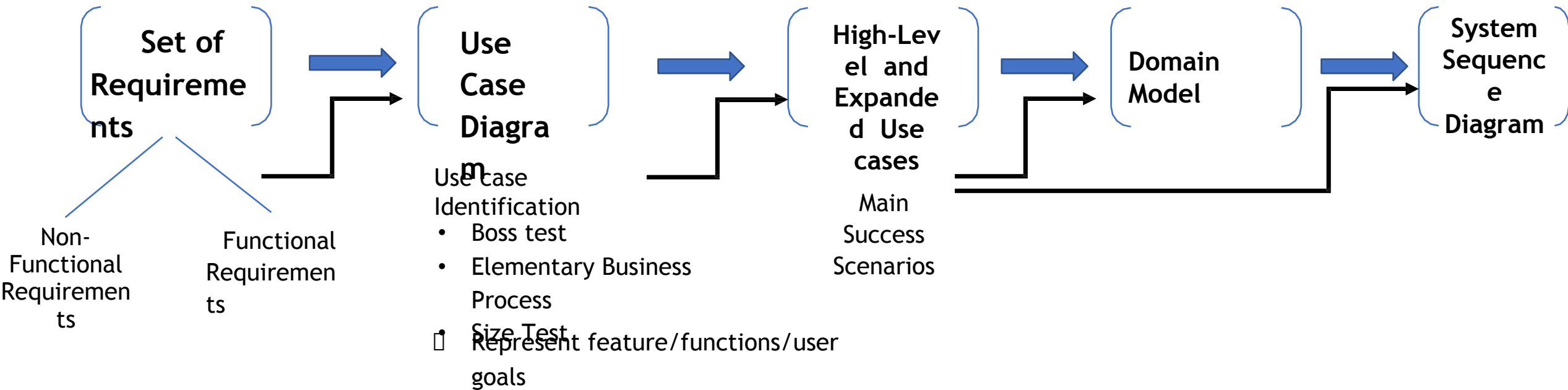


UML Modeling: System Sequence Diagram

Instructor: Mehroze Khan

Revision Up till Now



System Behavior

- System Behavior describes *what a system does*, without explaining *how it does it*.
 - Before making a logical design of how the software system should work, we need to investigate and define its behavior as “**Black box**”.
- System Sequence Diagrams (SSD)

System Sequence Diagrams

- Focuses on the interaction between the **system and external actors** (e.g., users, other systems).
- Provides a high-level view, showing **system-level interactions** without detailing internal logic or class interactions.
- Emphasizes what happens from a user's perspective, **capturing events triggered by external actors**.

System Sequence Diagrams

- System sequence diagram (SSD) are used to illustrate **interaction between actor and system** for a **particular scenario** of a **use case**.
 - A system sequence diagram is a picture that shows, **main success scenario**, the **events that actors generate**, their **order** and **inter-system events**
 - The purpose is to illustrate the **use case** in a **visual format** and *identify all system events*
- **SSD are drawn using UML notations**
 - We draw SSD using Sequence Diagram notation (UML)

Recall – Use Cases

- Use cases describe how actors interact with the software system and during this interaction
 - an actor generates events to a system
 - usually requesting some system operation to handle the event
- SSD show how certain tasks are done between actors and the system (use case scenarios).
 - The tasks could be simple, complex or repetitive tasks.

Creating System Sequence Diagrams

- Should be done for the typical course of **events/main success scenario** of the use case (and for the most **interesting alternative course**).
- A single system sequence diagram shows the interaction between an actor and the system for **one use case scenario**.
- It shows:
 - The system (as a black box)
 - The actor(s), especially the initiating actor
 - Each external system which sends messages to the system
 - The messages into and out of the system
 - The sequence in which the messages occur

System Events and System Operations

- **System operations** are the operations that the system as a black box component **offers in its public interface**.
- These are high-level operations **triggered by an event** generated by an **external actor**
- During system behavior analysis, system operations are assigned to a conceptual class **System**

Creating System Sequence Diagrams

- Should be done for the typical course of events/**main success scenario** of the use case (*and for the most interesting alternative course*).
 1. Draw a line representing the system as a black box.
 2. Identify each actor and draw a line for each such actor.
 3. From the use case *typical course of events* text, identify the system (external) events that each actor generates.
 4. Illustrate them on the diagram.
 5. Optionally, includes the use case text to the left of the diagram.

Actor Lifelines

- **User:** Represents a person interacting with the system (e.g., customer, admin, employee).
- **External System:** Represents other systems that interact with the main system (e.g., payment gateways, third-party services, databases).
- **Sensor/Device:** Represents hardware devices or sensors that interact with the system (e.g., RFID reader, IoT devices).

System Lifeline

- **System:** Represents the entire system as a single lifeline (e.g., "Banking System," "Inventory Management System").
- This lifeline does not represent individual classes or objects within the system; rather, it encapsulates the system as a whole.

Syntax of SSD

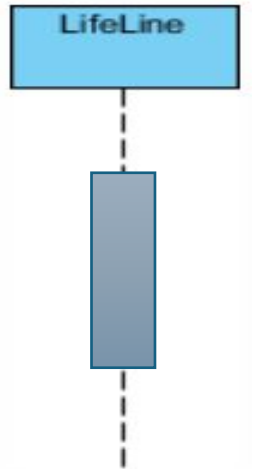
Sequence Diagram Notations

- **Lifelines:** Represents an actor or the system, depicted as a vertical dashed line starting from a rectangle (head).
- **Actor Lifeline:** Stick figure with the actor's name (e.g., Customer).
- **System Lifeline:** Box labeled with the system's name (e.g., Online Shopping System).



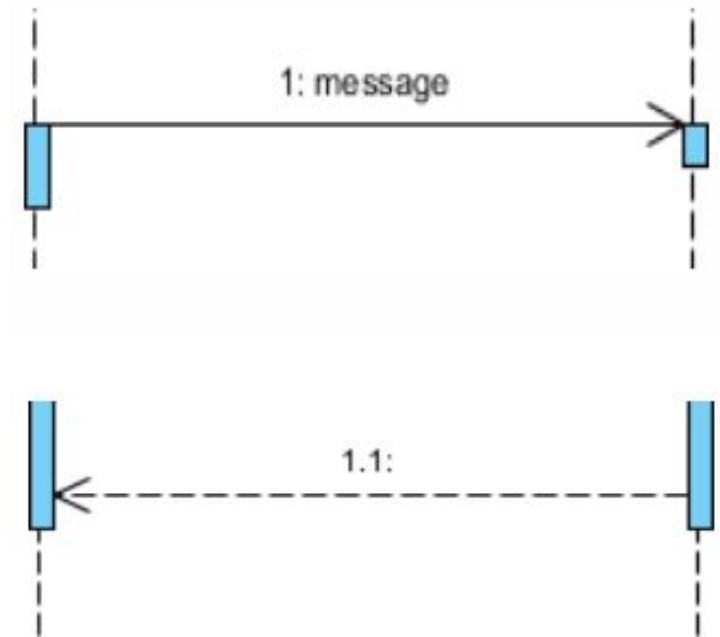
Sequence Diagram Notations




- **Activation Boxes:** Rectangles placed on the lifeline, representing the period during which an object is active and processing a message.
- Usage in SSDs: Rarely used, since SSDs focus on external interactions rather than internal processing.



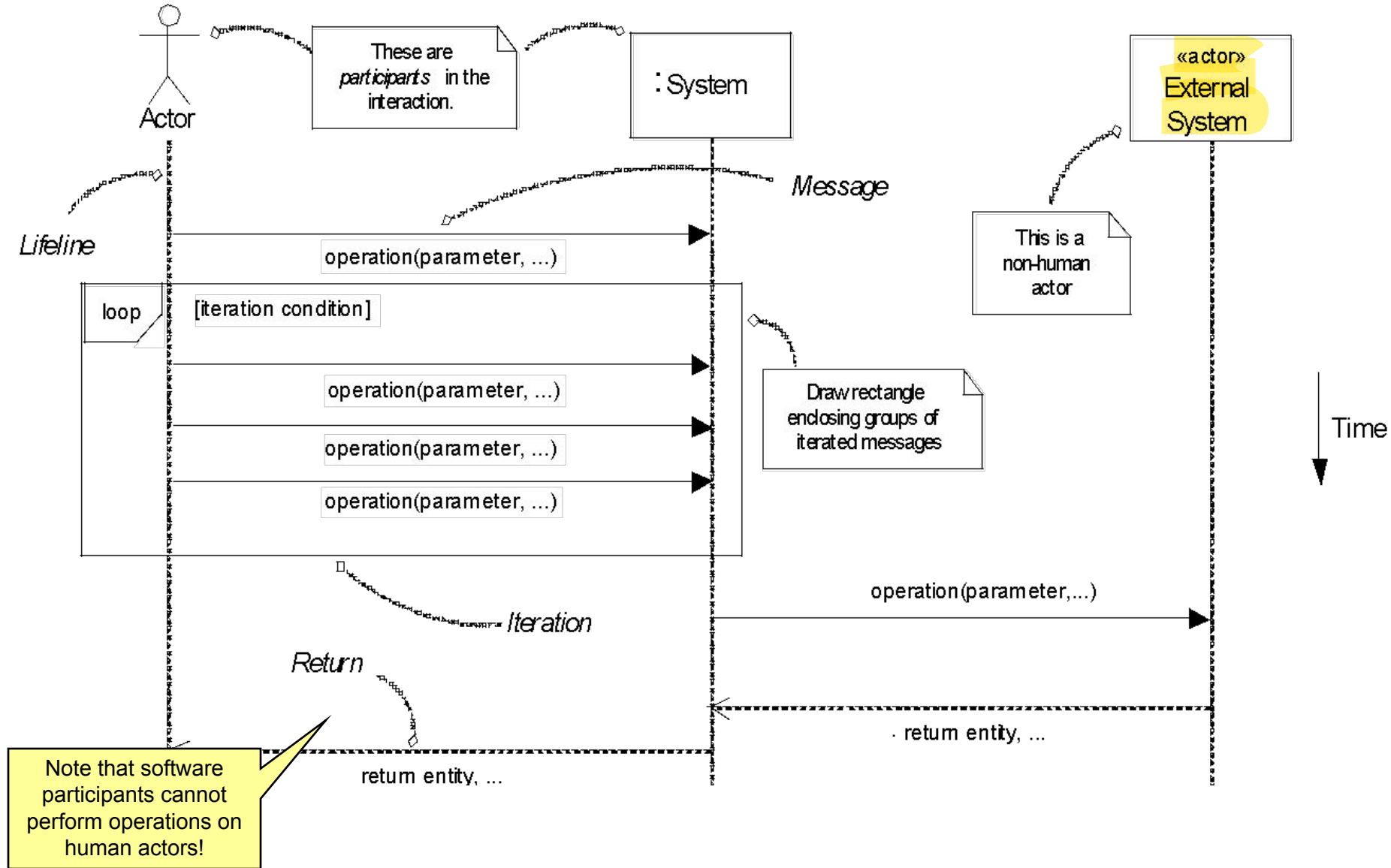
Sequence Diagram Notations

- **Call Message:** Horizontal arrows connecting lifelines, showing the flow of messages or interactions between the actor and the system.
 - Arrows: Solid lines with arrowheads pointing from the sender to the receiver.
 - Message Label: Text above or below the arrow, describing the interaction (e.g., `placeOrder()`, `login()`, `confirmPayment()`).
- **Return/Reply Message:**
 - Syntax: Dashed arrows with arrowheads, pointing back to the sender.
 - Shows responses or results from the system to the actor (e.g., `orderConfirmed`, `paymentSuccess`).



Symbol	Name	Description
 Blocking	Synchronous message symbol	Represented by a solid line with a solid arrowhead. This symbol is used when a sender must wait for a response to a message before it continues. The diagram should show both the call and the reply.
 Non-Blocking	Asynchronous message symbol	Represented by a solid line with a lined arrowhead. Asynchronous messages don't require a response before the sender continues. Only the call should be included in the diagram.
	Asynchronous return message symbol	Represented by a dashed line with a lined arrowhead.

System Sequence Diagram Notation



Process Sale

1. Customer arrives at POS checkout with goods and/or services to purchase.
2. Cashier starts a new sale.
3. Cashier enters item identifier.
4. System records sale line item and presents item description, price, and running total. Price calculated from a set of price rules.

Cashier repeats steps 3-4 until indicates done.

5. System presents total with taxes calculated.
6. Cashier tells Customer the total, and asks for payment.
7. Customer pays and System handles payment.
8. System logs completed sale and sends sale and payment information to the external Accounting system (for accounting and commissions) and Inventory system (to update inventory).
9. System presents receipt.
10. Customer leaves with receipt and goods (if any).

Recall the *Process Sale* Use case

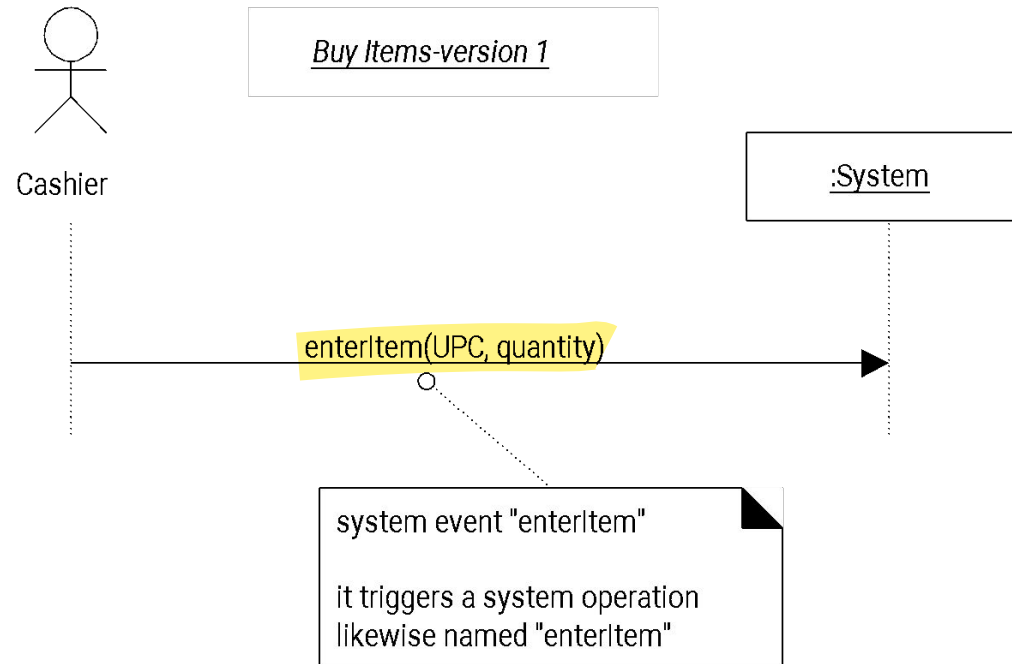
Actor Action	System Response
1. The use case begins when Customer arrives at the POS checkout with items to purchase. Cashier Starts a new Sale.	
2. Cashier records each item identifier . If there is more than one item, Cashier can enter the quantity as well.	3. Determines the Line item price and adds the Line item information to the running sales transaction. The description and price of the the current item are presented.
4. Cashier repeats 3 – 4 for all items. On completion of item entry, the Cashier indicates to the POS that item entry is complete and sale has ended .	5. Calculates and presents the sale total with taxes .
6. The Cashier tells the Customer the total and asks for payment.	
7. Customer makes payment by cash	8. System logs completed sale 9. Provides receipt

How to construct an SSD from a use case?

1. Draw System as black box on right side
2. For each actor that directly operates on the System, draw a stick figure and a lifeline.
3. For each System events that each actor generates in use case, draw a message.

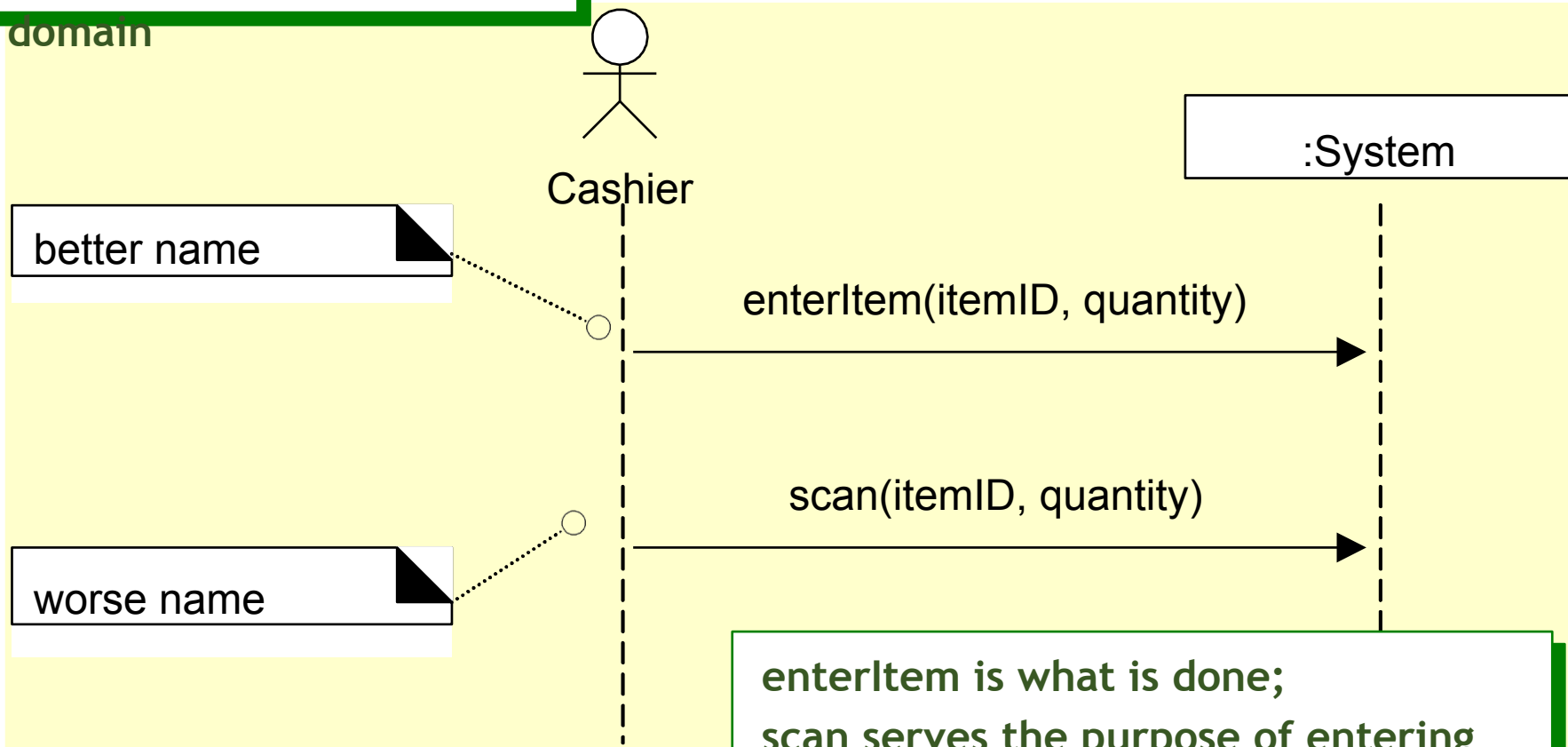
System Events

- **System event** is an external input event generated by an actor (may include parameters).



System Sequence Diagrams

Naming: wording should be consistent with the domain

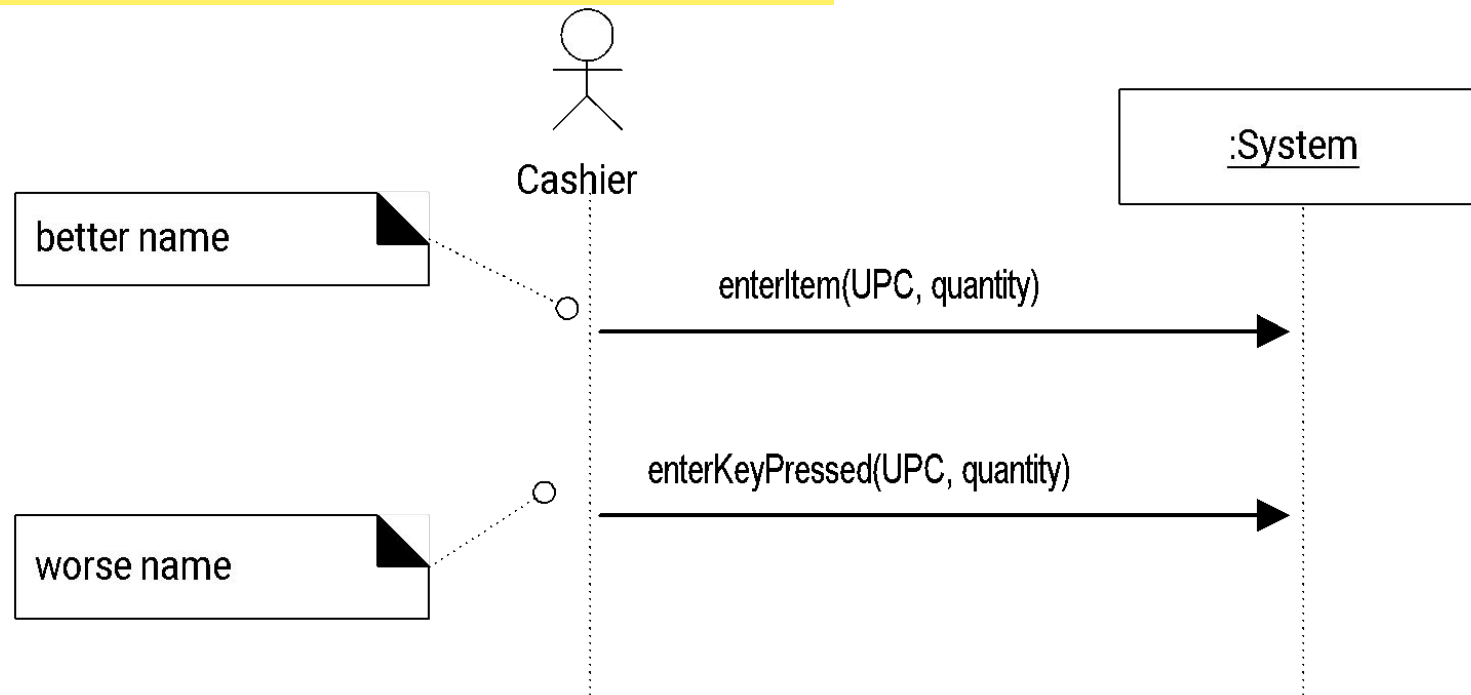


enterItem is what is done;
scan serves the purpose of entering
an item

and is system related (bad)

Naming System events & operations

- System events and associated system operations should be expressed at the level of intent
 - Use verbs from use case or make, start, enter, end etc.. rather than physical input medium or UI widget

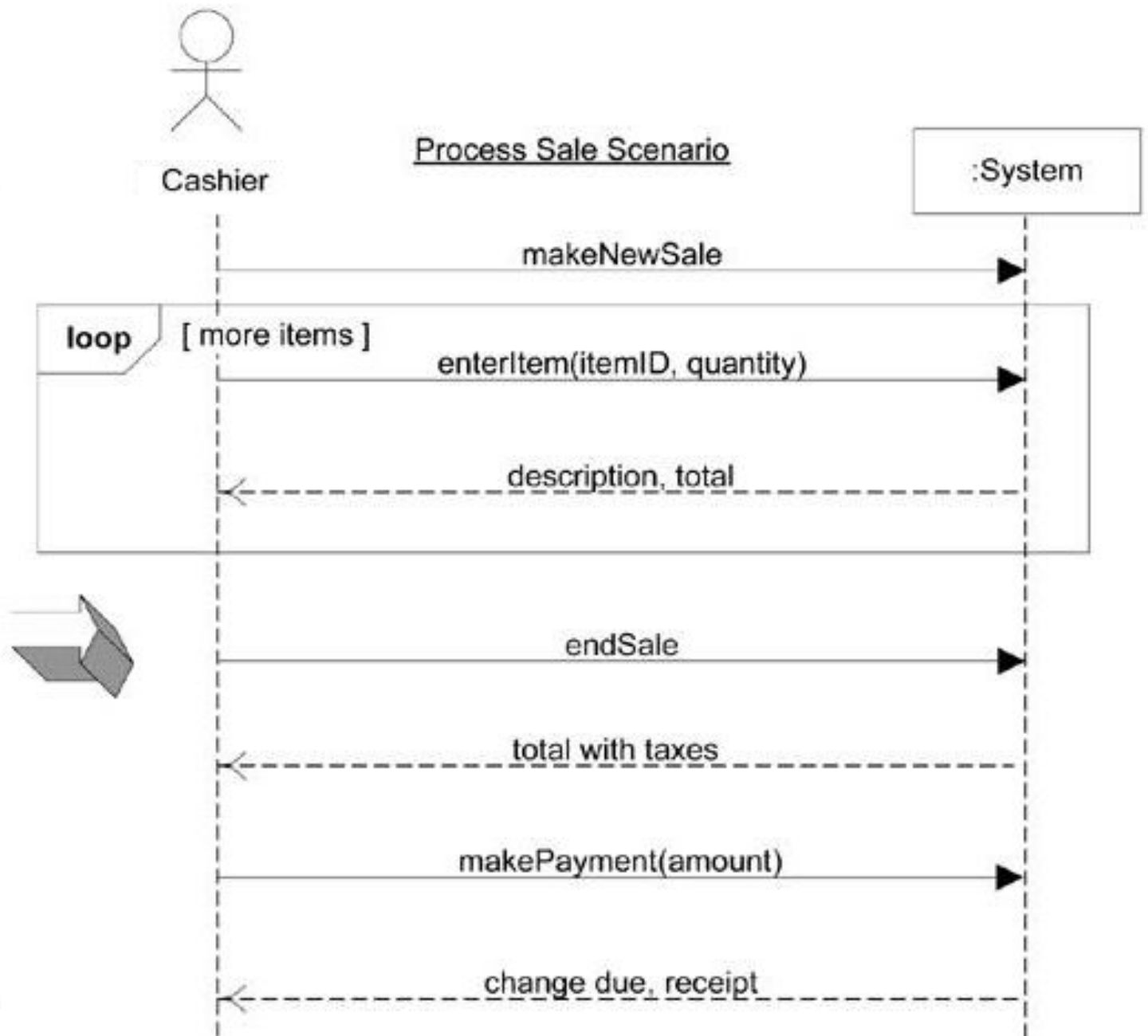


Recall the *Process Sale* Use case

Actor Action	System Response
1. The use case begins when Customer arrives at the POS checkout with items to purchase. Cashier Starts a new Sale.	
2. Cashier records each item identifier . If there is more than one item, Cashier can enter the quantity as well.	3. Determines the Line item price and adds the Line item information to the running sales transaction. The description and price of the the current item are presented.
4. Cashier repeats 3 – 4 for all items. On completion of item entry, the Cashier indicates to the POS that item entry is complete and sale has ended .	5. Calculates and presents the sale total with taxes .
6. The Cashier tells the Customer the total and asks for payment.	
7. Customer makes payment by cash	8. System logs completed sale 9. Provides receipt

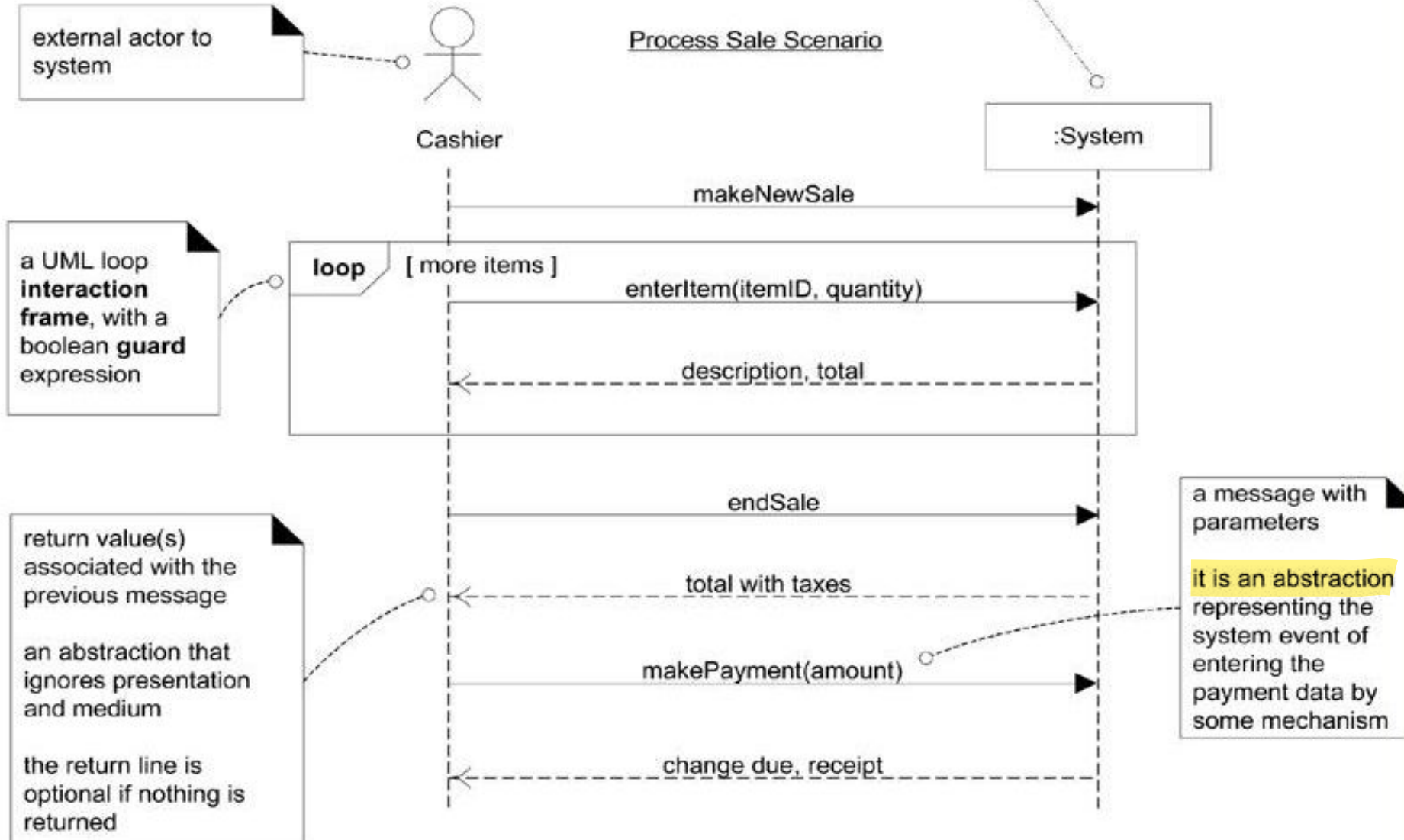
Simple cash-only Process Sale scenario:

1. Customer arrives at a POS checkout with goods and/or services to purchase.
 2. Cashier starts a new sale.
 3. Cashier enters item identifier.
 4. System records sale line item and presents item description, price, and running total.
- Cashier repeats steps 3-4 until indicates done.
5. System presents total with taxes calculated.
 6. Cashier tells Customer the total, and asks for payment.
 7. Customer pays and System handles payment.
- ...

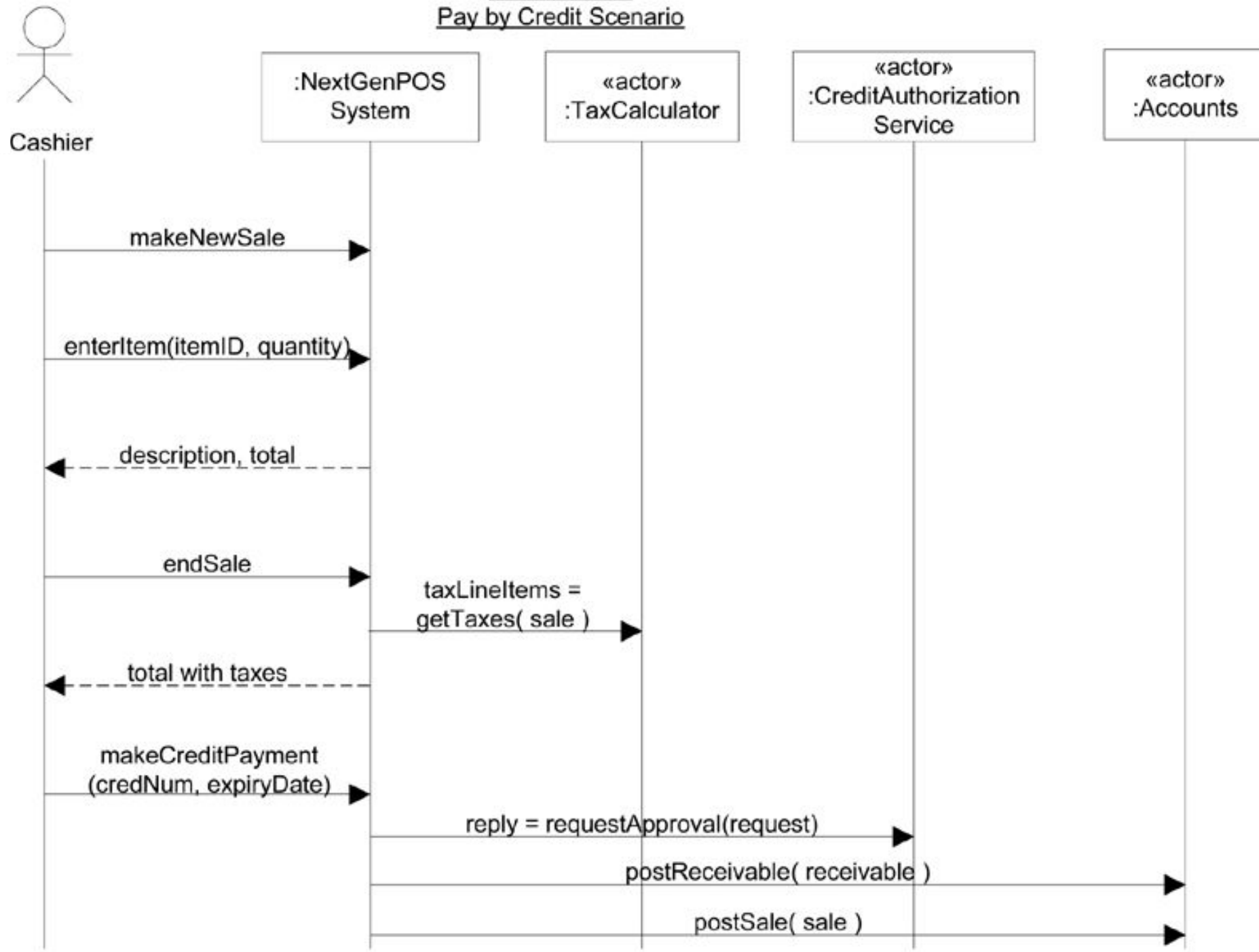


the name could be "NextGenPOS" but "System" keeps it simple

the ":" and underline imply an instance, and are explained in a later chapter on sequence diagram notation in the UML



Process Sale
Pay by Credit Scenario



Search Book : Use Case

- Main scenario -

The Customer specifies an author on the Search Page and then presses the Search button.

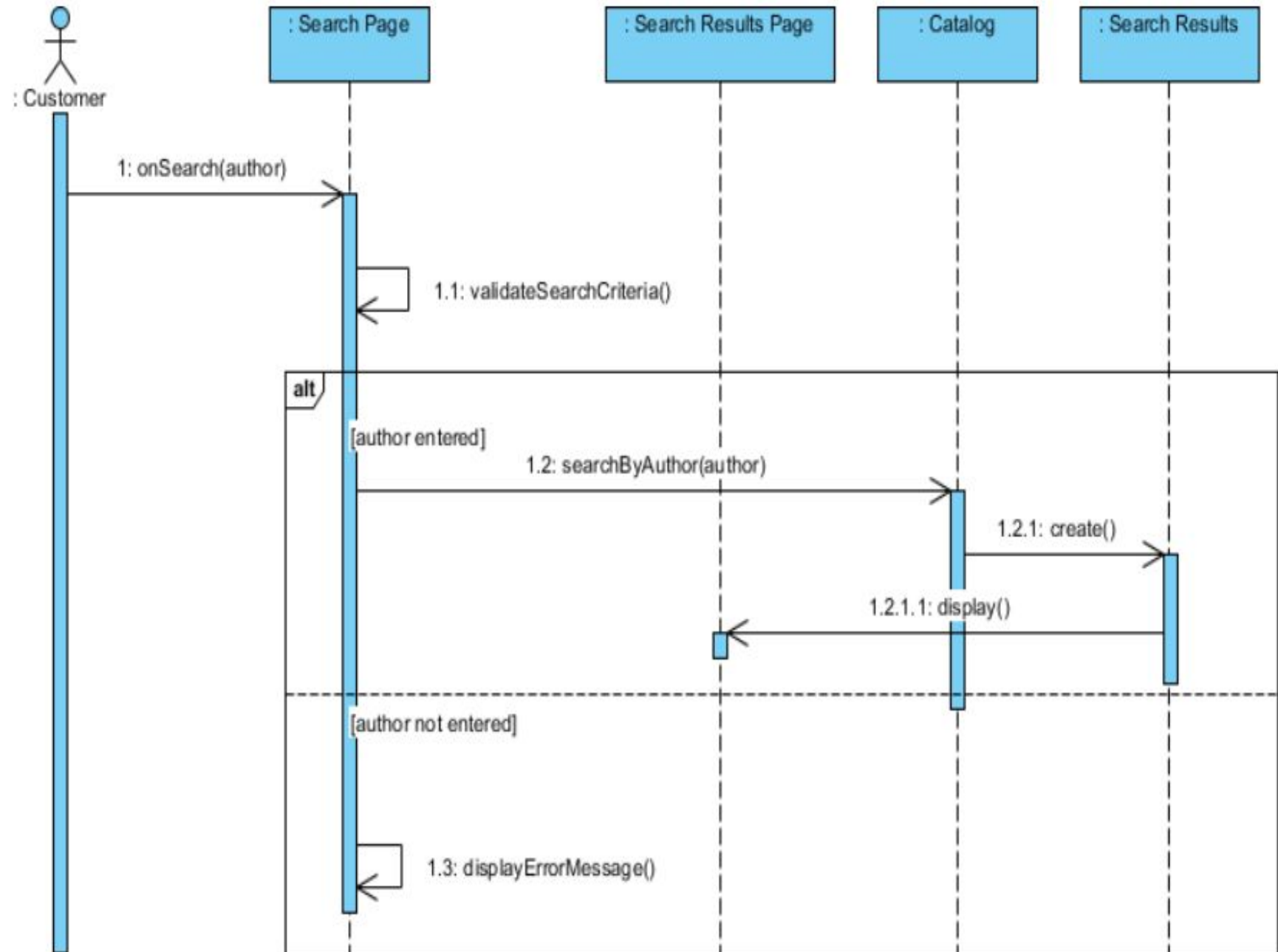
The system validates the Customer's search criteria.

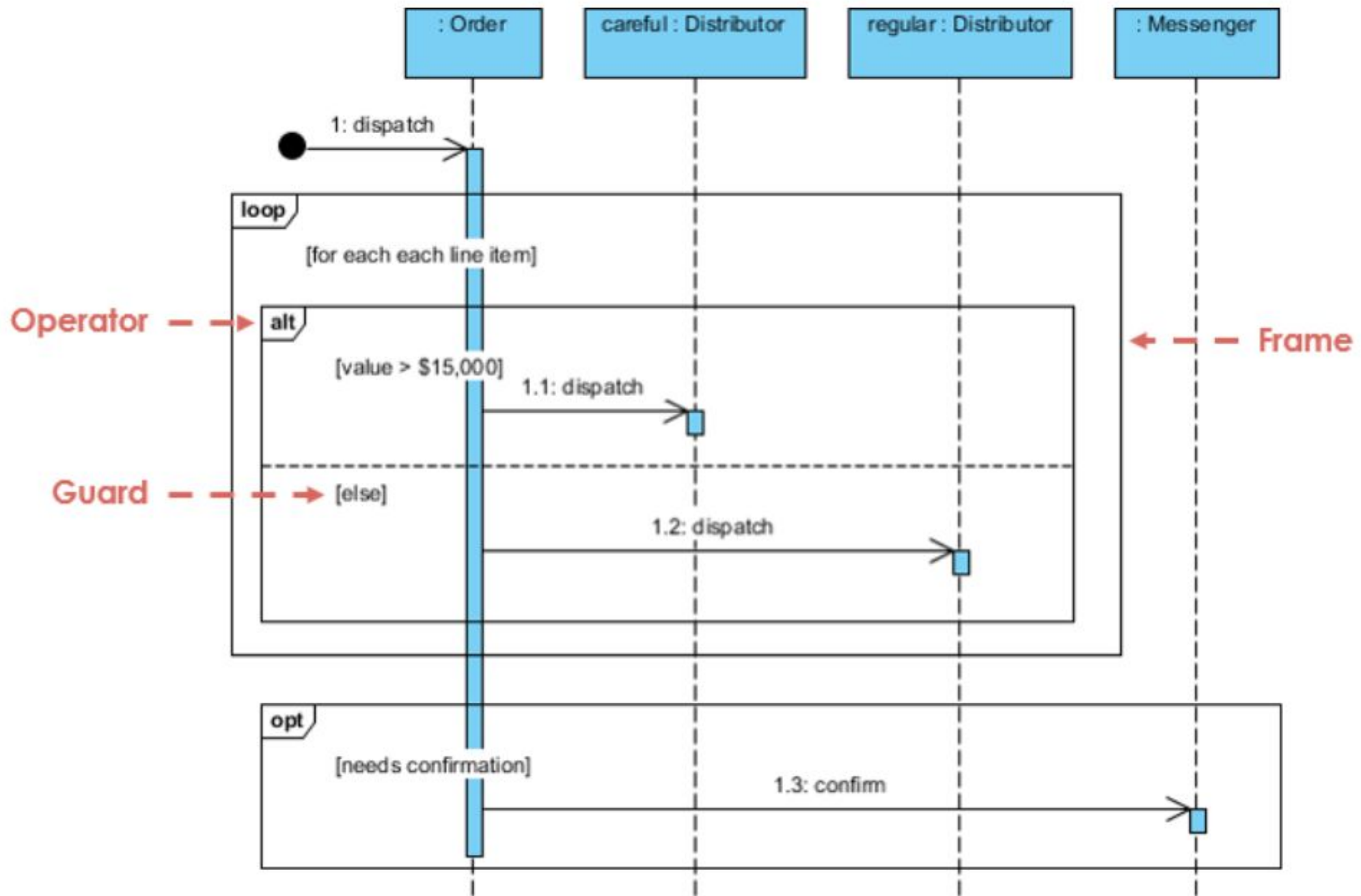
If author is entered, the System searches the Catalog for books associated with the specified author.

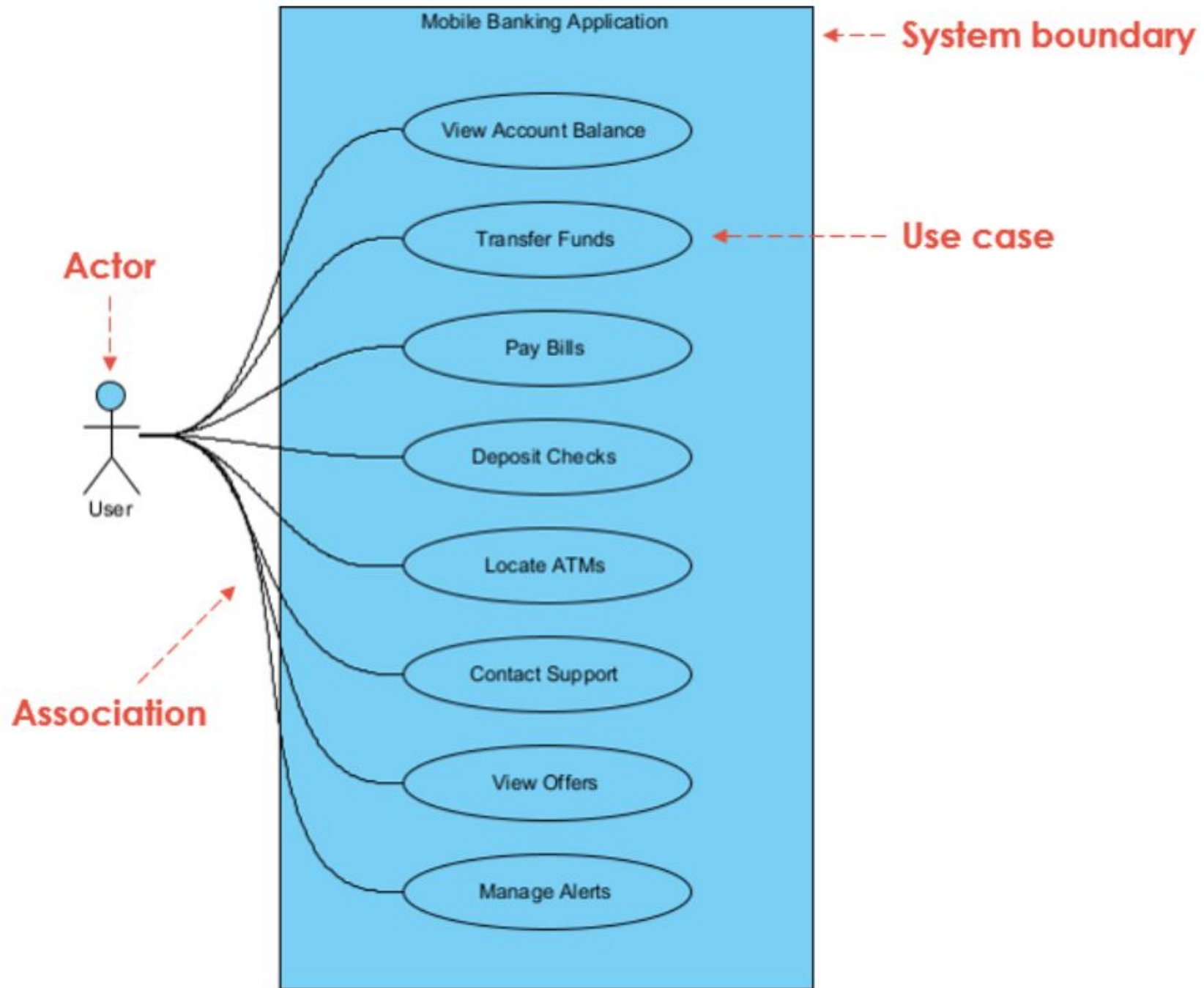
When the search is complete, the system displays the search results on the Search Results page.

- Alternate path -

If the Customer did not enter the name of an author before pressing the Search button, the System displays an error message







Use case description for the “Transfer Funds”

Use Case Name: Transfer Funds

Actors: User

Summary: This use case allows the user to transfer funds between their own accounts or to other accounts.

Preconditions:

- The user must be logged in to the mobile banking application.
- The user must have at least one account set up in the application.

Basic Flow of Events:

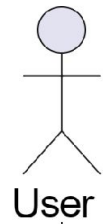
1. The user selects the “Transfer Funds” option from the main menu of the mobile banking application.
2. The application presents the user with a list of their accounts.
3. The user selects the account they want to transfer funds from.
4. The application presents the user with a form to fill out with the recipient’s account information, including the account number and the name of the recipient.
5. The user provides the account information and enters the amount they want to transfer.
6. The application validates the recipient’s account information and the available balance in the user’s account from the bank.

7. If the validation is successful:

1. The application deducts the transfer amount from the user’s account and adds it to the recipient’s account.
2. The application displays a confirmation message to the user with the details of the transfer.

8. If validation fails:

1. The application displays an error message to the user specifying the reason for the failure (e.g., insufficient balance, invalid recipient account information).



User

:Mobile Banking Application

:Banking System

1 Select "Transfer Funds"

2 Display list of accounts

3 Select account to transfer from

4 Display transfer form

5 Enter recipient details and amount

6 Validate recipient's account info and available balance

alt

[Validation successful]

6.1 Deduct amount from user's account

6.2 Transfer successful

6.3 Display confirmation message

[Validation fails]

6.1 Transfer failed (e.g. insufficient balance)

6.2 Display error message

References

- Larman, Craig. Applying UML and patterns: an introduction to object oriented analysis and design and interactive development. Pearson Education India, 2012.
- Object-Oriented Analysis and Design with Applications, Grady Booch et al., 3rd Edition, Pearson, 2007.
- Timothy C. Lethbridge, Robert Laganaiere, Object-Oriented Software Engineering (2nd Edition), McGraw Hill, 2005
- Object-Oriented Modeling and Design with UML, Michael R. Blaha and James R. Rumbaugh, 2nd Edition, Pearson, 2005.