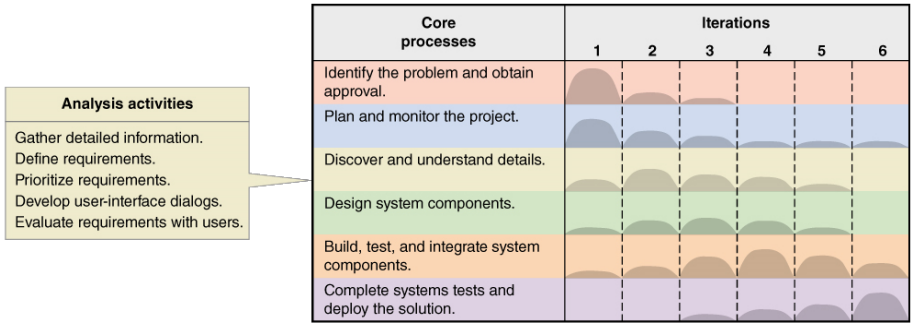


System Analysis Activities-

More About Use Cases – Use Case Modeling

Dr. Yuehua Wang
yuehua.wang@tamuc.edu

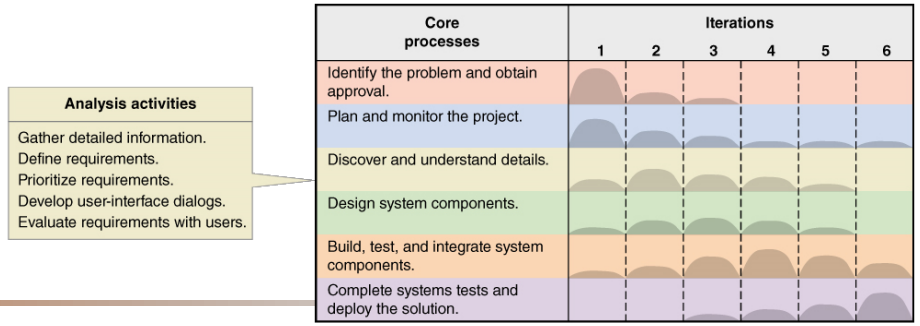
TEXAS A&M UNIVERSITY COMMERCE

MORE

Learning Objectives

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

Why?



- The main objective of defining requirements in system development is
  - understanding users' needs
  - how the business processes are carried out, and
  - How the system will be used to support those business processes.
- What we have learned
  - Activates for discovering and understanding the requirements of a new system
  - User stories and use cases
    - *User stories are sometimes used in place of use cases with Agile development*
    - *Use cases are identified by using the user goal technique and the event decomposition technique.*
    - *UML Use case diagrams to show actors, use cases, and interactions*

- An information system also needs to record and store information about the business processes.

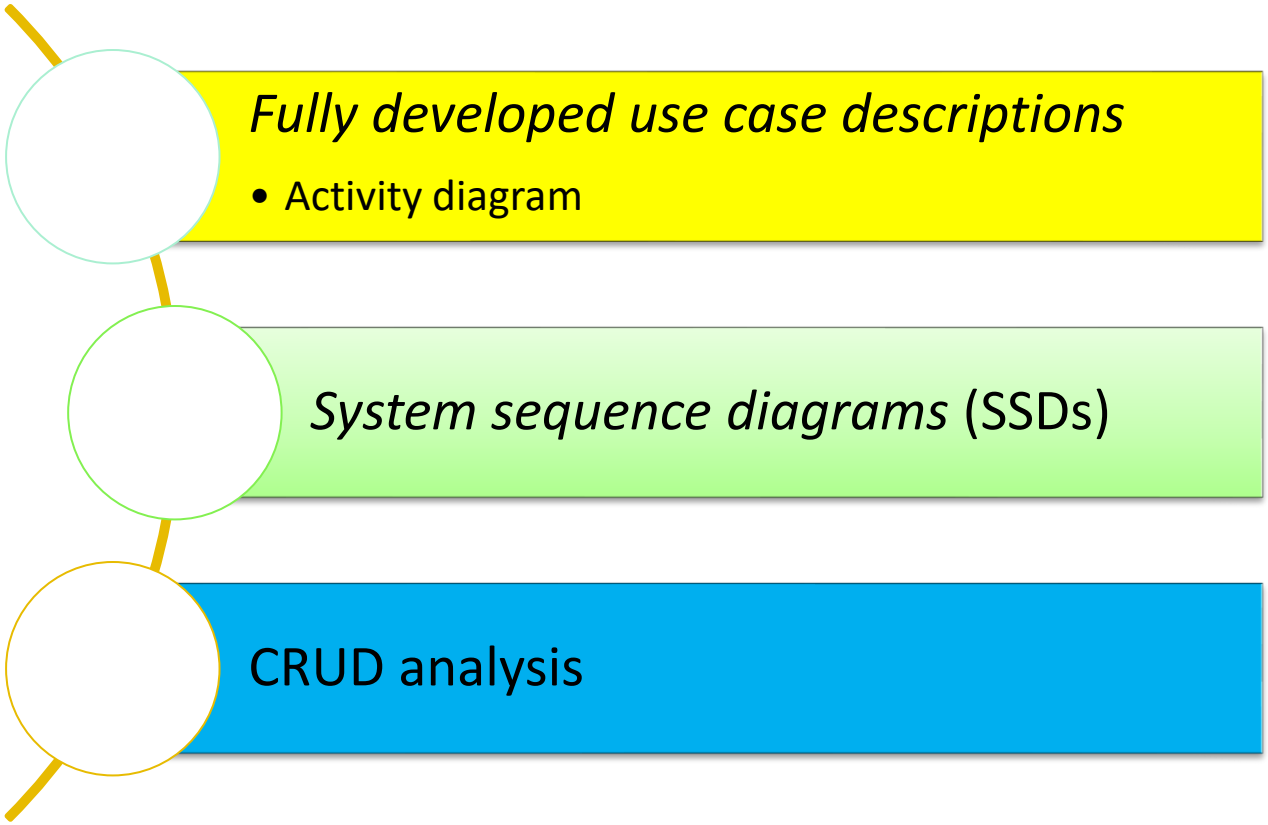
Overview

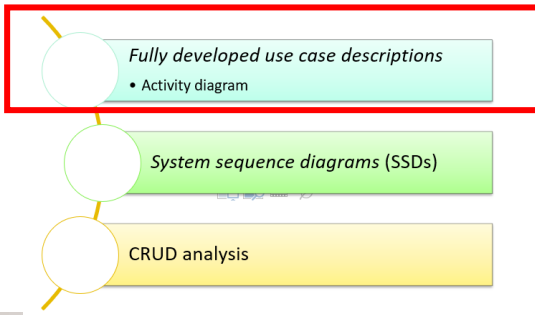
- An information system also needs to record and store information about the business processes.
  - In a manual system, the information is recorded on paper and stored in a filing cabinet.
  - In an automated system, the information is stored in electronic files or a database.
  - The information storage requirements of a system are documented in UML use case diagrams and
    - fully developed use case descriptions, UML activity diagrams, UML system sequence diagrams (SSDs) , and CRUD analysis
    - learn additional techniques and models that will allow you to extend the requirements models to show additional information about the use cases for the system.

Overview (continued)

- *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 can also 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

Not all use cases are modelled at the same level of detail. Only model when there is complexity and a need to communicate details


Fully developed use case descriptions
• Activity diagram
System sequence diagrams (SSDs)
CRUD analysis



Use Case Descriptions

- Use case description
  - a textual model that lists and describes the processing details for a use case
- Depending on an analyst's needs, use case descriptions tend to be written at two separate levels of detail:
  - brief description and fully developed description.
    - Write a *brief description* as shown below 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.

Fully Developed Use Case Descriptions

- The fully developed description is the most formal method for documenting a use case.
  - Note that one of the major difficulties for software developers is that they often struggle to obtain a deep understanding of the users' needs.
  - But if you create a fully developed use case description, you increase the probability that you thoroughly understand the business processes and the ways the system must support them.

Fully Developed 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

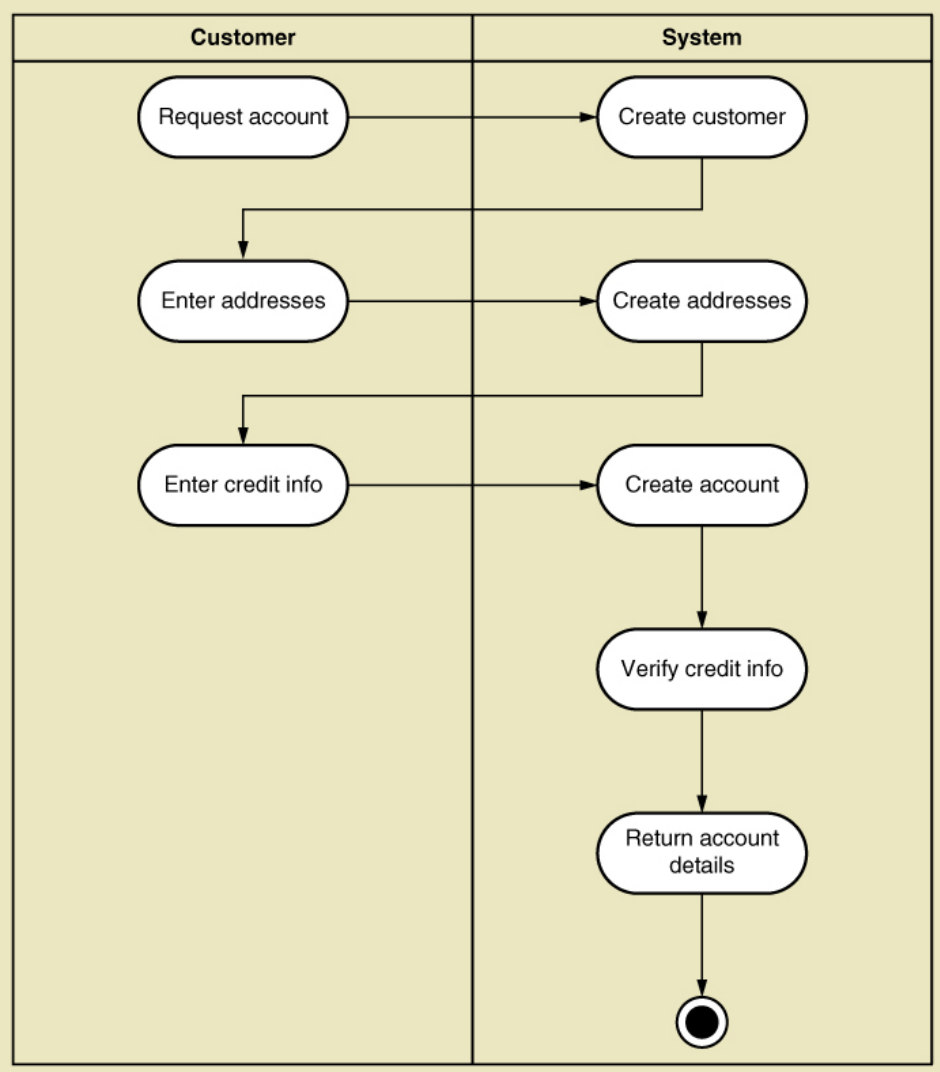
Use Case Description Details

- 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

UML Activity Diagram for Use Case

- *Create Customer Account*
- Note: this shows flow of activities only

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.	



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.

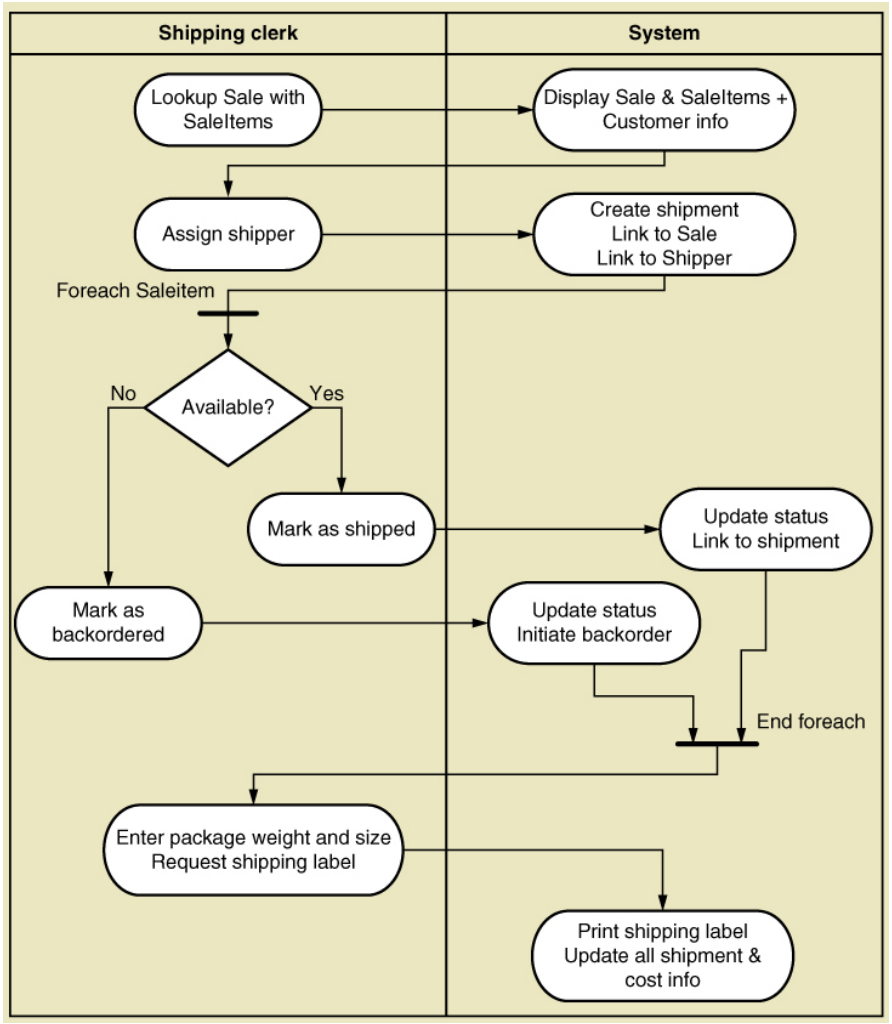
Fully Developed Use Case Description
Ship items (part 2)

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.	

Activity Diagram for *Ship Items* Use Case

- Note:
  - Synchronization bar for loop
  - Decision diamond

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

- *Fill shopping cart*

- Note: this shows use case with <<includes>> relationship or <<uses>> relationship

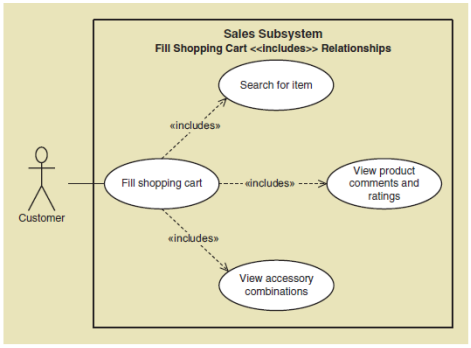
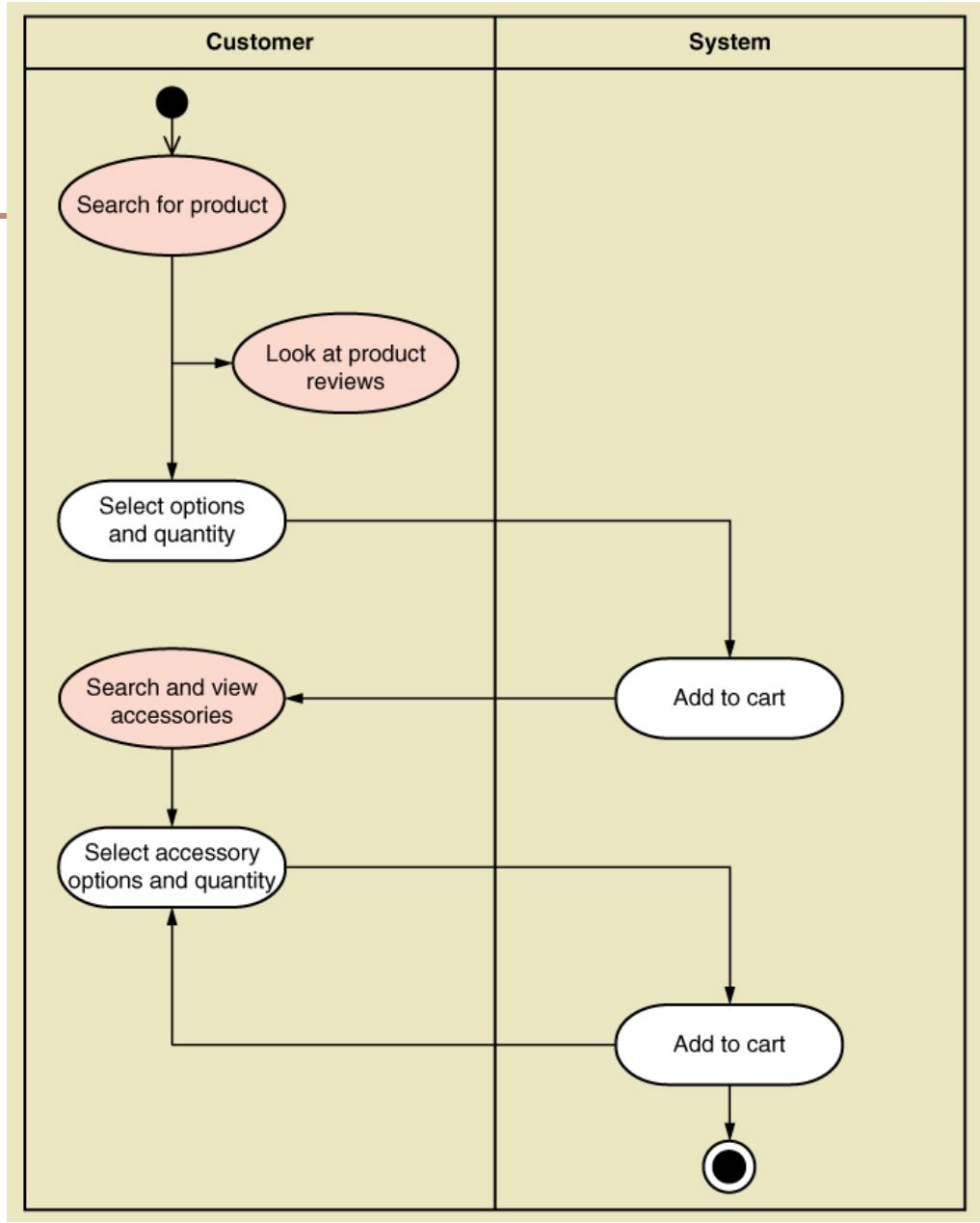
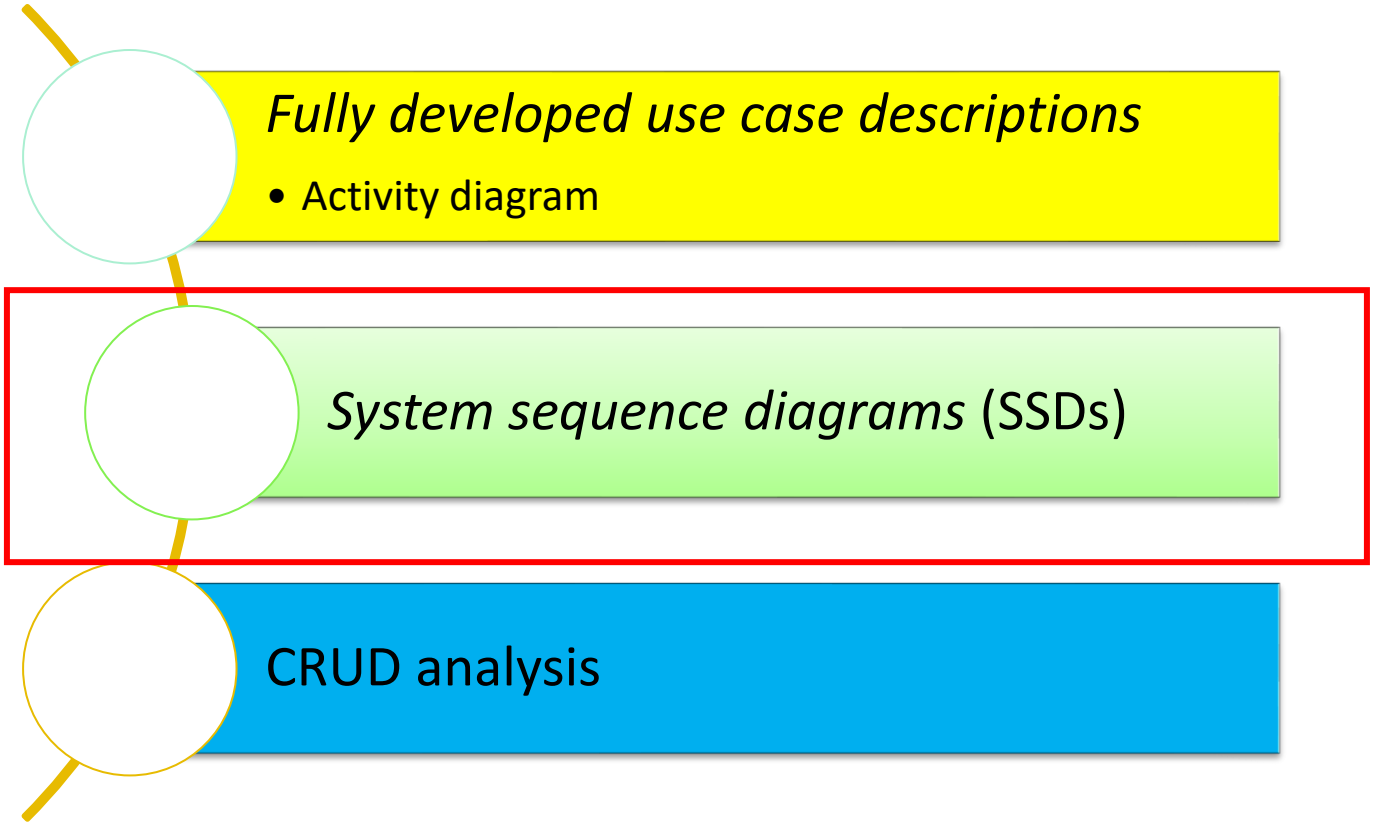
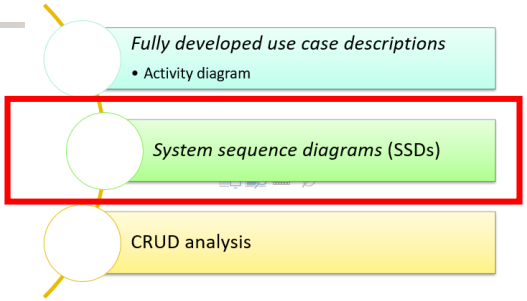


Figure 3-15




Fully developed use case descriptions
• Activity diagram
System sequence diagrams (SSDs)
CRUD analysis

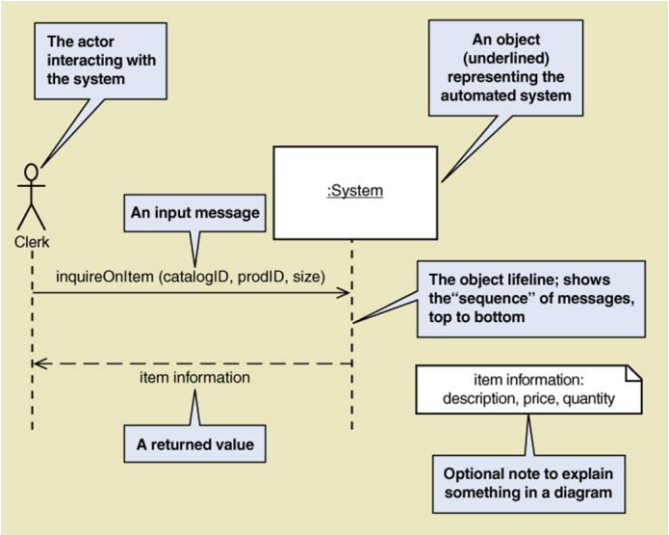
System Sequence Diagram (SSD)



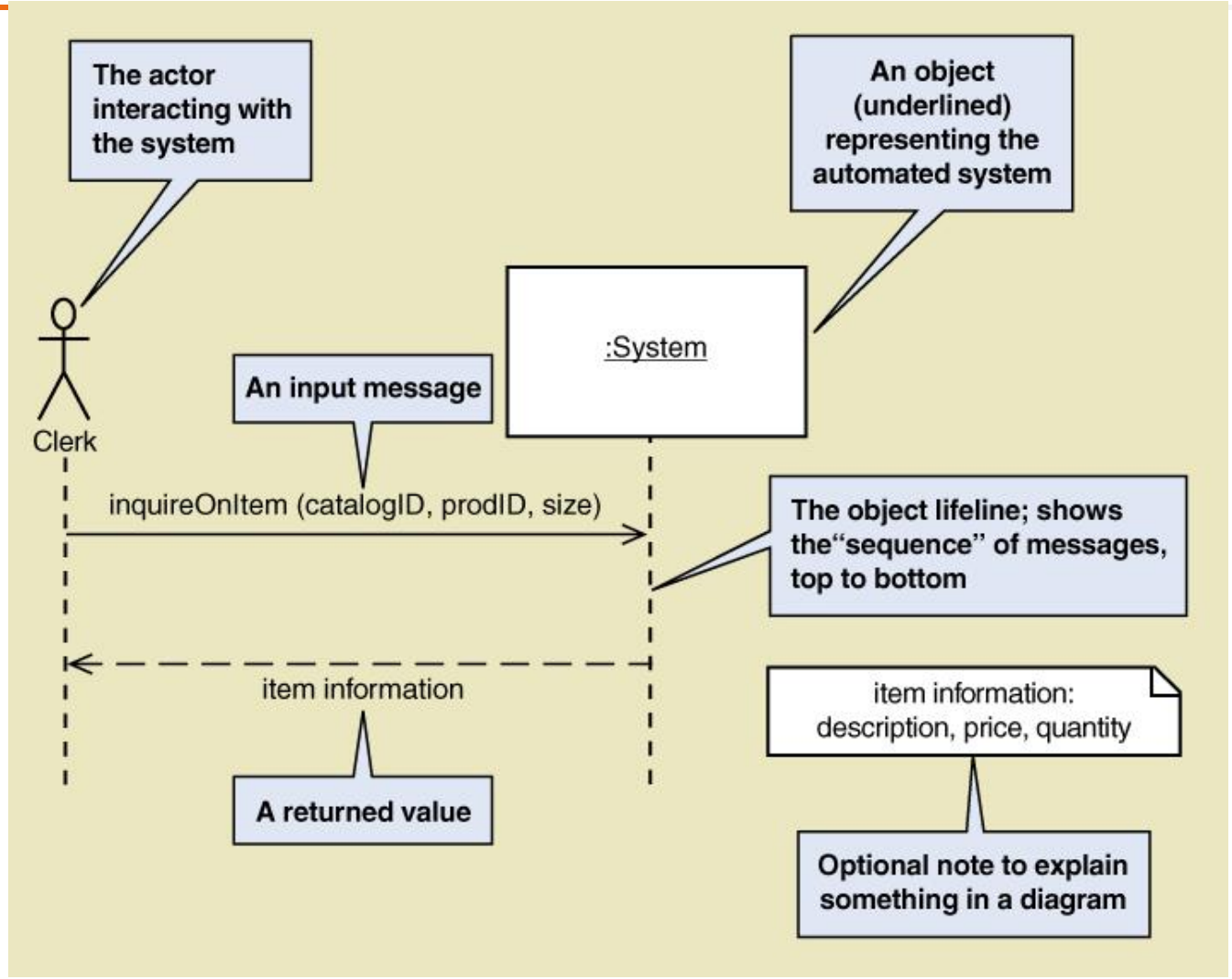
- A special type of UML sequence diagram
- Is used to
  - describe this flow of information into and out of the automated portion of the system
  - identifies the interaction between actors and the system
- is an effective tool to help in the initial design of the user interface by identifying the specific information that flows from the user into the system and the information that flows out of the system back to the user.

System Sequence Diagram (SSD)

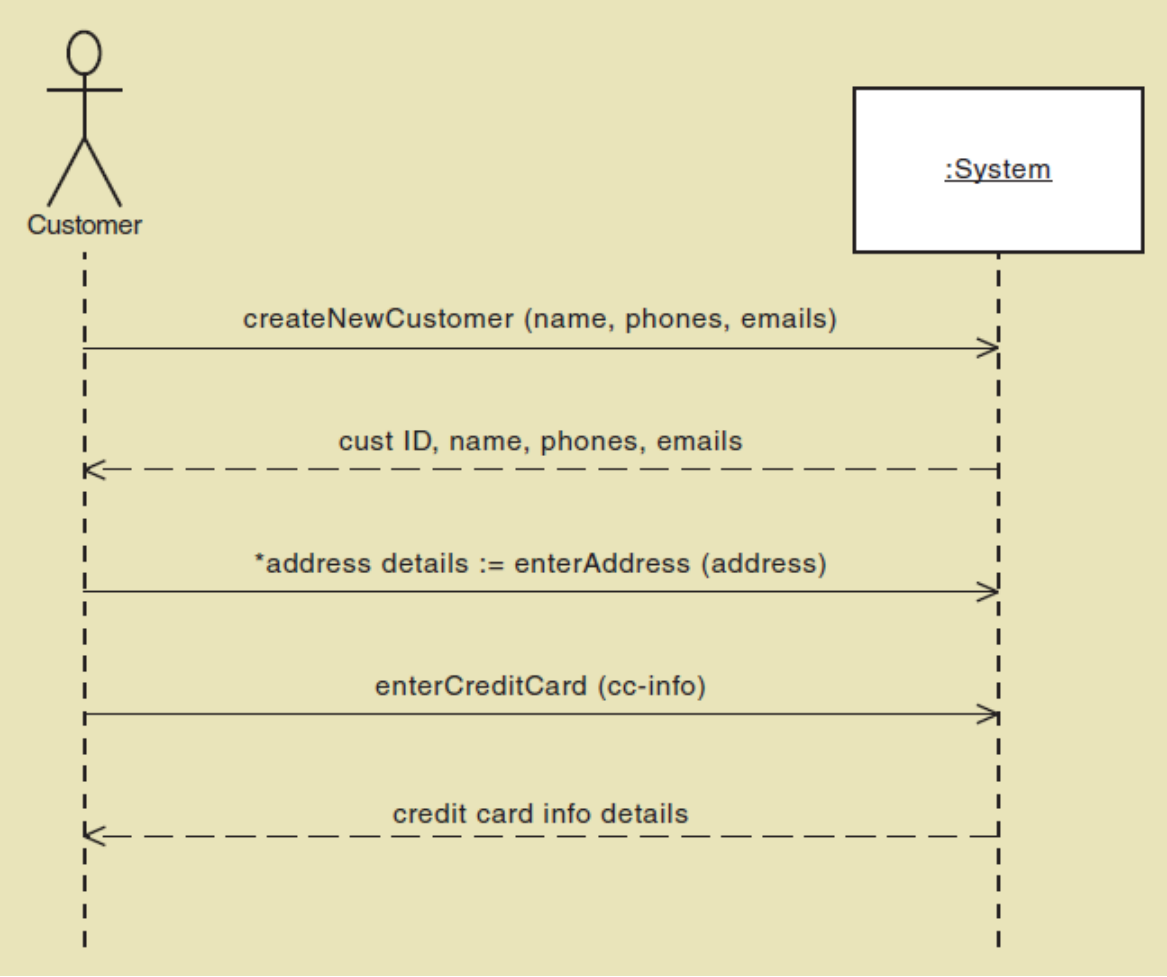
- Special case for a sequence diagram
  - Only shows actor and one object
  - The one object represents the complete system
  - Shows input & output messaging requirements for a use case
- Actor, :System, object lifeline
- Messages



System Sequence Diagram (SSD) Notation

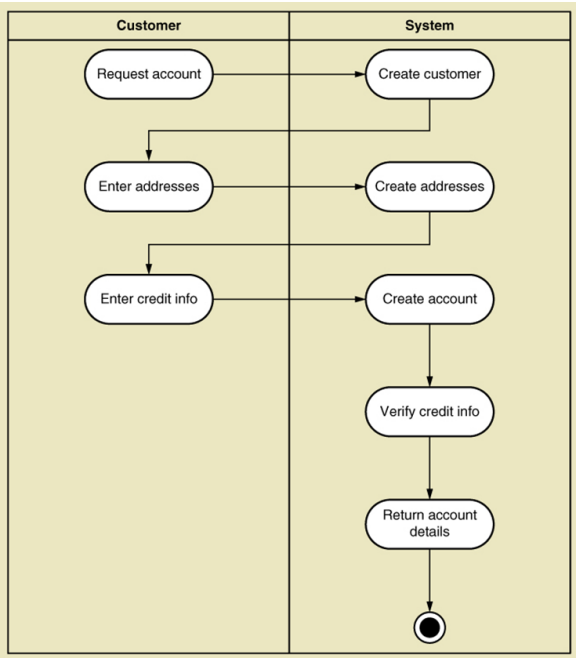


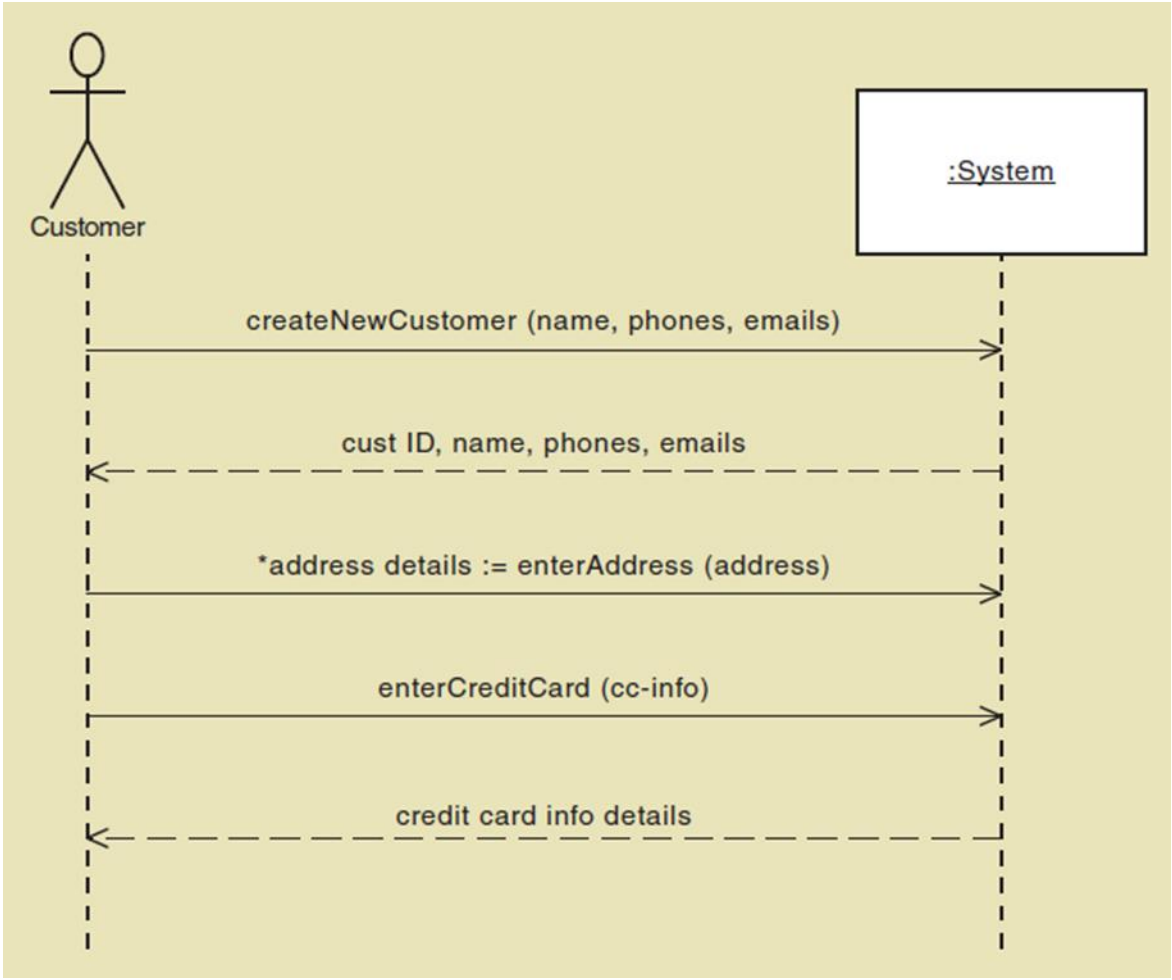
SSD for *Create customer account* Use case



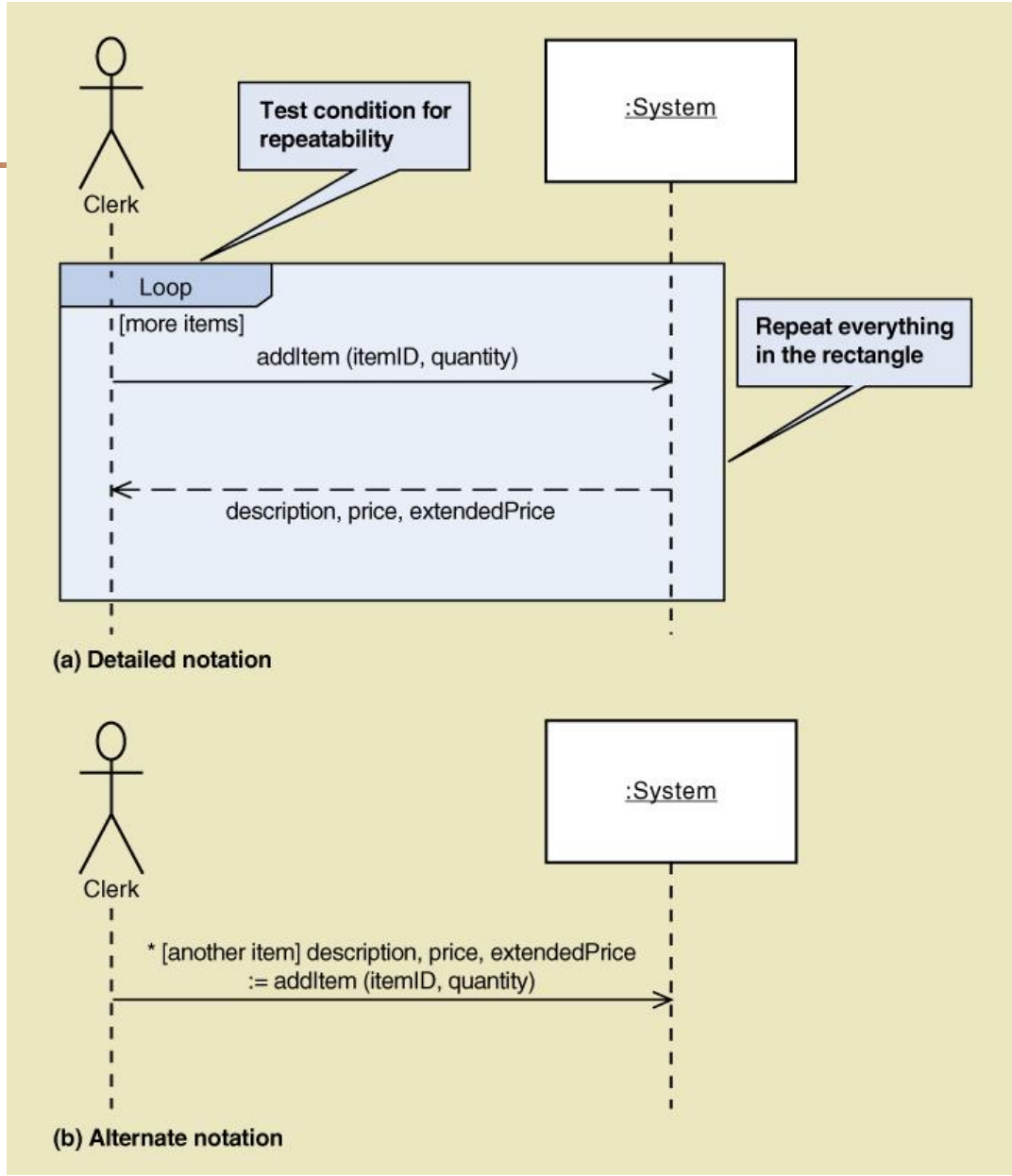
Message Notation for SSD

- *[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.


Customer
System
Request account
Create customer
Enter addresses
Create addresses
Enter credit info
Create account
Verify credit info
Return account details


Customer
:System
createNewCustomer (name, phones, emails)
cust ID, name, phones, emails
*address details := enterAddress (address)
enterCreditCard (cc-info)
credit card info details

SSD Message Examples with Loop Frame

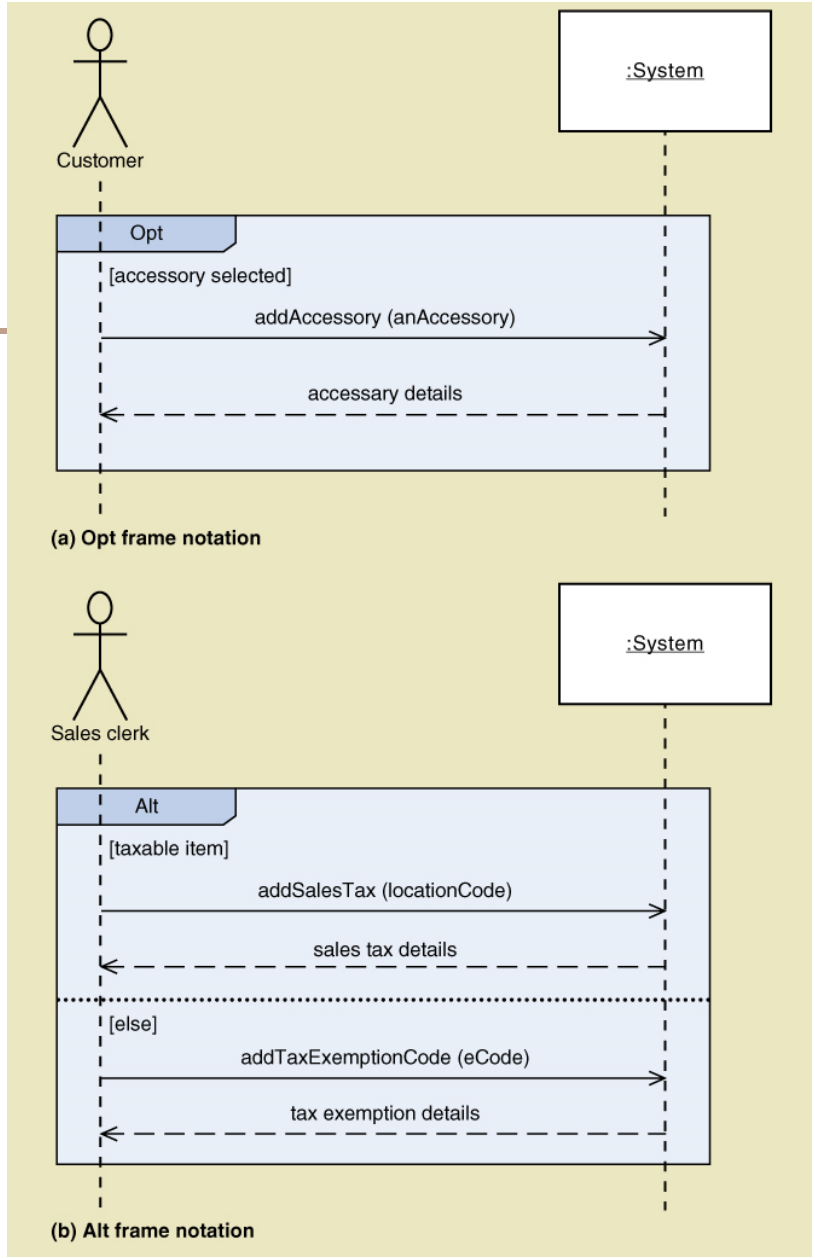


SSD Message Examples

- Opt Frame (optional)

- Alt Frame

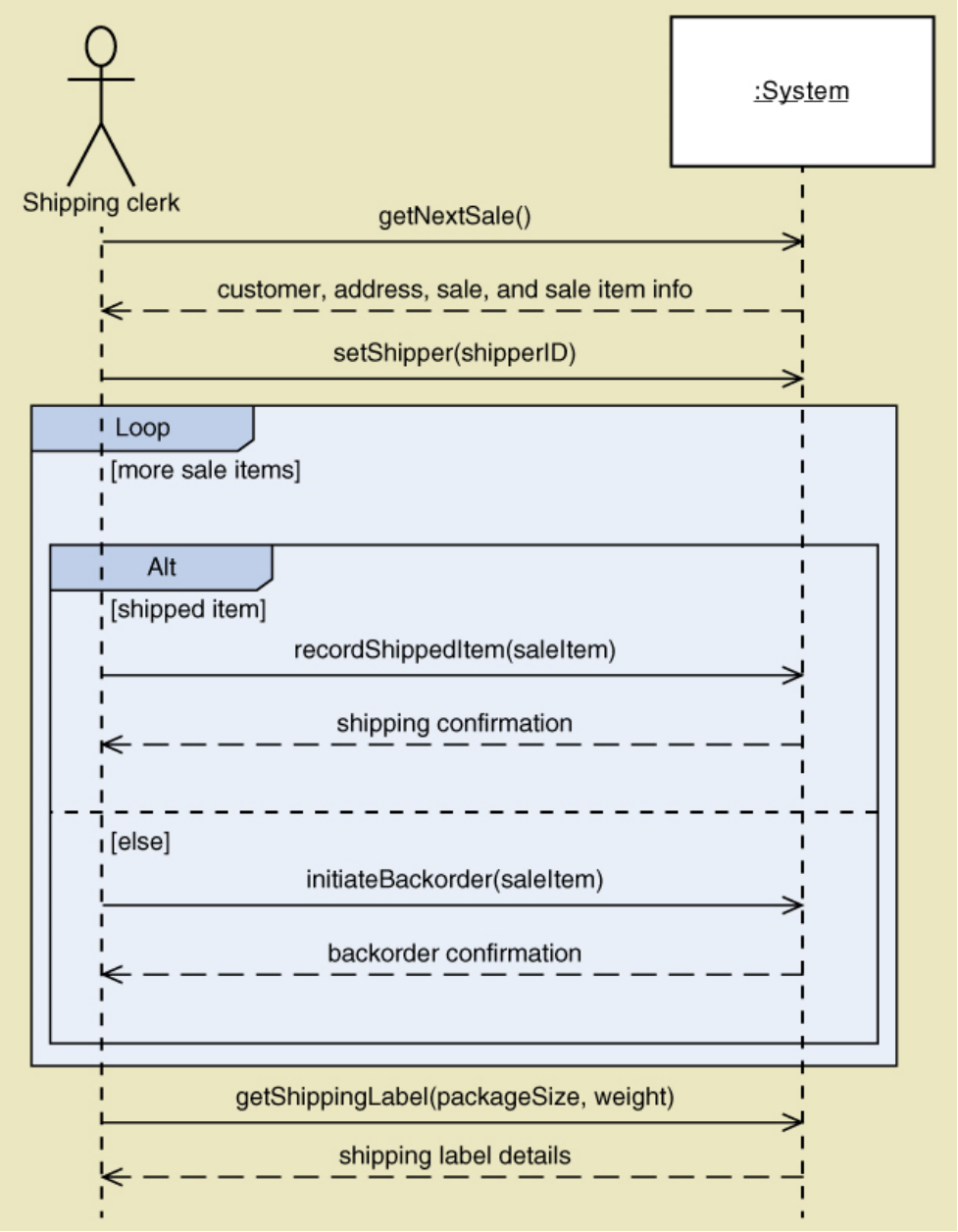
  (if-else)

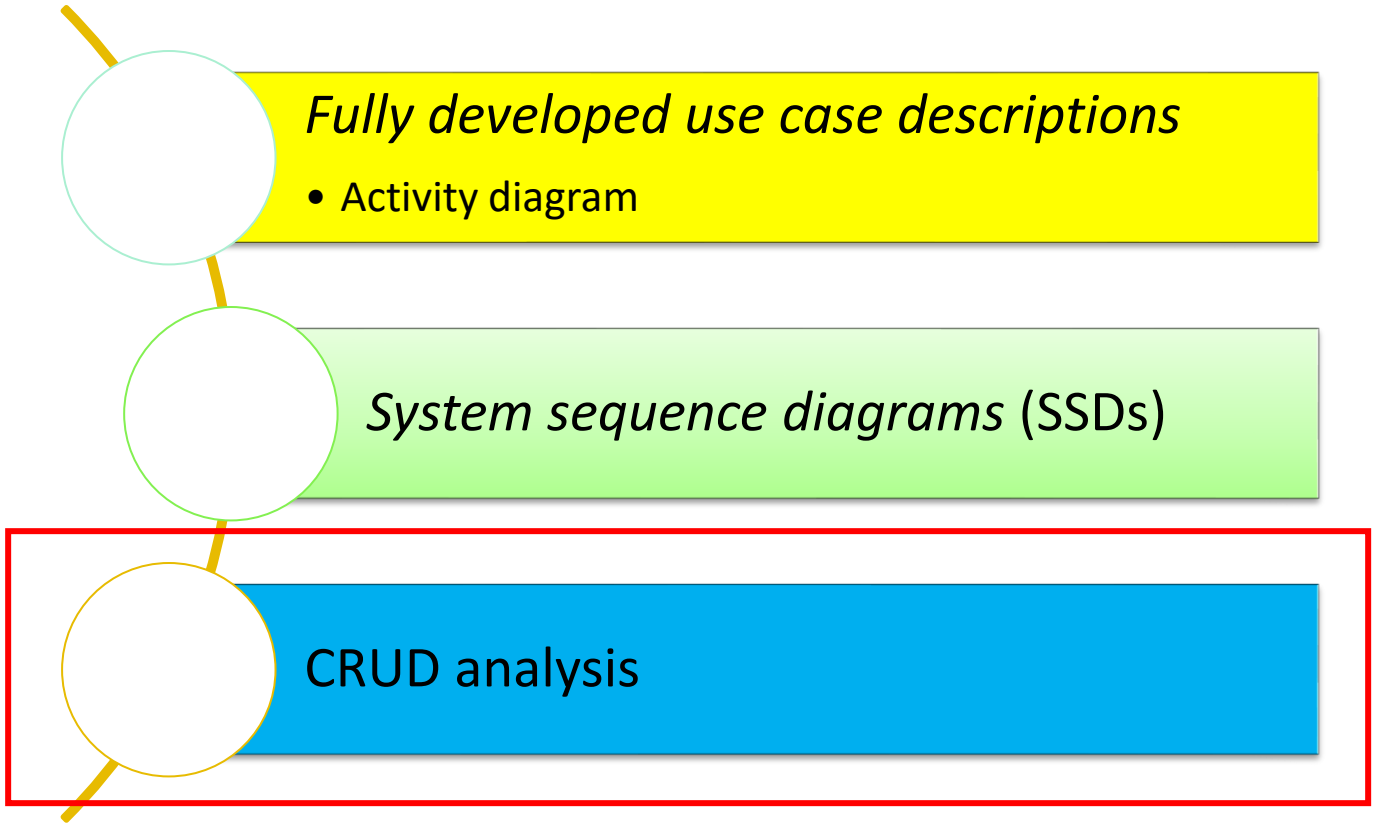


Steps for Developing SSD

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 Ship items Use Case




Fully developed use case descriptions
• Activity diagram
System sequence diagrams (SSDs)
CRUD analysis

Use Cases and CRUD

- CRUD technique –
  - Create
  - Read/Report
  - Update
  - Delete
- A good cross-check against the existing set of use cases.
- involves verifying that all of the needed use cases have been identified to maintain the data represented by the domain model class diagram.
- 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

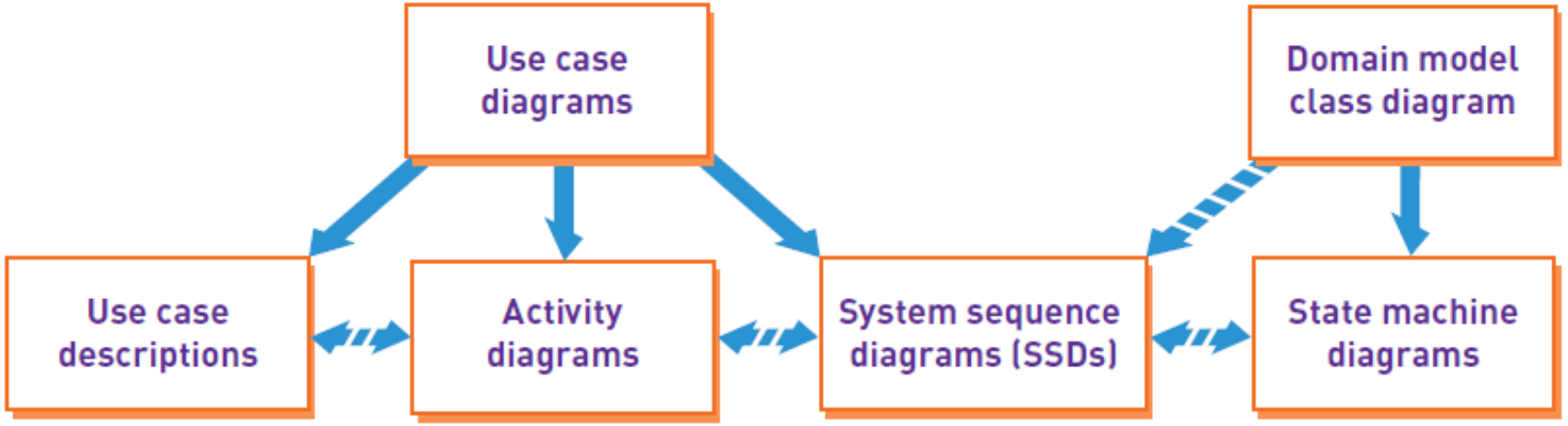
Sample CRUD Matrix

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



Summary

- 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* 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.


Ethics

A question of ethics

- This is your first week in your new job at Safety Zone, a leading producer of IT modeling software. Your prior experience with a smaller competitor gave you an edge in landing the job, and you are excited about joining a larger company in the same field.

- So far, all is going well and you are getting used to the new routine. However, you are concerned about one issue. In your initial meeting with the IT manager, she seemed very interested in the details of your prior position, and some of her questions made you a little uncomfortable. She did not actually ask you to reveal any proprietary information, but she made it clear that Safety Zone likes to know as much as possible about its competitors.

- Thinking about it some more, you try to draw a line between information that is OK to discuss, and topics such as software specifics or strategy that should be considered private.

- This is the first time you have ever been in a situation like this. ***How will you handle it?***