the merge.
7. **Split**– a black bar with one flow coming in and two or more flows going out. Actions on parallel flows beneath the fork can occur in any order or concurrently.
7. **Join** – a black bar with two or more flows coming in and one flow going out, noting the end of concurrent processing. All actions coming into the join must be completed before processing continues.
8. **Activity final** – the solid circle inside the hollow circle representing the end of the process.

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.
1.1 Basic customer data are incomplete. 2.1 The address isn't valid. 3.2 Credit/debit information isn't valid.		

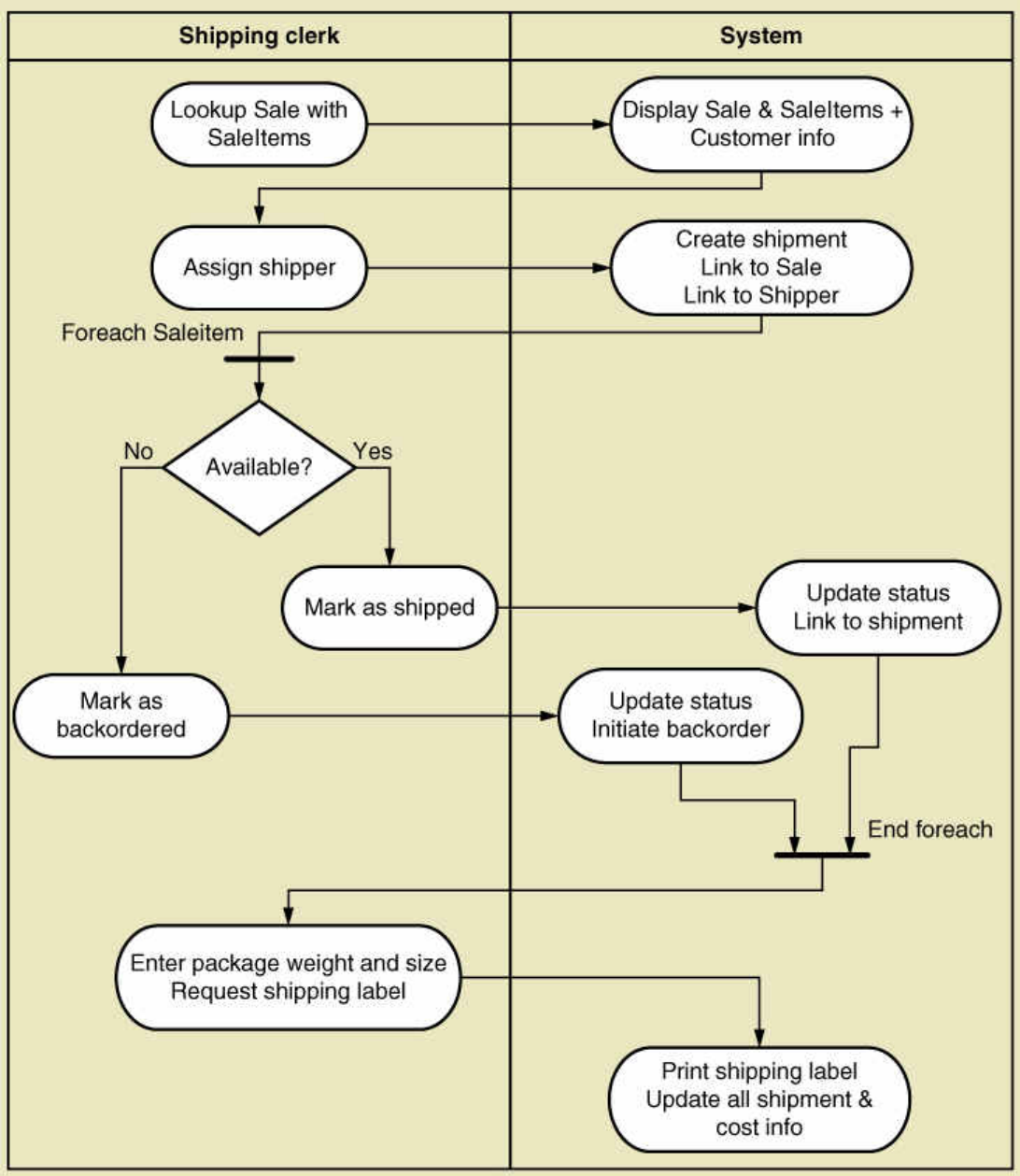


Activity diagram and equivalent description for *Create Customer Account*

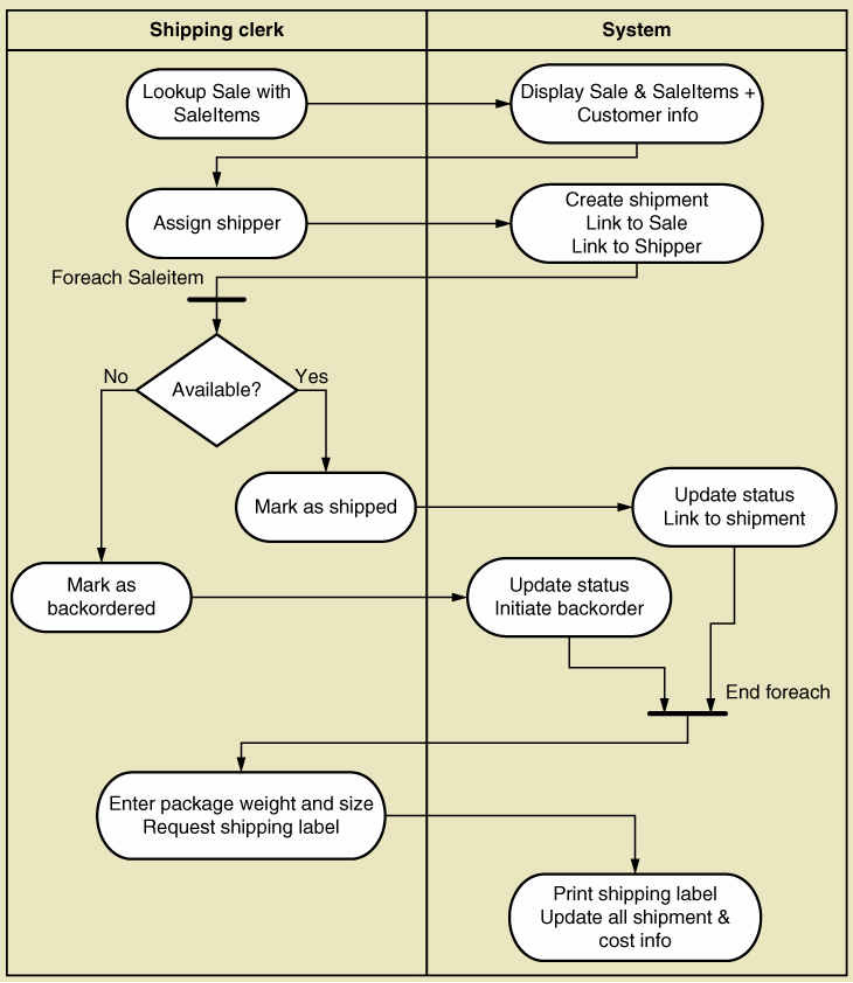
Activity diagram for use case *Ship Items*

Note:

- Synchronization bars for loop
- Diamond for decision point



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.
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.		



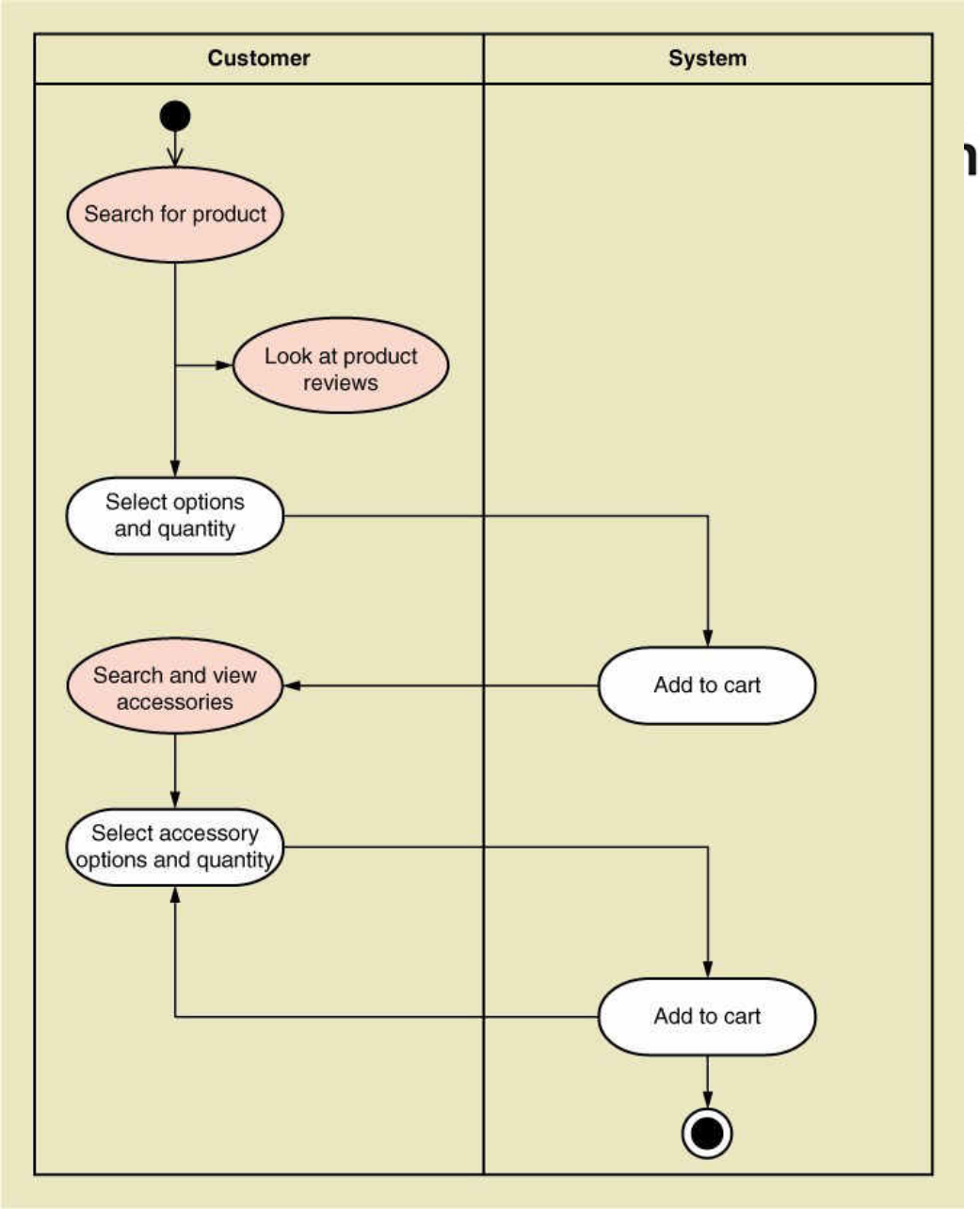
Ship items use case

Summing up…

- **Activity diagrams** are a diagrammatic method for representing activities in a sequence and the actor responsible for them
- Multiple actors, sequence, decisions, looping and parallel activities can all be represented
- They are useful for documenting the steps in complex use cases including the interaction between actor and system
- Note that activity diagrams are also useful in requirements gathering for capturing business workflow processes

Activity diagram for use case *Fill shopping cart*

This shows the flow of activities for Fill Shopping Cart use case, *plus* other use cases that are invoked (shown in shaded ovals)



System Sequence Diagram (SSD)

System sequence diagram

Murdoch UNIVERSITY

- **System sequence diagrams** (SSD) can supplement use case descriptions and activity diagrams
- Whereas activity diagrams and descriptions help the analyst understand the flow of activities, the SSD describes the associated *inputs* and *outputs* that are passed between the user and the system
- Shows sequence of interactions as ***messages*** during flow of activities
- System is shown as one object: a “black box”

SSD for *Create customer account* use case

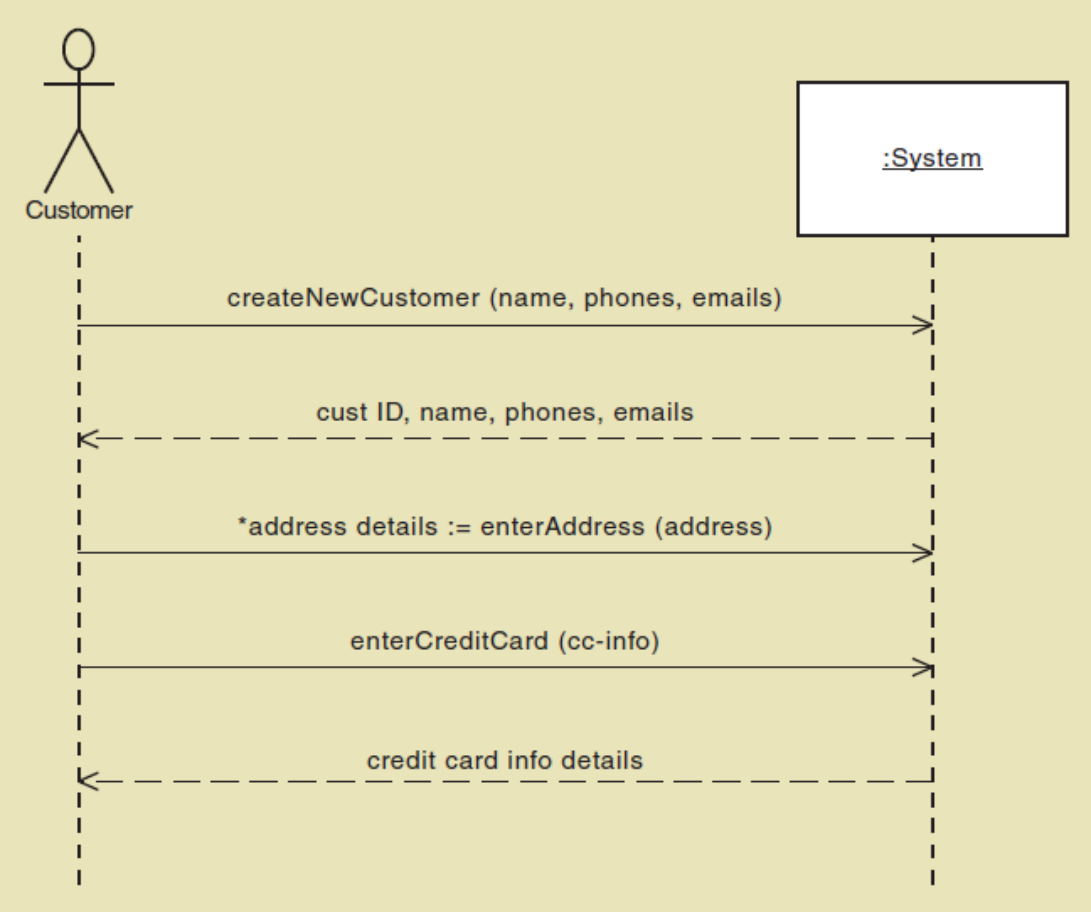


Image from: Systems Analysis and Design in a Changing World, 7th Edition ©2016. Cengage Learning

System sequence diagram

Murdoch UNIVERSITY

- Components: Actor, :System, object lifeline, messages
- Shows actor and one object, which represents the complete system
- Shows input and output messaging requirements for a use case
- Can be used to help develop user interface
- Is a special case of a UML *sequence diagram* (later topic) which also shows the internal classes inside the system

System sequence diagram (SSD) notation

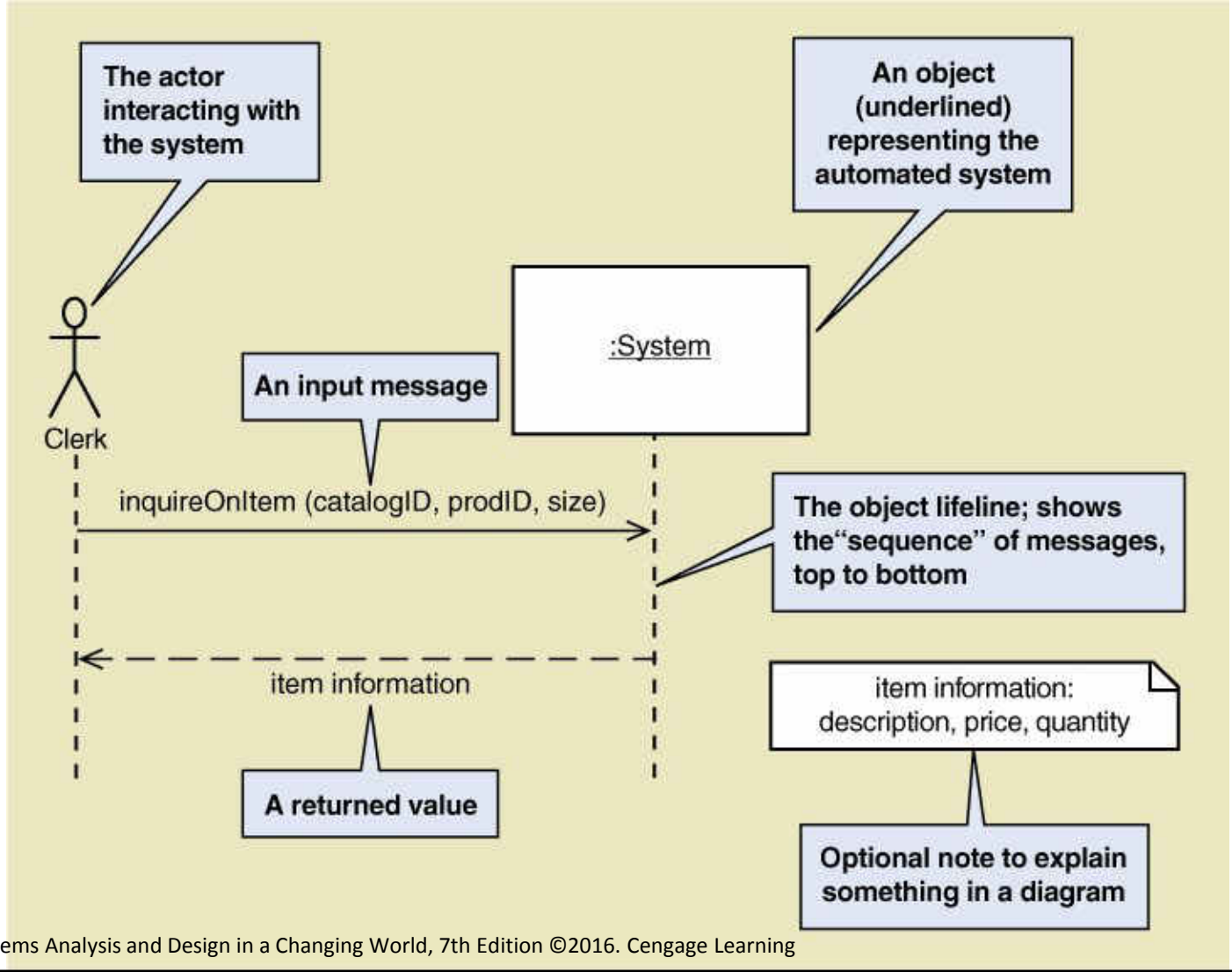


Image from: Systems Analysis and Design in a Changing World, 7th Edition ©2016. Cengage Learning

SSD message notation

Murdoch UNIVERSITY

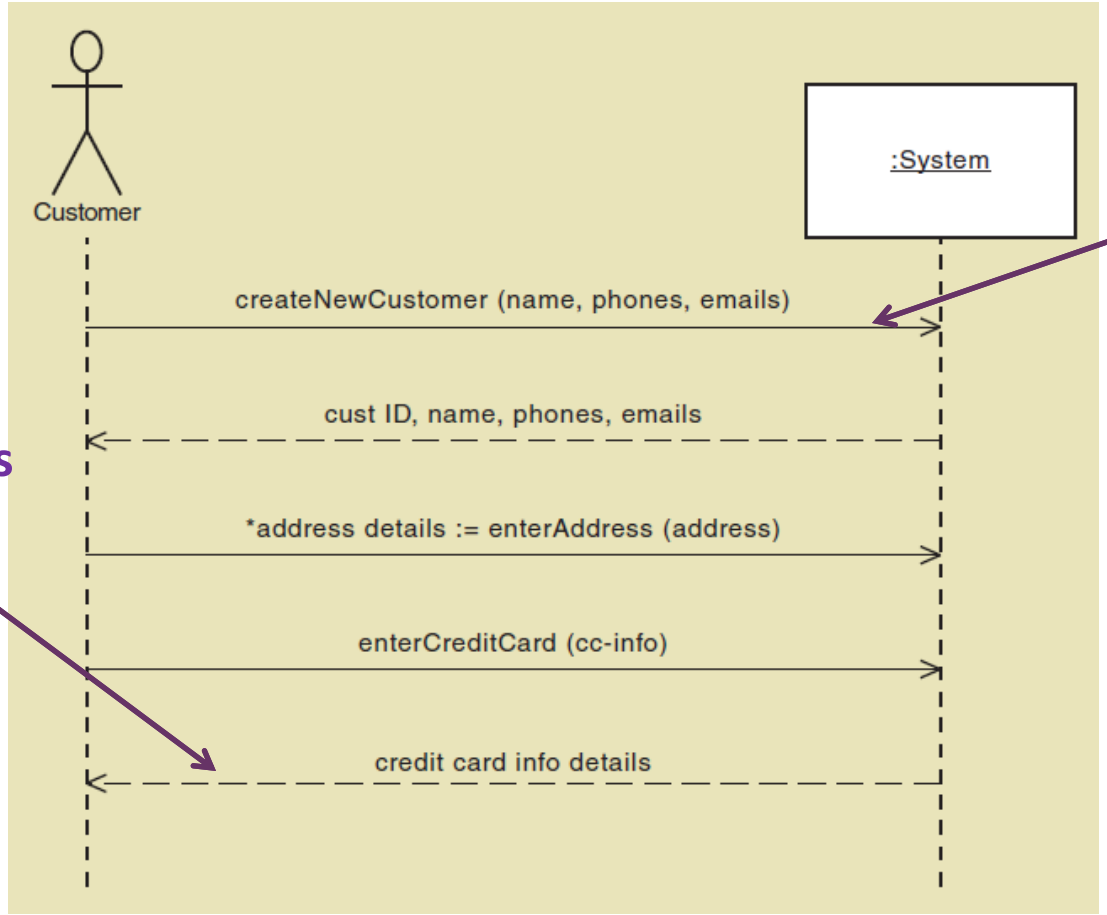
Input message:

- Solid line going from Actor to System
- Message name (verb-noun)
- Parameter list - input data (e.g. to identify particular item needed)

Return:

- Dashed line going from System to Actor
- No message name
- Returned value(s)

SSD for *Create customer account* use case



Input message

Return values

Image from: Systems Analysis and Design in a Changing World, 7th Edition ©2016. Cengage Learning

SSD message notation cont'd

Murdoch UNIVERSITY

Loop frame:

- can be used to indicate that a message is sent repeatedly

Opt frame:

- indicates that a message is optional based on some condition

Alt frame:

- enables if-then-else logic

SSD alternatives with looping

Notice that (a) and (b) are the same logic.

The alternative notation in (b) shows the looping, input and return messages in a single line

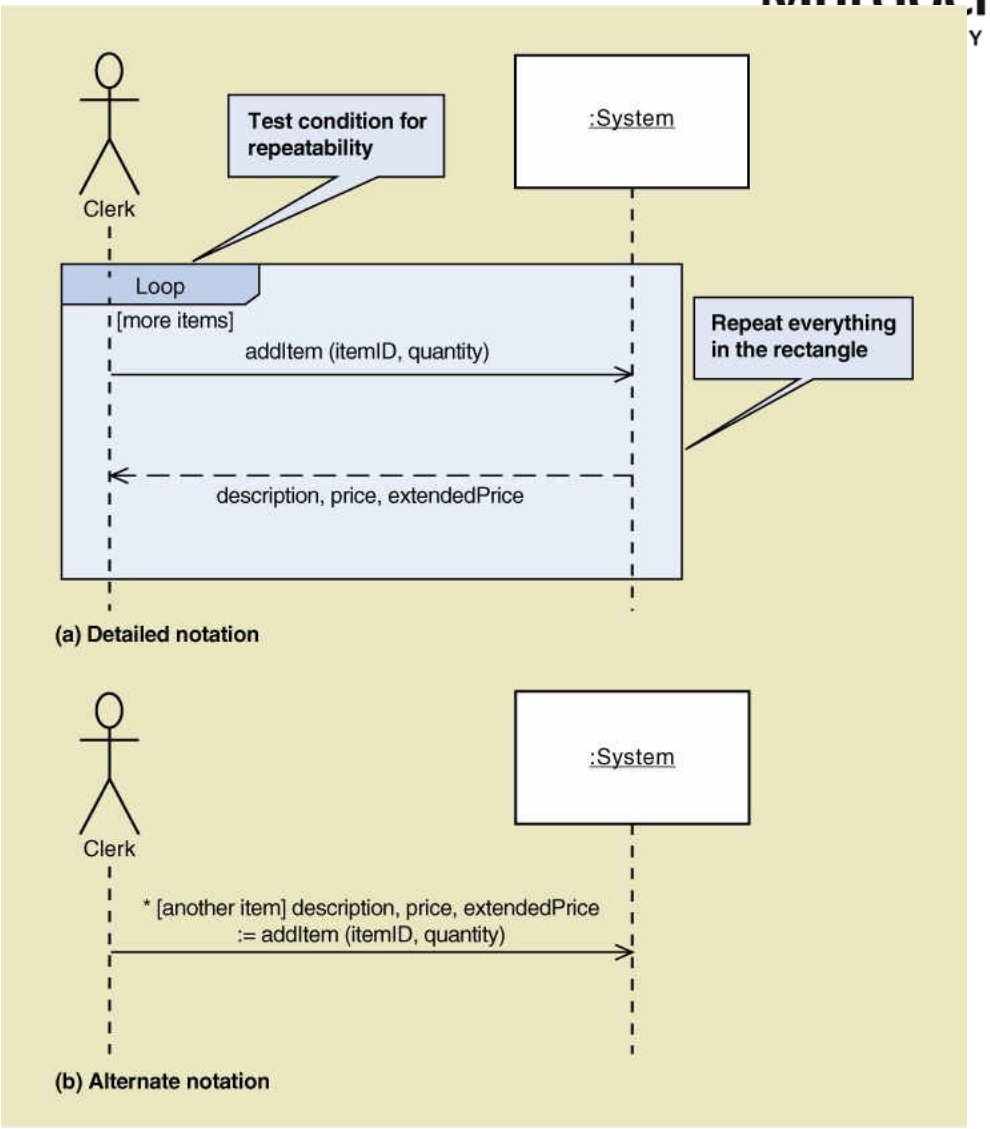
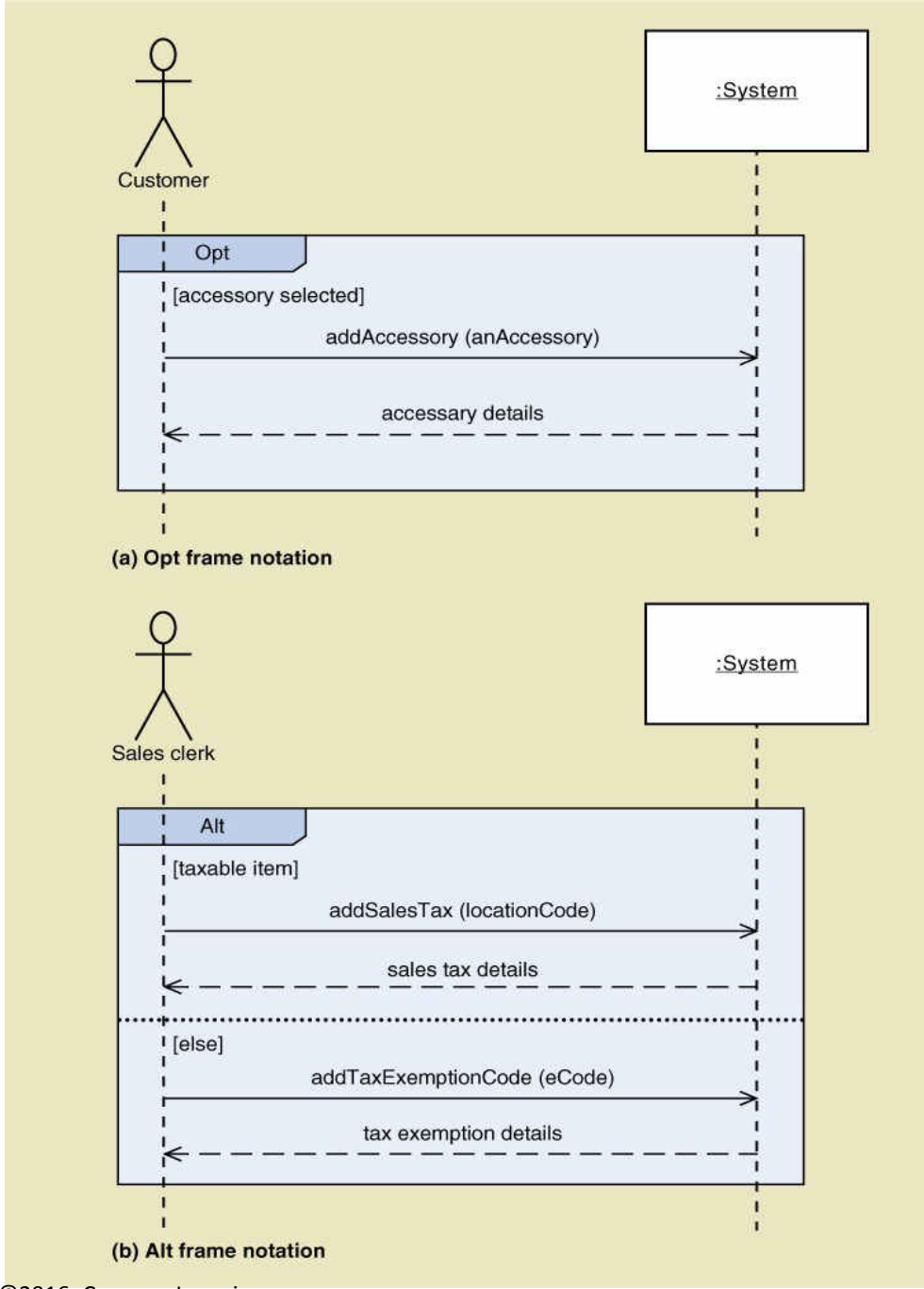


Image from: Systems Analysis and Design in a Changing World, 7th Edition ©2016. Cengage Learning

SSD examples

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



Message notation - complete

[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.

Text from p142 in Satzinger, J. Jackson, R. & Burd, S. (2016) Systems Analysis and Design in a Changing World, 7th edition. Course Technology, Thomson Learning.

Steps for developing a SSD

Murdoch UNIVERSITY

1. Identify input messages:
   - See use case flow of activities description or activity diagram
   - Wherever an arrow in the activity diagram crosses the automation boundary there will be a message

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 input parameters the system will need
   - These will likely be attributes from the class diagram

Steps for developing a SSD

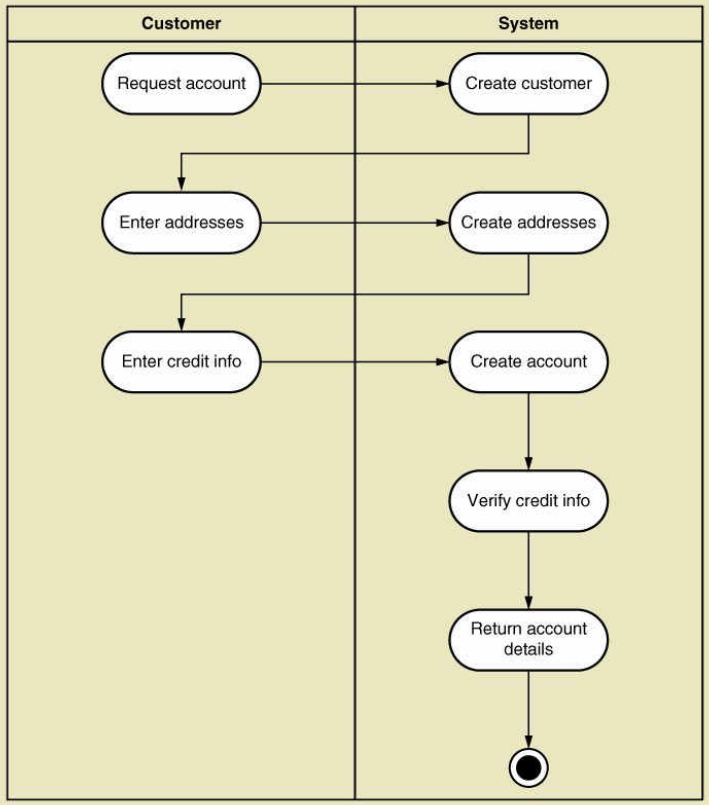
Murdoch UNIVERSITY

3. Identify any special conditions on input messages
   - Iteration/loop frame (or use * on the input message)
   - Opt or Alt frame

4. Identify and add output return values:
   - On message itself: aValue:= getValue(valueID)
   - As explicit return on separate dashed line

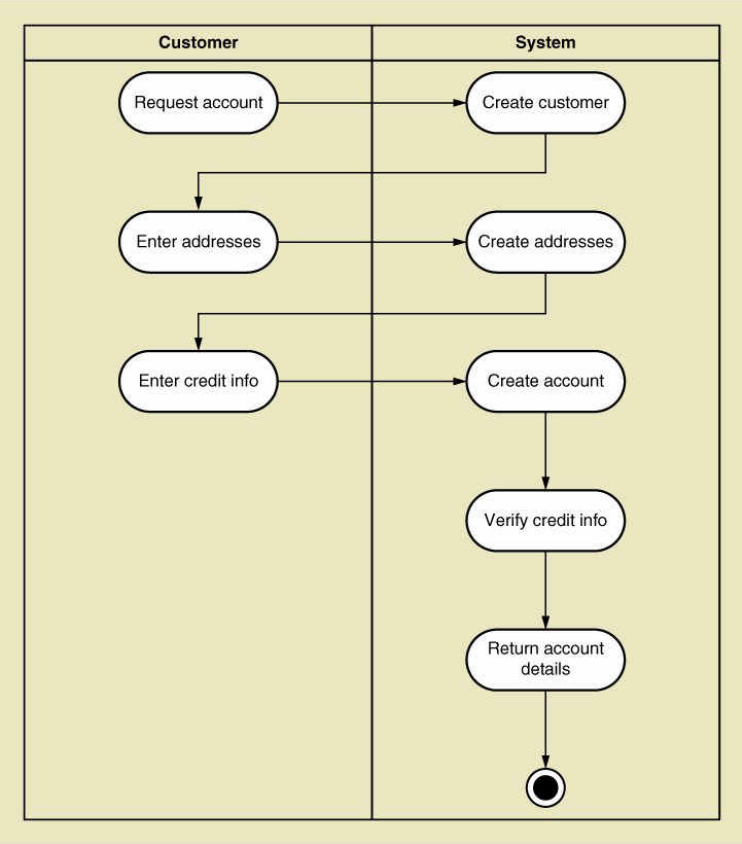
5. Check sequence of messages and returns is shown top-bottom and that nothing internal to the system object is shown

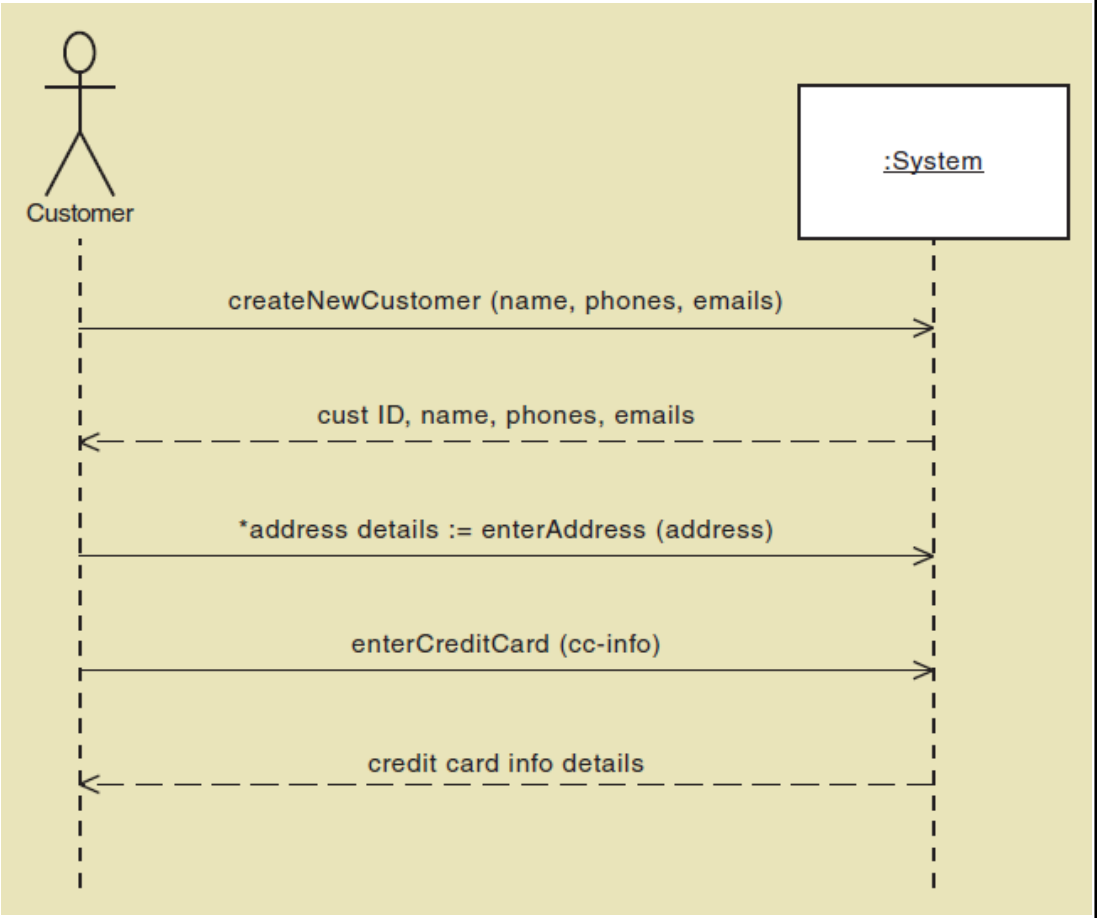
Flow of activities:	Actor	System
	1. Customer indicates desire to create customer account and enters basic customer information.	1.1 System creates a new customer. 1.2 System prompts for customer addresses.
	2. Customer enters one or more addresses.	2.1 System creates addresses. 2.2 System prompts for 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.
1.1 Basic customer data are incomplete. 2.1 The address isn't valid. 3.2 Credit/debit information isn't valid.		



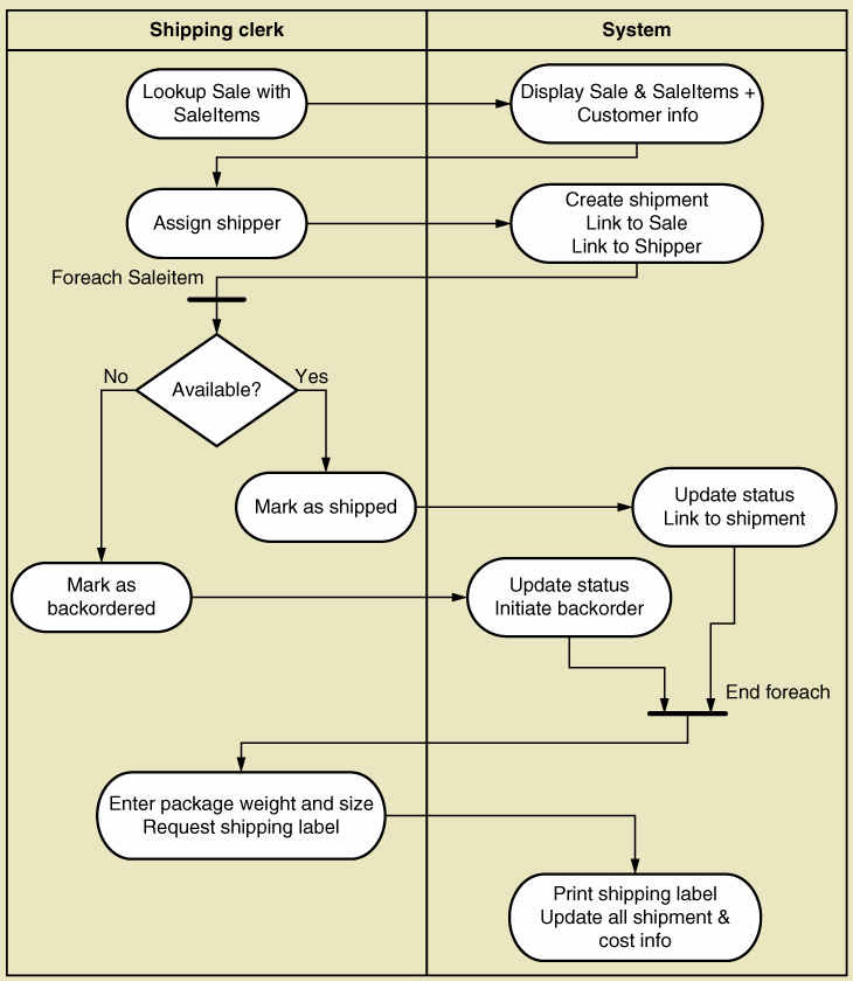
Activity diagram and equivalent description for *Create Customer Account*

SSD for *Create customer account* use case



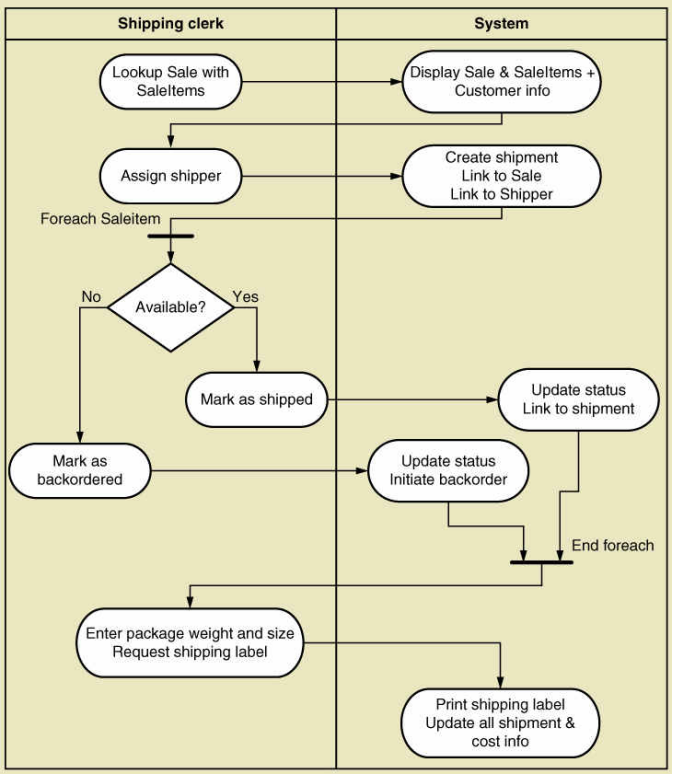


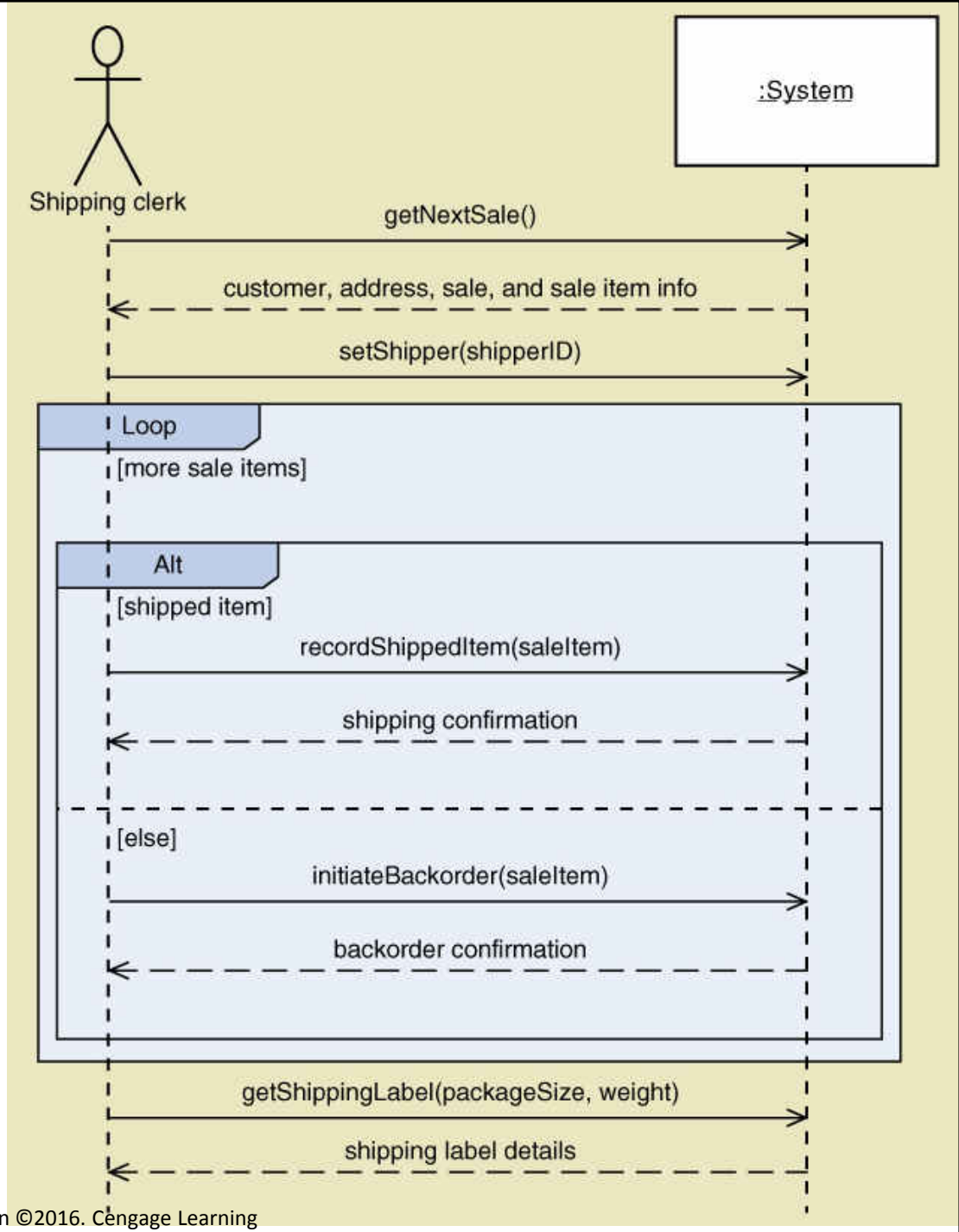
Flow of activities:	Actor	System
	1. Shipping requests sale and sale item information.	1.1 System looks up sale and returns customer, address, sale, and sales item information.
	2. Shipping assigns shipper.	2.1 System creates shipment and associates it with the shipper.
	3. For each available item, shipping 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.
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.		



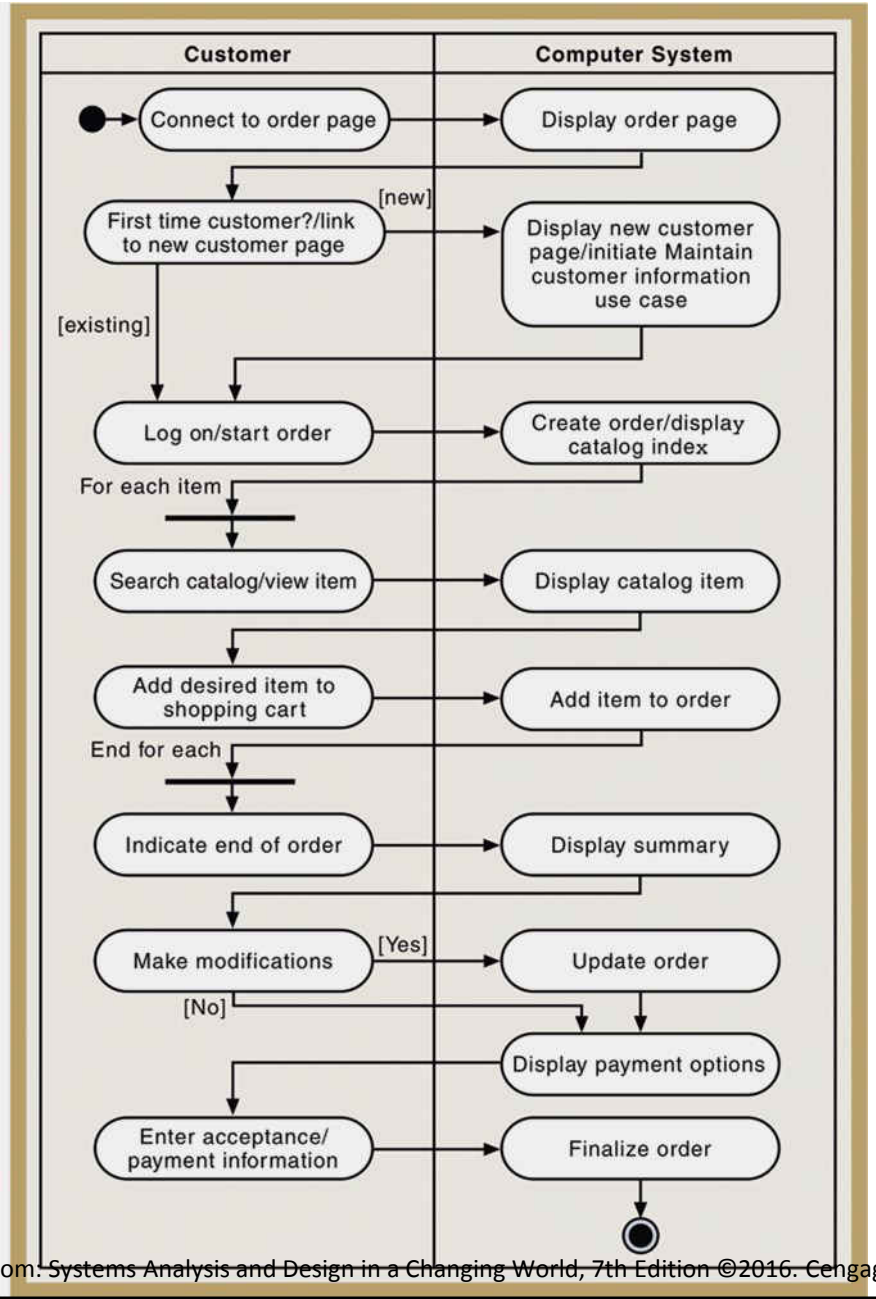
Ship items use case

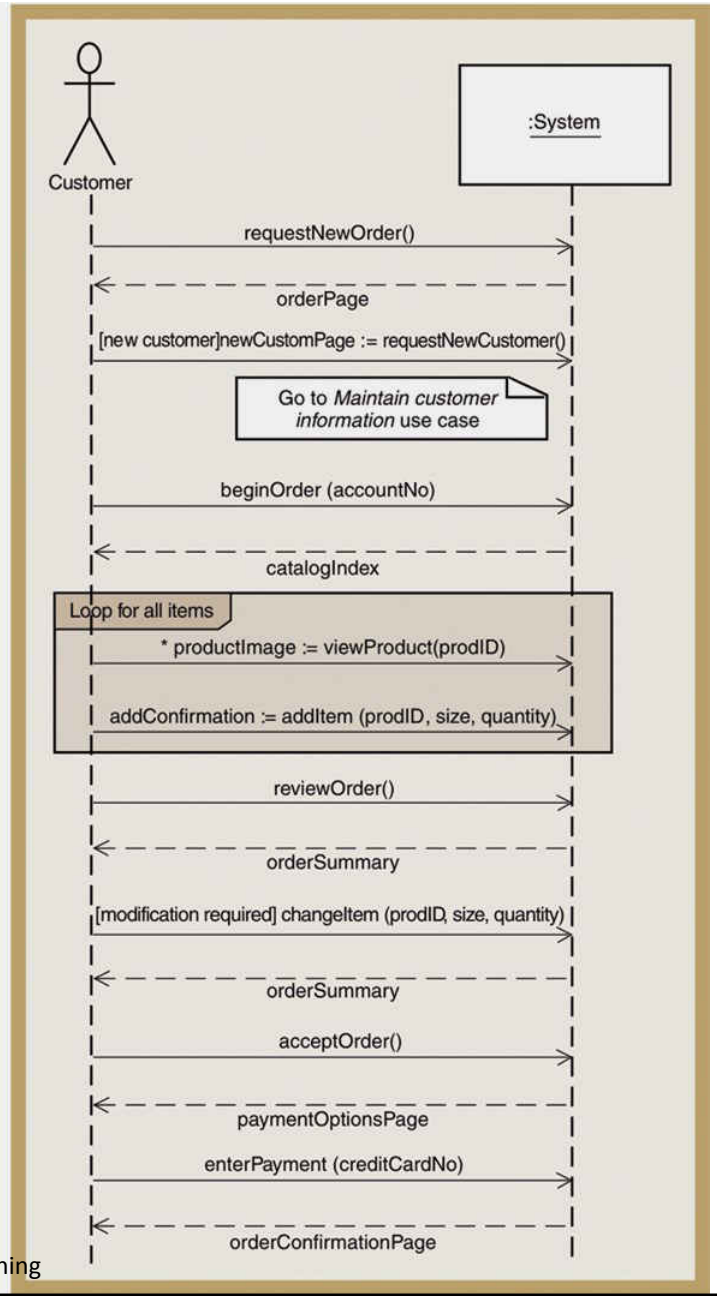
SSD for *Ship items* Use Case





Activity diagram and SSD for Web Order





Summing up…

- System Sequence Diagrams (SSD) document the input and output messaging requirements for a use case
- A SSD can be developed readily from the flow of activities section in the use case description
- The messages from actor to system show any information that is passed to the system, and return messages are passed back from the system
- The system itself is treated as a single object 'black box' and no internal workings are shown
- The SSD is expanded in the system design phase to a Sequence Diagram that shows the interaction between objects within the system

State Machine Diagrams – modelling object behaviour

Murdoch UNIVERSITY

Object behaviour – states and transitions

Murdoch UNIVERSITY

- Some objects (not all) have a life cycle with state conditions that change and should be tracked. For example -
- A Student in a unit can be in any of several possible *states*: Enrolled, Invalid, Discontinued, Completed
- The various use cases that involve a Student object can move it from one state to another, e.g. 'Withdraw from unit' would transition it from the Enrolled to the Discontinued state
- Here, the particular state for the student is recorded as a *value* in attribute 'Status'

Example: SMD for a 'SaleItem'

- A SaleItem can be in any of four states: open, on back order, ready to ship, shipped

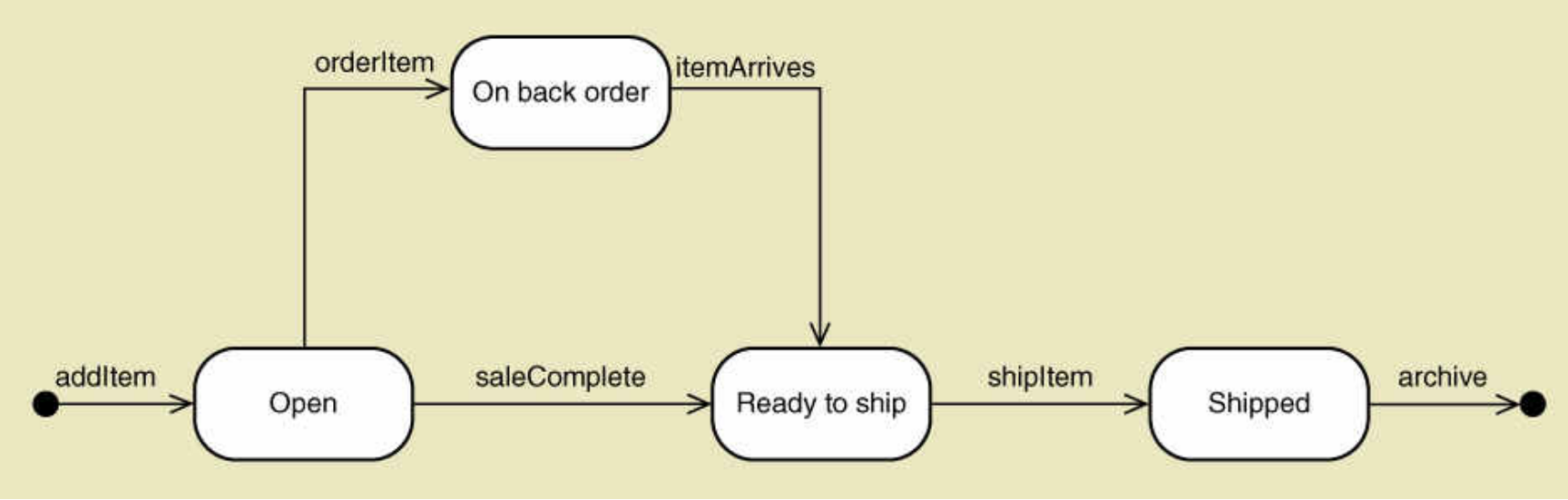


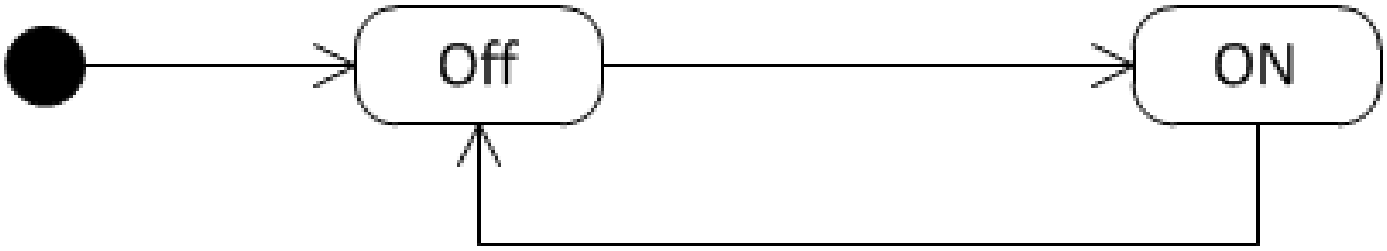
Image from: Systems Analysis and Design in a Changing World, 7th Edition ©2016. Cengage Learning

State Machine Diagram (SMD)

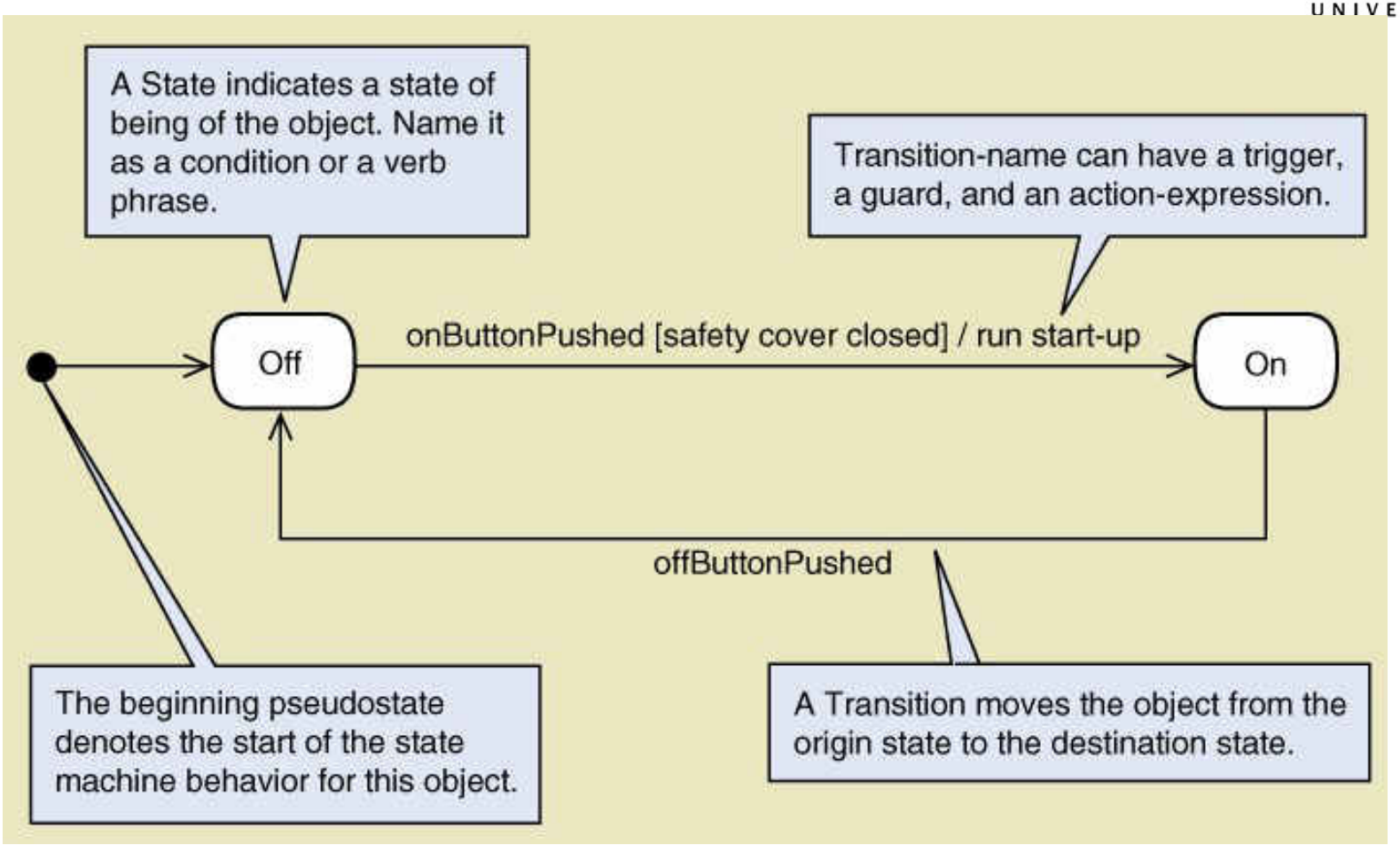
- A **State Machine Diagram (SMD)** is a diagram that shows the possible behaviour of an object with states and transitions
- **State** – a condition during an object's life when it satisfies some criterion, performs an action, or waits for an event
- **Transition** – the movement of an object from one state to another
- (SMD is also called a State Transition Diagram)

State Machine Diagram components

- **Origin state** – the original state of an object before it begins a transition
- **Destination state** – the state to which an object moves after completing a transition
- **Transition** – moves the object from the origin state to the destination state
- **pseudostate** – the starting point in a state machine diagram



State Machine Diagram for a printer



Syntax of transition statement

transition-name (parameters, …) [guard-condition] / action-expression

State Machine Diagram components

- **Transition name** – what causes the transition to occur

  OnButtonPushed

- **guard-condition** – a true/false test to see whether a transition can fire

  SafetyCoverClosed

- **action-expression** – some activity that must be completed as part of a transition

  RunSelfTest

Any of these may be empty, although there is usually a transition name

Example: SMD for a 'SaleItem'

- A SaleItem can be in any of four states: open, on back order, ready to ship, shipped

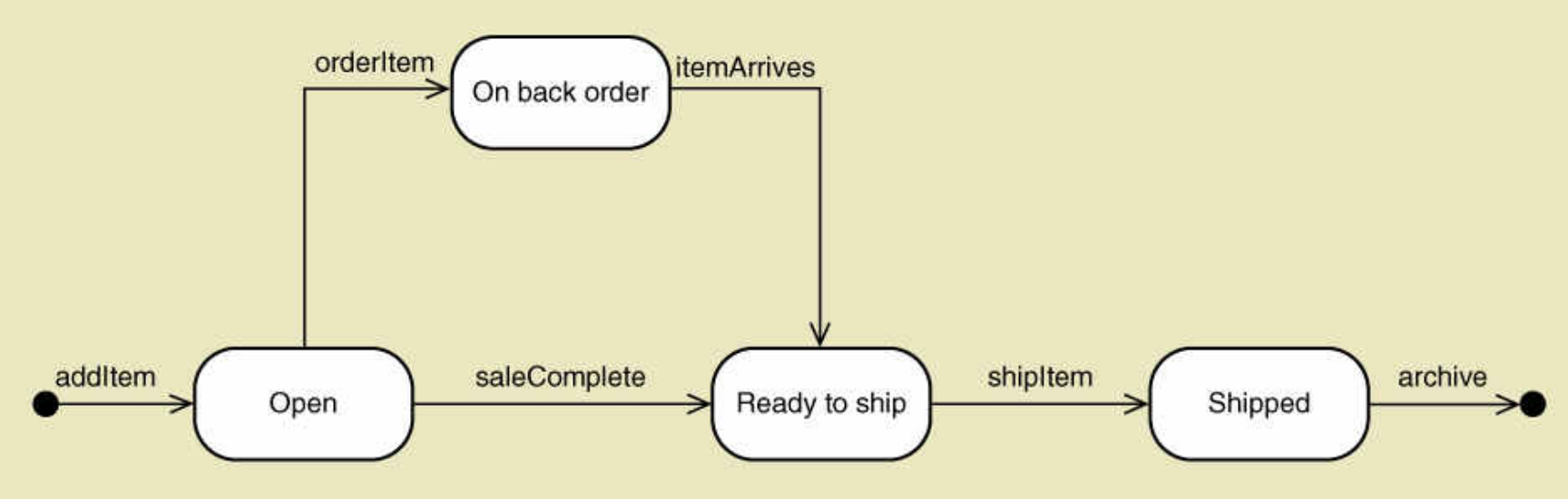


Image from: Systems Analysis and Design in a Changing World, 7th Edition ©2016. Cengage Learning

Creating a State Machine Diagram - steps

Murdoch UNIVERSITY

1. Review the class diagram and select classes that might require state machine diagrams
2. For each class, make a list of status conditions (states) you can identify
3. Begin building diagram fragments by identifying transitions that cause an object to leave the identified state
4. Sequence these states in the correct order and aggregate combinations into larger fragments
5. Review paths and look for independent, concurrent paths

Example: steps in creating a State Machine Diagram for RMO 'SaleItem'

Murdoch UNIVERSITY

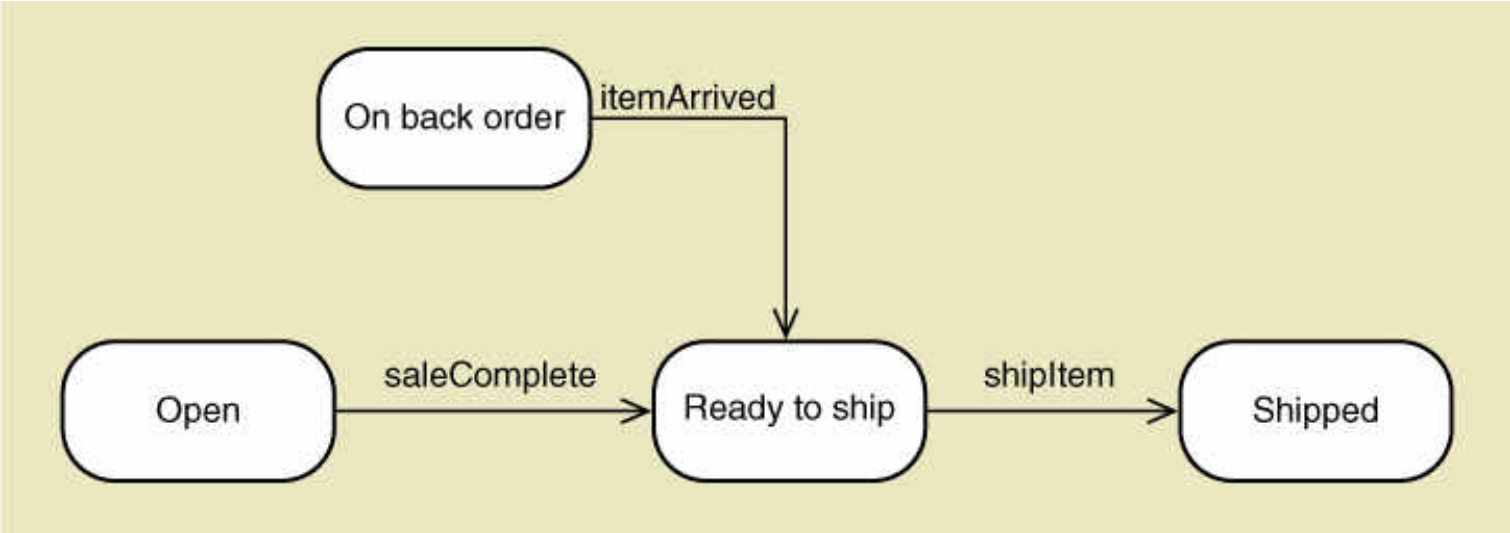
1. Choose SaleItem. It has status conditions that need to be tracked: ready to ship, etc
2. List the states and exit transitions

State	Transition causing exit
Open	saleComplete
Ready to Ship	shipItem
On back order	itemArrived
Shipped	No exit transition defined

Example: steps in creating a State Machine Diagram for RMO 'SaleItem'

Murdoch UNIVERSITY

3. Build fragments – see figure below
4. Sequence in correct order – see figure below
5. Look for concurrent paths – none



Creating a State Machine Diagram – steps cont'd

Murdoch UNIVERSITY

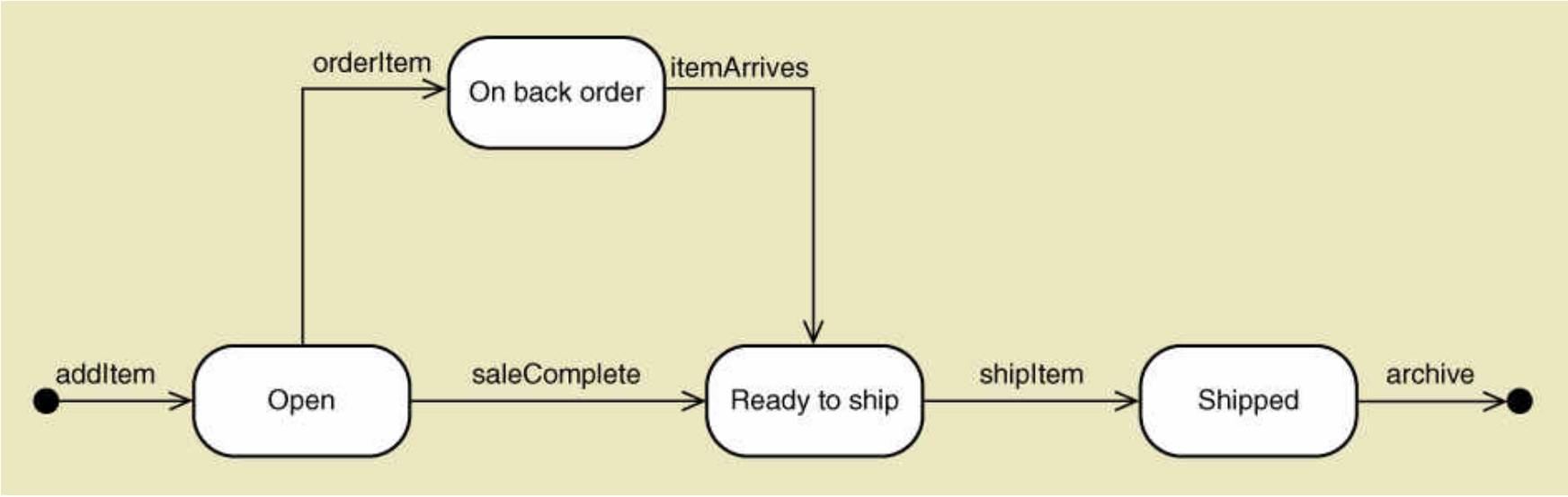
6. Look for additional transitions and test both directions
7. Expand each transition with appropriate message event, guard condition, and action expression
8. Review and test the state machine diagram for the class
   - Make sure state are really states for the object in the class
   - Follow the life cycle of an object coming into existence and being deleted
   - Be sure the diagram covers all exception condition
   - Look again for concurrent paths and composite states

Example: steps in creating a State Machine Diagram for RMO 'SaleItem'

Murdoch UNIVERSITY

6. Add other required transitions
7. Expand with guard, action-expressions etc.
8. Review and test

Below is the final State Machine Diagram

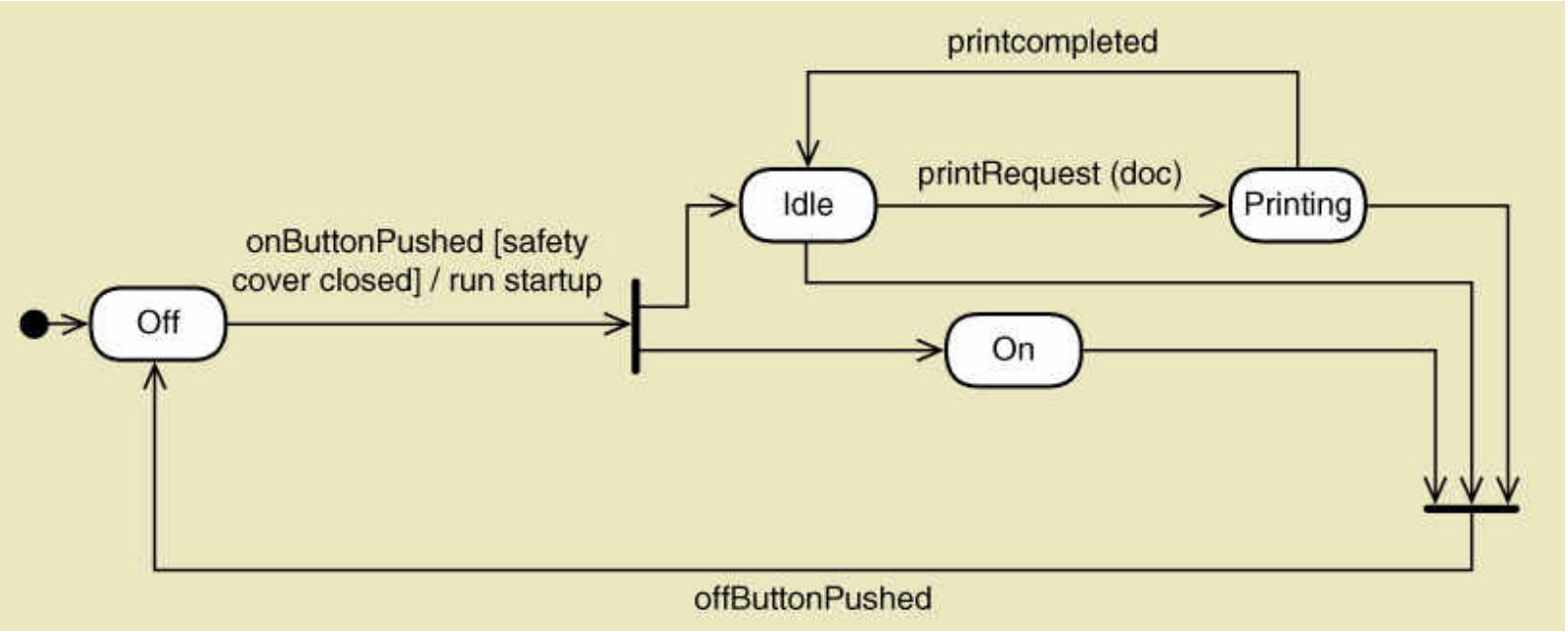


Concurrency in a State Machine Diagram

Murdoch UNIVERSITY

- **Concurrent states** – when an object is in one or more states at the same time
  - e.g. a printer can be both On and Idle
- **Path** – a sequential set of connected states and transitions
- **Concurrent paths** – when multiple paths are being followed concurrently, i.e. when one or more states in one path are parallel to states in another path

Example: Printer with concurrent paths



- Concurrent paths often shown by synchronization bars (similar to Activity Diagram)
- Multiple exits from a *synchronization bar* is an "AND" condition (printer is On and Idle)
- Multiple exits from a *state* is an "OR" – the object follows only one of the paths

Summing up…

- Some objects have a life cycle with status conditions that change and should be tracked
- These status conditions are part of the business requirements of the system – e.g. an order can be dispatched, on back order, etc
- State Machine Diagrams (SMD) are used to document the behaviour of these objects
- SMDs show the *states* an object can be in, and the *transitions* that cause it to move from one state to another
- SMD thus add further detail to the domain modelling side of the analysis

Extending and integrating the requirements models

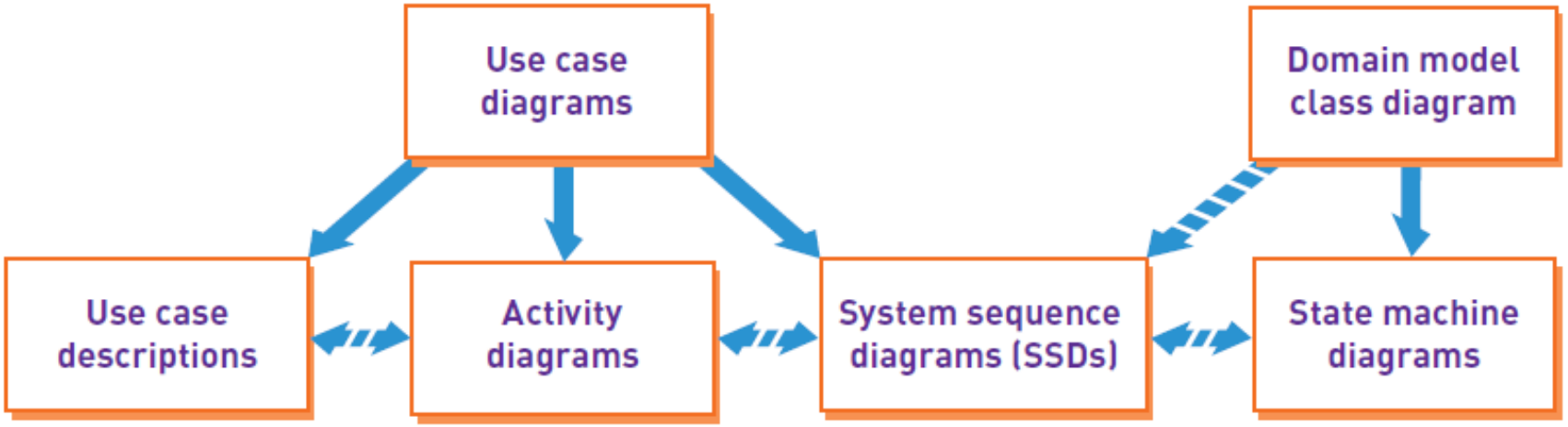


Image from: Systems Analysis and Design in a Changing World, 7th Edition ©2016. Cengage Learning

Topic learning outcomes revisited

After completing this topic you should be able to:

- Explain how additional information about use cases can be represented in detail
- Create a **CRUD** table (CRUD matrix) to verify use cases against the domain model
- Interpret and write **fully developed use case descriptions**
- Develop **activity diagrams** to document the flow of activities within a use case
- Develop **system sequence diagrams** to model the interaction between actors and the system
- Develop **state machine diagrams** to model object behavior

Murdoch UNIVERSITY

What's next?

In the next tutorial, we'll continue applying various techniques to extend the requirements models.

The models we've discussed in this topic will form a basis for the design models that we will go on to create. Before that, though, we'll give a brief overview of the systems design phase and the activities in it.

Murdoch UNIVERSITY