Software Engineering

جامعة أبوظبي
Abu Dhabi University

Chapter 5-E – System Modeling

{Behavioral Modeling/ Sequence Diagrams}

Topics covered

- ✧ Understand the relationship between the behavioral models and the structural and functional models.
- ✧ Understand the rules and style guidelines for sequence diagrams.
- ✧ Understand the processes used to build sequence diagrams.
- ✧ Be able to build sequence diagrams.

Introduction

- Behavioral models describe the internal behavior of a system
- Behavioral model types:
  - Representations of the details of a business process identified by use-cases
    - Interaction diagrams (Sequence & Communication)
    - Shows how objects collaborate to provide the functionality defined in the use cases.
  - Representations of changes in the data
    - Behavioral state machines
- Focus (for now) is on the dynamic view of the system, not on how it is implemented
- One of the primary purposes of behavioral models is to show how the underlying objects in a problem domain will work together to form a collaboration to support each of the use cases.

Behavioral Models

- ✧ Software Engineers view the problem as a set of use cases supported by a set of collaborating objects
- ✧ Behavioral models depict this view of the business processes:
  - How the objects interact and form a collaboration to support the use cases
  - An internal view of the business process described by a use case
- ✧ Building behavioral models is an iterative process which may induce changes in other models
- ✧ In this chapter, we discuss how software engineers use behavioral models to represent the internal behavior or dynamic view of a software system.
- ✧ There are behavioral models used to represent the underlying details of a business process portrayed by a use-case model. In UML, interaction diagrams (sequence and communication) are used for this type of behavioral model.

Interaction Diagrams

- Objects—an instantiation of a class
  - Patient is a class
  - Rashed Saeed is an instantiation of the patient class (object)
- Attributes—characteristics of a class
  - Patient class: name, address, phone, etc.
- Operations—the behaviors of a class, or an action that an object can perform
- Messages—information sent to objects to tell them to execute one of their behaviors
  - A function call from one object to another
- Types
  - **Sequence Diagrams—emphasize message sequence**
  - Communication Diagrams—emphasize message flow

Introduction – System Sequence Diagram

✧A **system sequence diagram** is a fast and easily created artifact that illustrates input and output events related to the systems under discussion

✧Before proceeding to a logical design of how a software application will work, we should investigate and define the system behavior as a "**black box**“.

System Event and Response

- Use cases describe how external actors interact with the software system.

- During this interaction an actor generates events to a system, usually requesting some operation in response.

For example, when a cashier enters an item's ID, the cashier is requesting the POS system to record that item's sale. That request event initiates an operation upon the system.

System sequence diagram

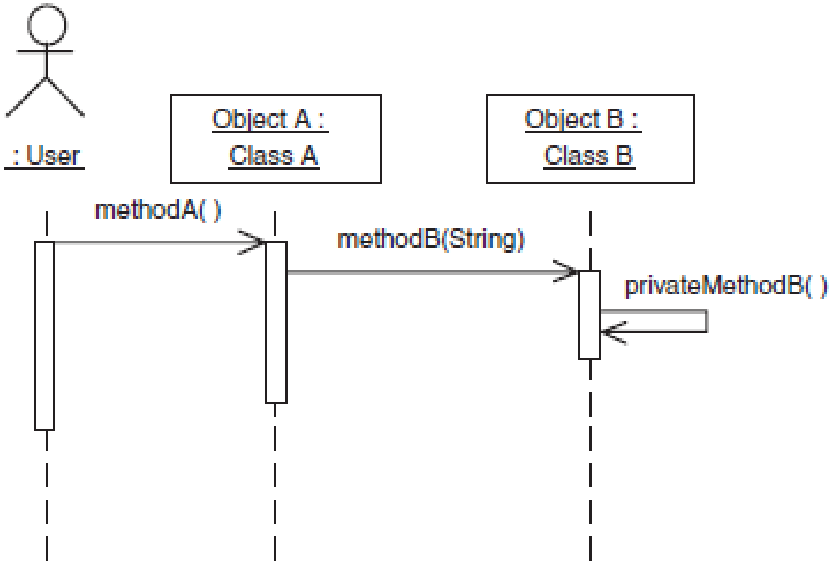
- A system sequence diagram (SSD) is a picture that shows, for a particular scenario of a use case, the events that external actors generate, their order, and inter-system events.
  - The emphasis of the diagram is events that cross the system boundary from actors to systems.
  - An SSD should be done for the main success scenario of the use case, and frequent or complex alternative scenarios.
  - An SSD is generated from inspection of a use case

Suggestion:

- One SSD – one Use Case

Introduction to Sequence Diagrams

- The Figure below shows a simple sequence diagram. In the example, an object of type *USER* triggers the occurrence of some event by calling the method *METHODA* in the object *OBJECTA* of type *CLASSA*. *METHODA* then calls the method *METHODB* in *OBJECTB* of type *CLASSB*. *METHODB* then calls method *PRIVATEMETHODB* within the object that contains *METHODB*.
- The arrowhead points to the method that was called. Information can flow both ways. If the method is fully specified, as is the case for METHODB of OBJECTB, the information about the call is included in the method call. The return value is not shown in the sequence diagram.



Identifying Inputs and Outputs — the System Sequence Diagram

✧ System sequence diagram (SSD)

- Describes flow of information
- Identifies interaction between actors and system
- Message oriented

SSD Notation

- Actor “interacts” with the system via input/output
- SSDs use object notation
  - Box (rectangle) refers to individual object
  - Name of the object underlined
- Lifeline
  - is a vertical line under object or actor to show passage of time for object
  - Indicates sequence of the messages sent/received
  - If vertical line dashed
    - Creation and destruction of thing is not important for scenario
  - Long narrow rectangles
    - Activation lifelines emphasize that object is active only during part of scenario
- Message is labeled on arrows to show messages sent to or received by actor or system

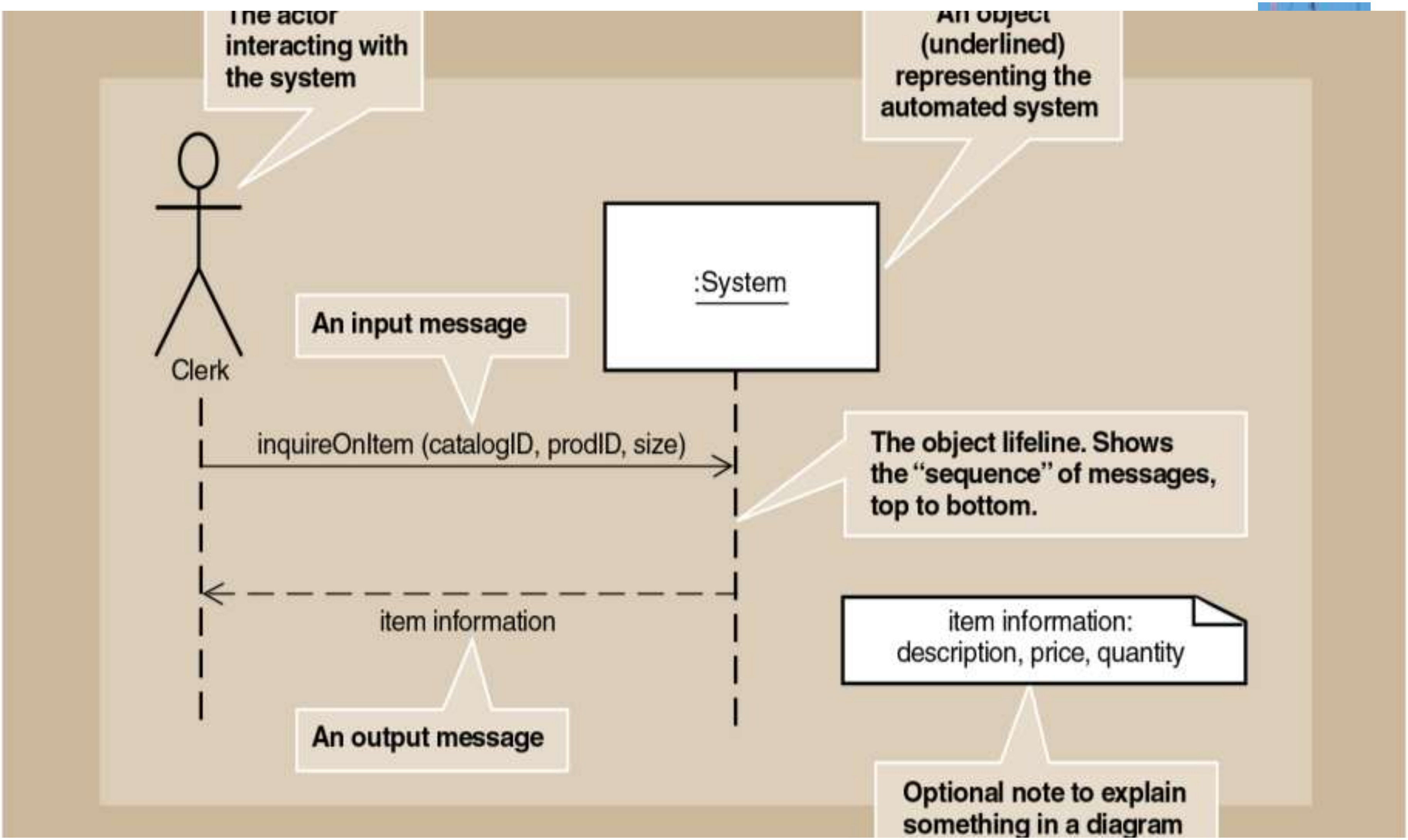
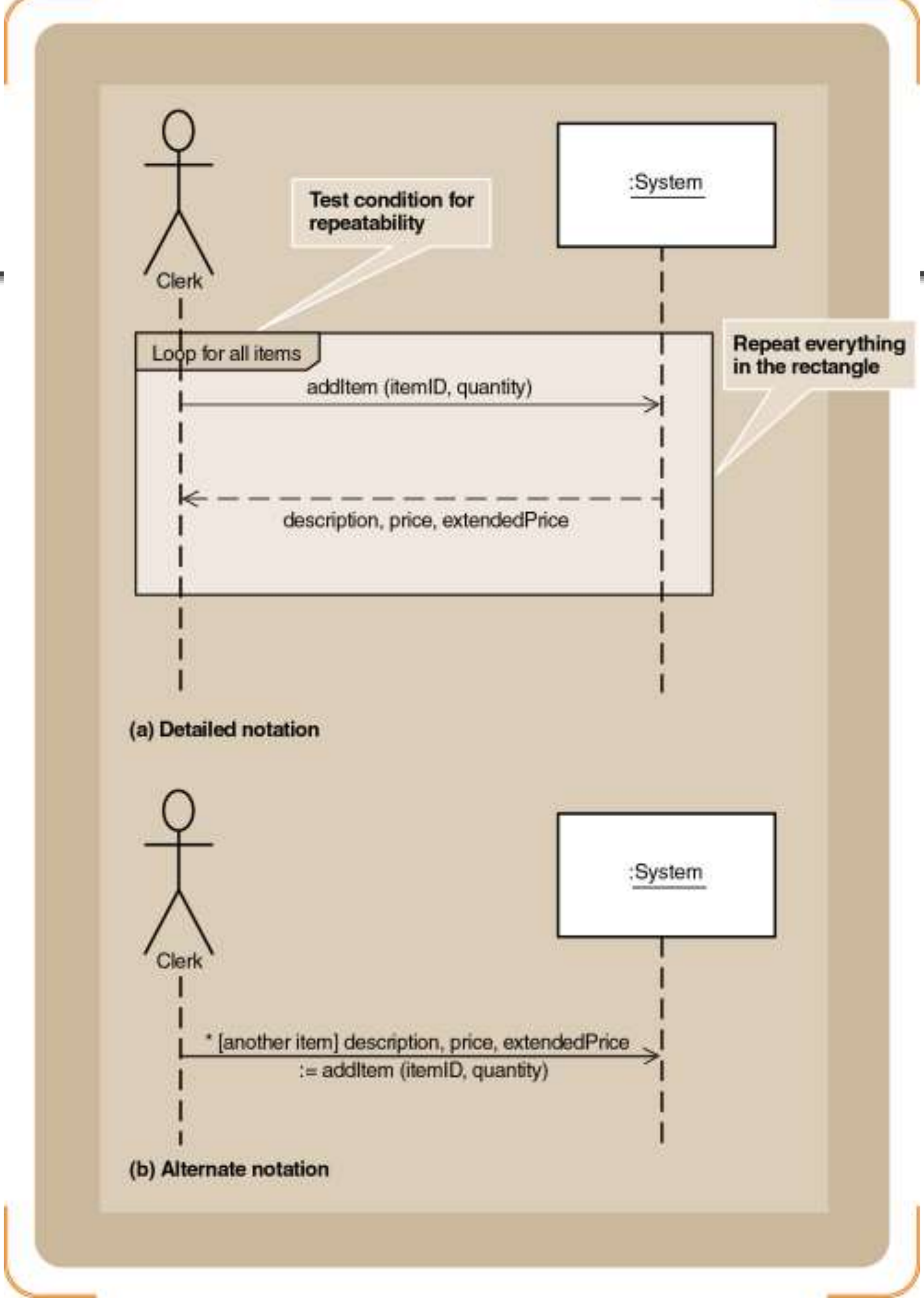

The actor interacting with the system
An object (underlined) representing the automated system
:System
An input message
Clerk
inquireOnItem (catalogID, prodID, size)
The object lifeline. Shows the "sequence" of messages, top to bottom.
item information
item information: description, price, quantity
An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

Figure. Sample System Sequence Diagram

SSD Notation (continued)

- Message syntax can take several forms
  - Depends on send/return direction
- Message semantics: actions (like commands) invoked on destination object
- Complete message notation:*[true/false condition] return-value := message-name (parameter-list)

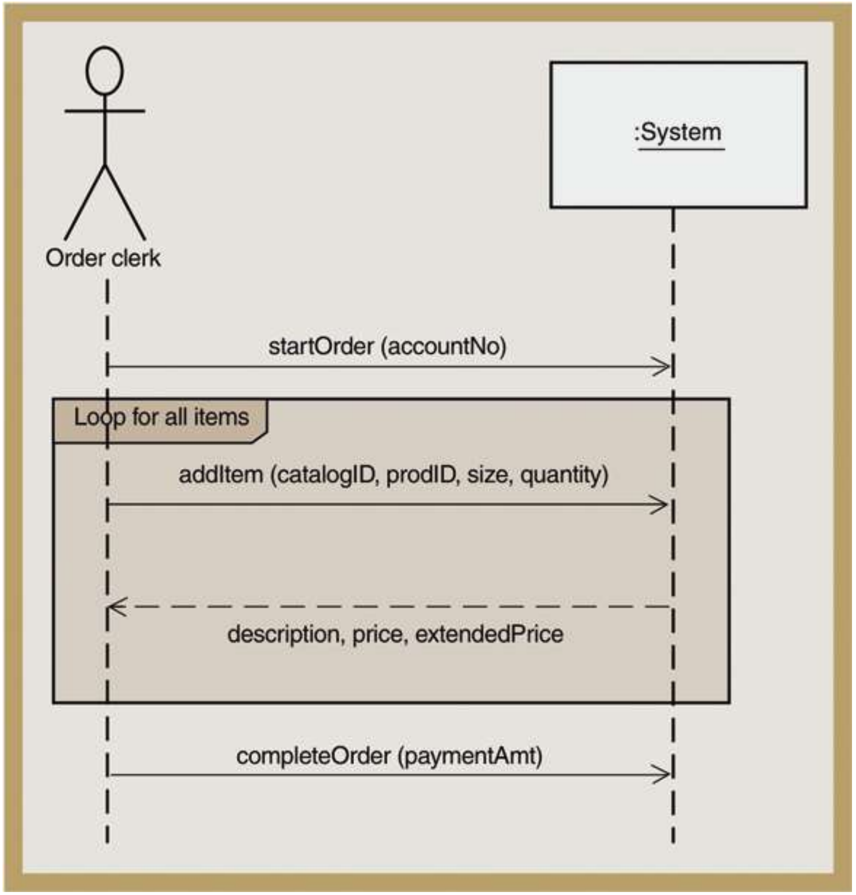
Repeating Message (A)
Detailed Notation (B)
Alternate Notation


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

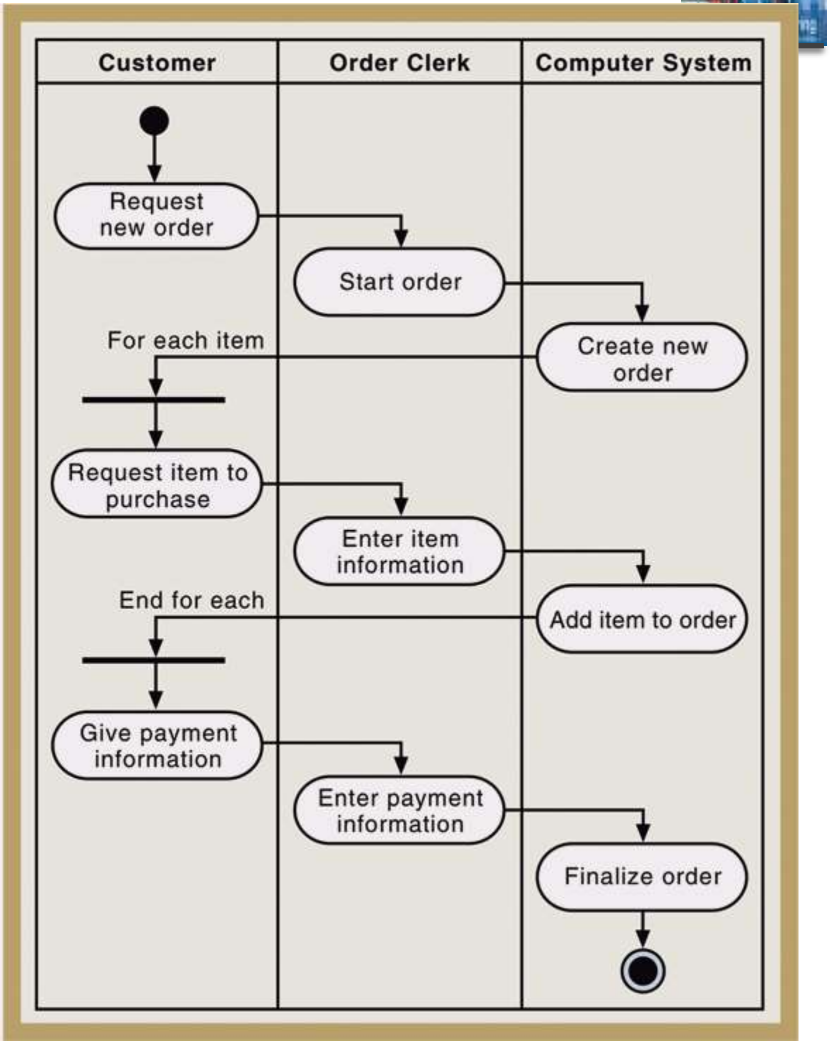
Developing a System Sequence Diagram

- Begin with detailed description of use case
  - Fully developed form
  - Activity diagrams (out of our subject)
- (4) step process for turning activity diagram or UC Description into SSD
  - [1] Identify the input messages
  - [2] Describe messages from external actor to system
  - [3] Identify/apply special conditions to input messages
  - [4] Identify and add the output return messages

Activity Diagram and Resulting SSD for Telephone Order Scenario



Order clerk
:System
startOrder (accountNo)
Loop for all items
addItem (catalogID, prodID, size, quantity)
description, price, extendedPrice
completeOrder (paymentAmt)



Case Study: How to do SSD in the POS Project

✧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.

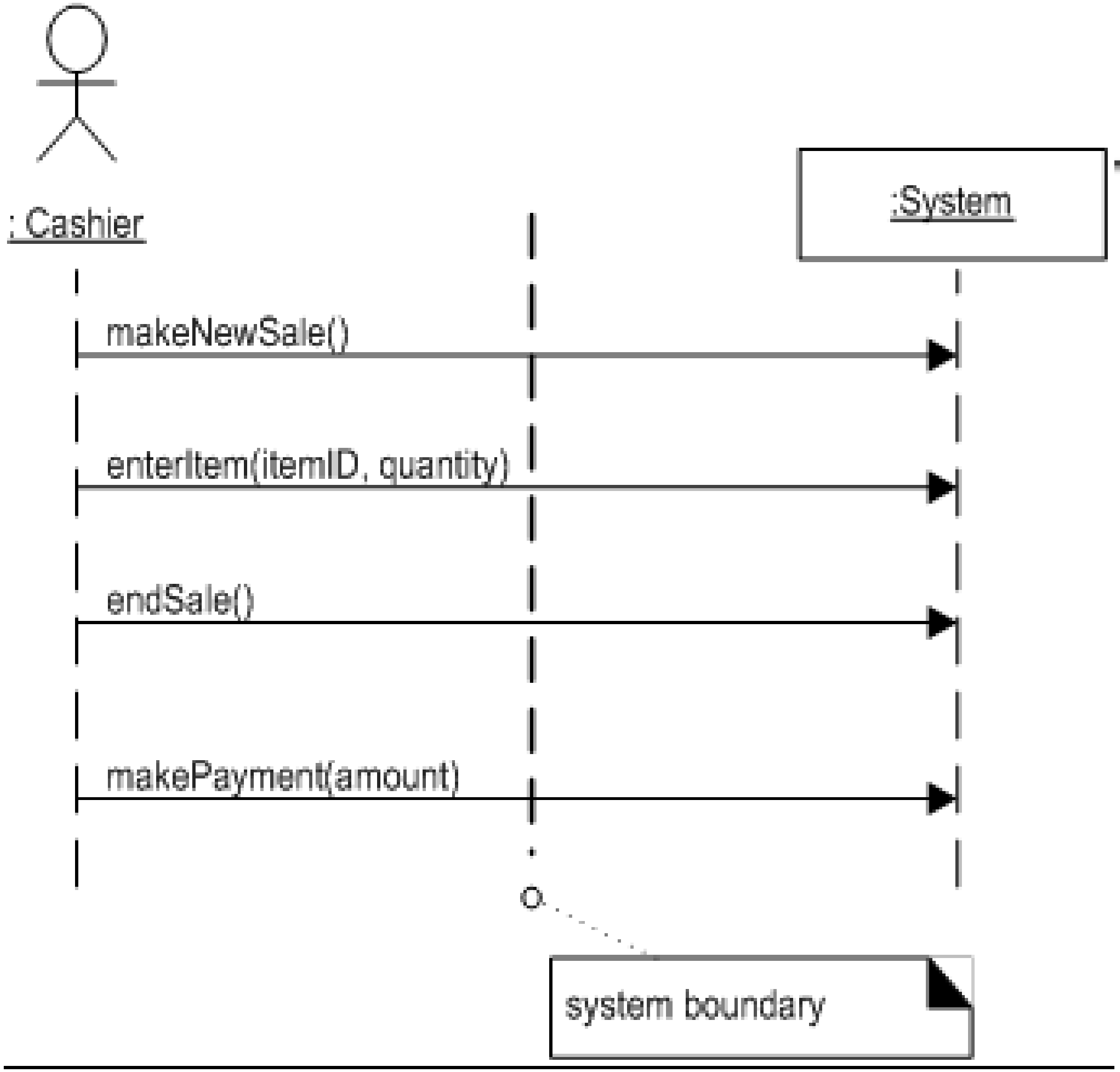
5. System presents total with taxes calculated.

6. Cashier tells Customer the total, and asks for payment.

7. Customer pays and System handles payment.

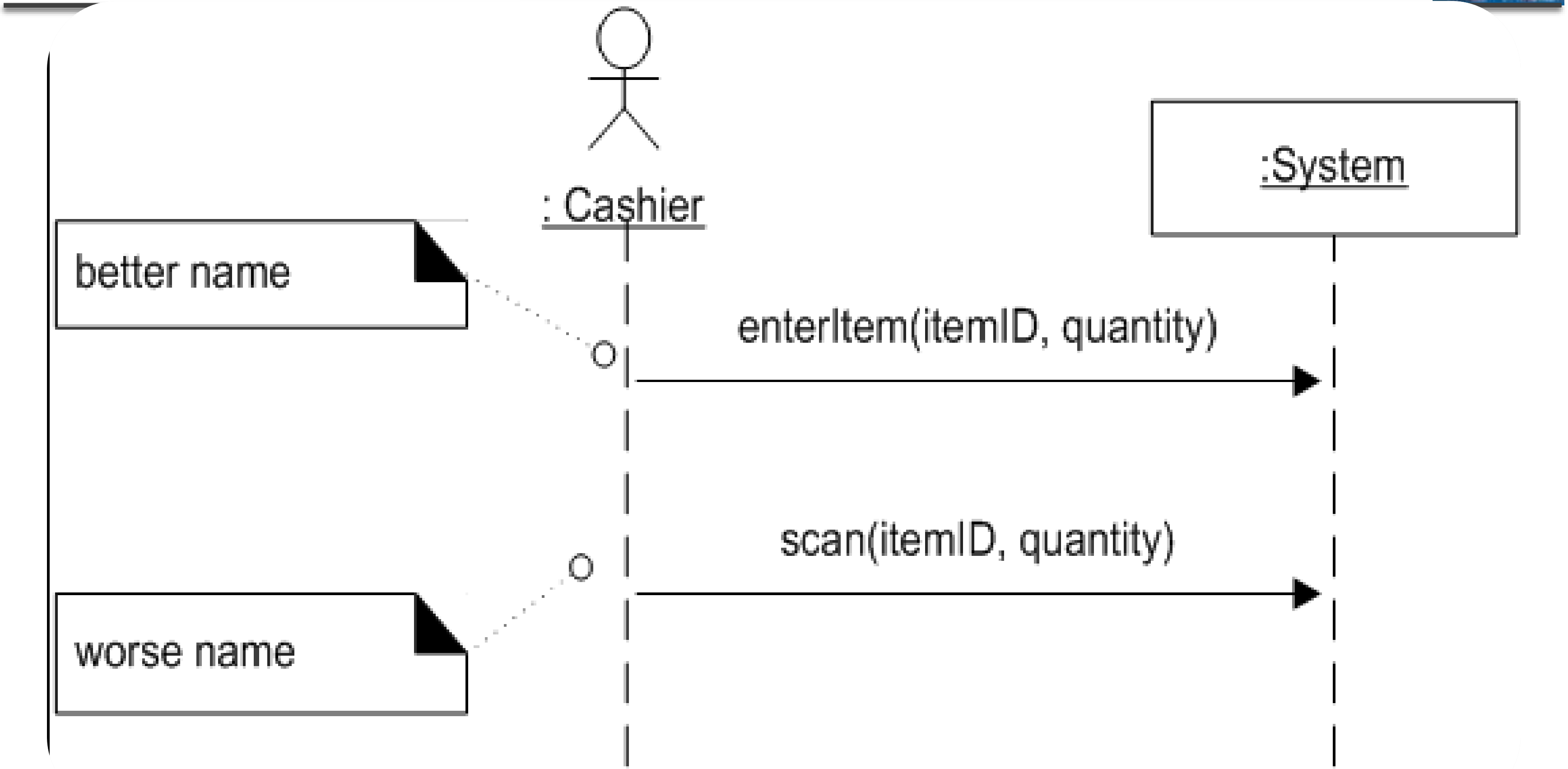
...

Step 1: Define System Events and the System Boundary



the system boundary is usually chosen to be the software system itself; in this context, a system event is an external event that directly stimulates the software

Step 2: Naming System Events and Operations



Step 3: show Use Case

✧Simple **cash-only *Process Sale*** scenario:

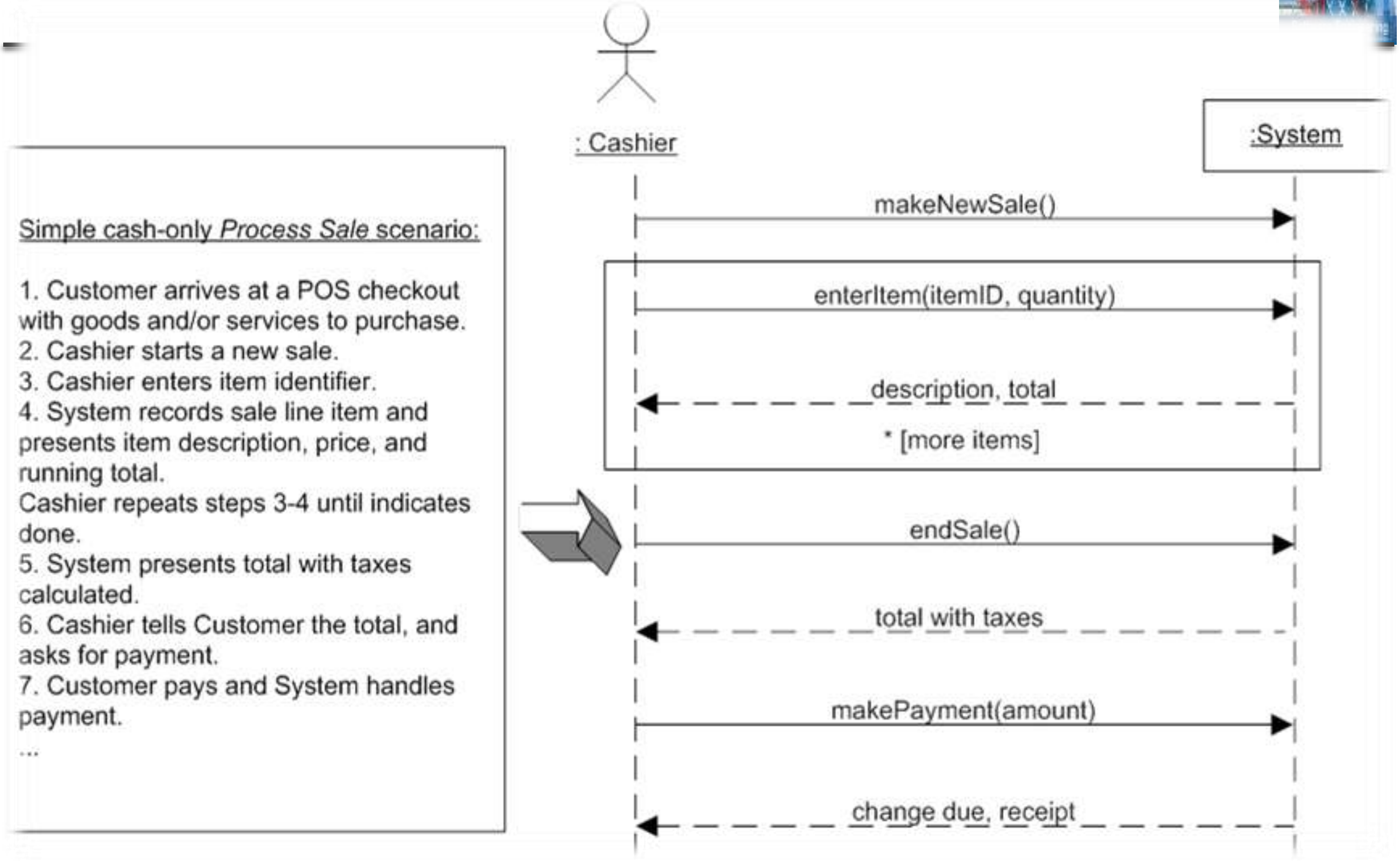
1. Customer arrives at a POS checkout with goods and/or services to purchase.
2. Cashier starts a new sale.
3. Cashier enters item identifier.
4. System records sale line item and presents item description, price, and running total.

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.

...

Step 4: Do SSD



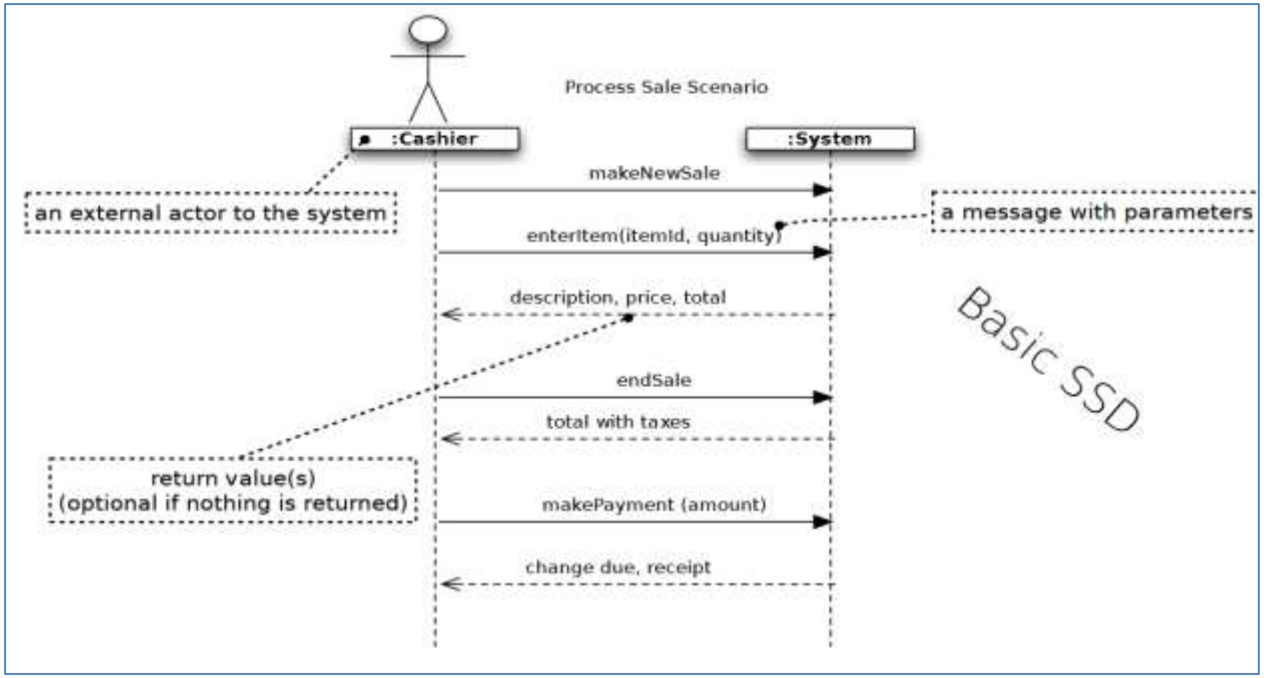
Simple cash-only *Process Sale* scenario:

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.
...

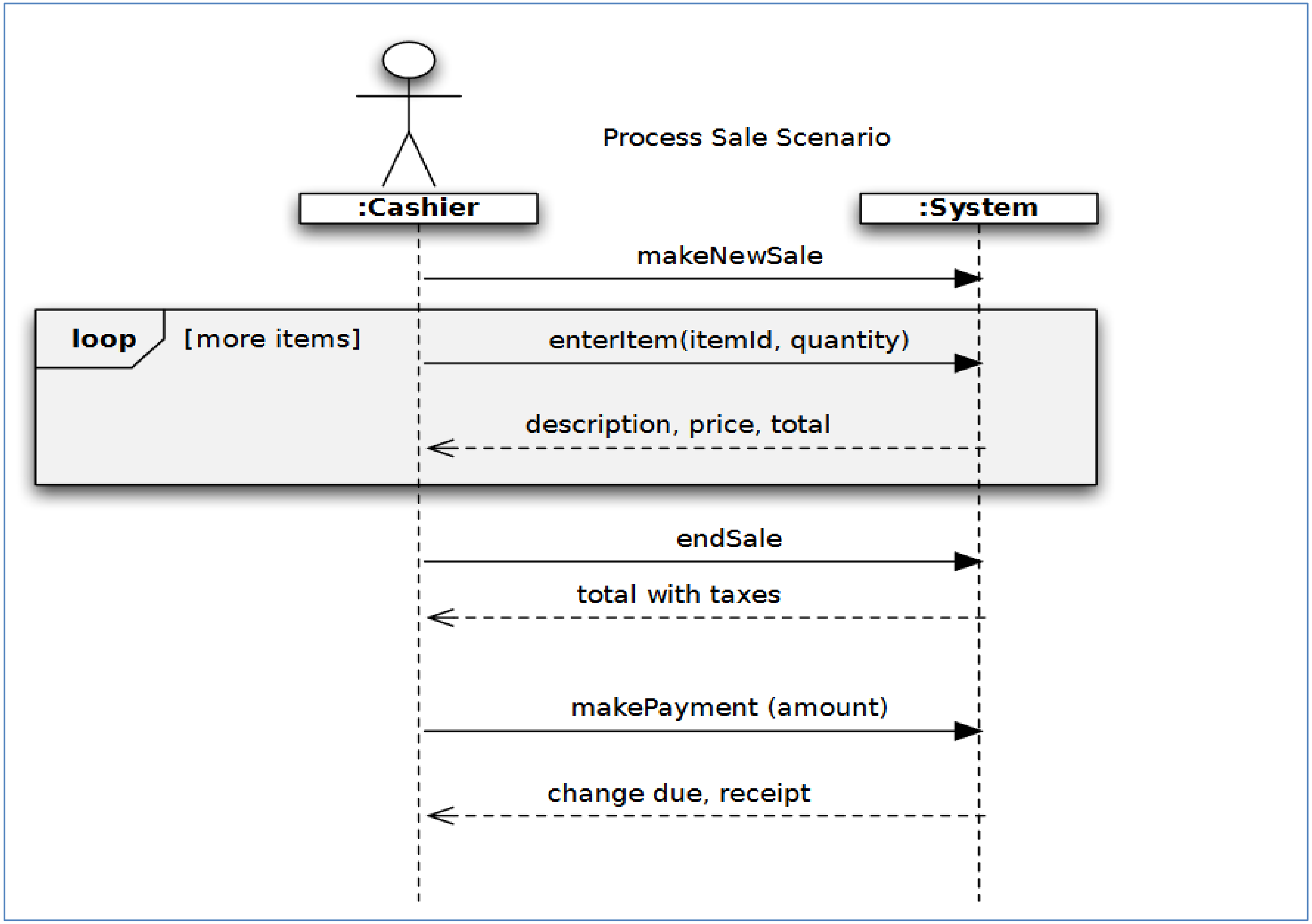
Example of a SD (Sequence Diagram) for the Process Sale Scenario

✧ Use Case: Process Sale Scenario - Main Success Story

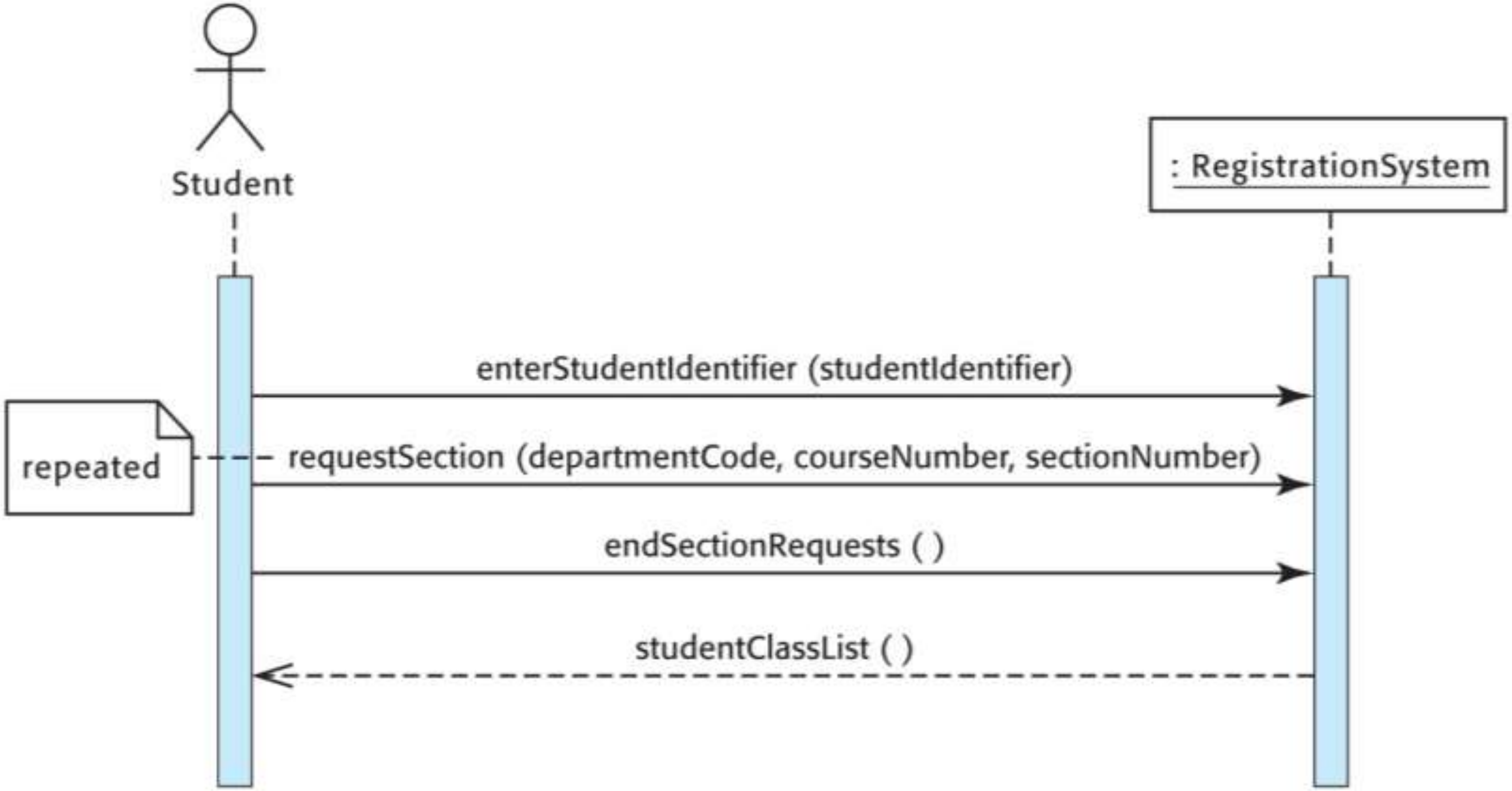
1. Cashier starts new sale
2. Cashier enters item identifier
3. System records sale line item and presents item description, price and running total
   Steps 2 and 3 are repeated until all items are processed.
4. System presents total with taxes calculated
5. Cashier tells Customer the total and asks for payment
6. Customer pays and System handles payment


Process Sale Scenario
:Cashier
:System
makeNewSale
an external actor to the system
a message with parameters
enterItem(itemId, quantity)
description, price, total
endSale
total with taxes
return value(s)
(optional if nothing is returned)
makePayment (amount)
change due, receipt
Basic SSD

Example of a SD (Sequence Diagram) for the Process Sale Scenario



Example - Sequence Diagram


Student
: RegistrationSystem
enterStudentIdentifier (studentIdentifier)
repeated
requestSection (departmentCode, courseNumber, sectionNumber)
endSectionRequests ( )
studentClassList ( )