Information Systems Analysis & Modeling

Lecture 9: Sequence diagram

Mona Taghavi

LaSalle College
Montréal

The System Sequence Diagram—Identifying Inputs and Outputs

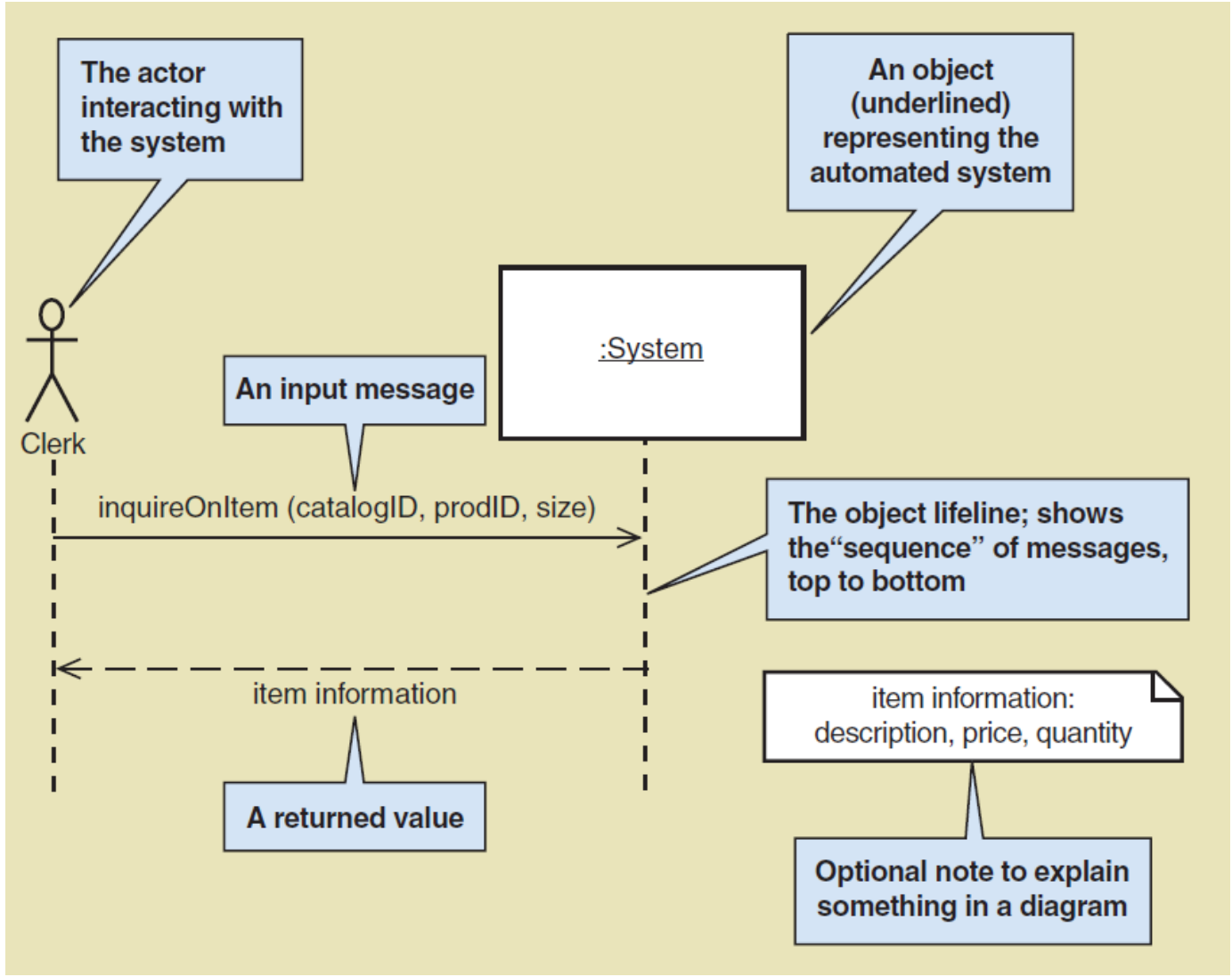
- Sequence diagram: an “interaction diagram” that models a single scenario executing in a system
  - 2nd most used UML diagram
  - Shows what messages are sent and when
- Documents the inputs and the outputs and identifies the interaction between actors and the system. It is an effective tool to help 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.

Key parts of a sequence diagram

- Participant: an object or an entity; the sequence diagram actor

- Message: communication between objects
- Axes in a sequence diagram:
  - horizontal: which participant is acting
  - vertical: time (↓ forward in time)

Sample system sequence diagram



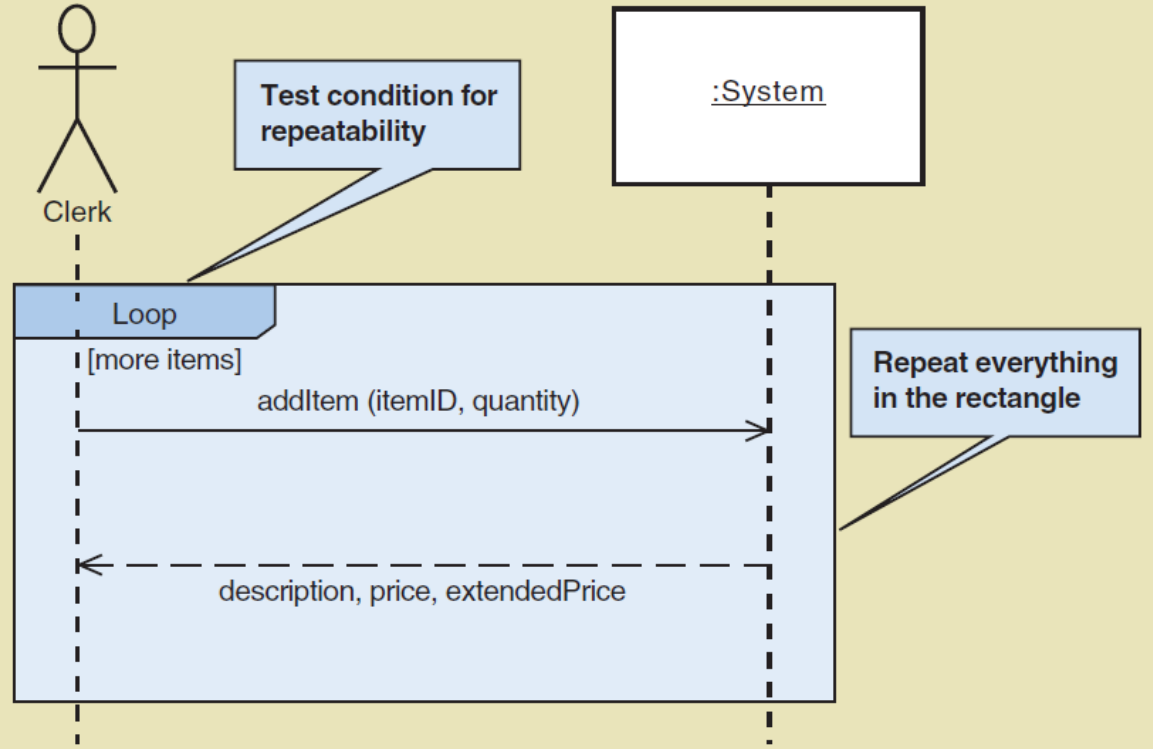
Repeating message in detailed loop frame notation

In ***detailed notation***, the message and its return are located inside a larger rectangle called a **loop frame**.

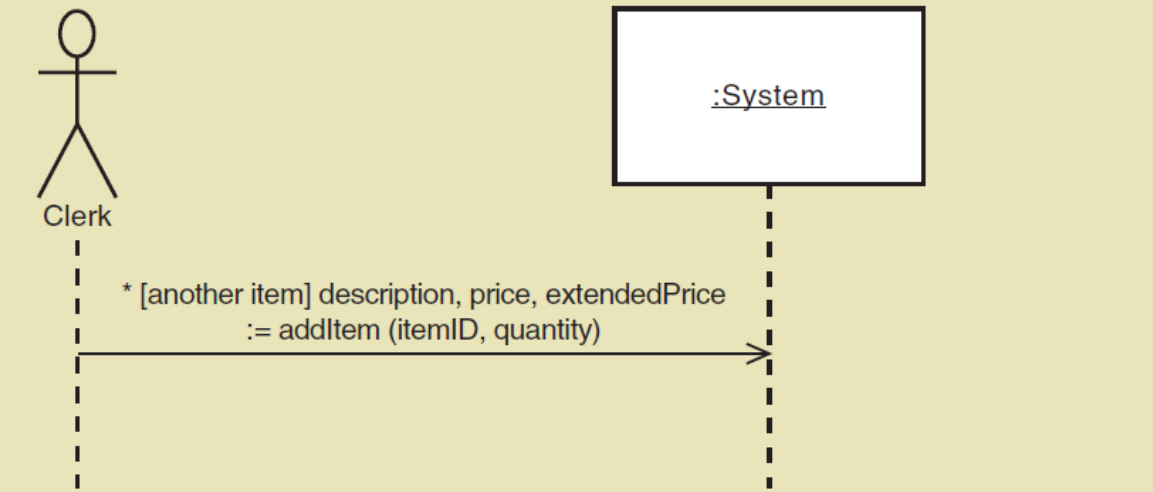
In ***alternate notation***, the square brackets and text inside them are called a **true/false condition** for the messages. The asterisk (*) preceding the true/false condition indicates that the message repeats as long as the true/false condition evaluates to true.
[true/false condition] return-value := message-name (parameter-list)

Pros-Cons of abbreviated notion:

- A message and the returned data can be shown in one step.
- The true/false condition is placed on the message itself.
- If several messages are included within the repeat or there are multiple messages, the loop frame is more explicit and precise.



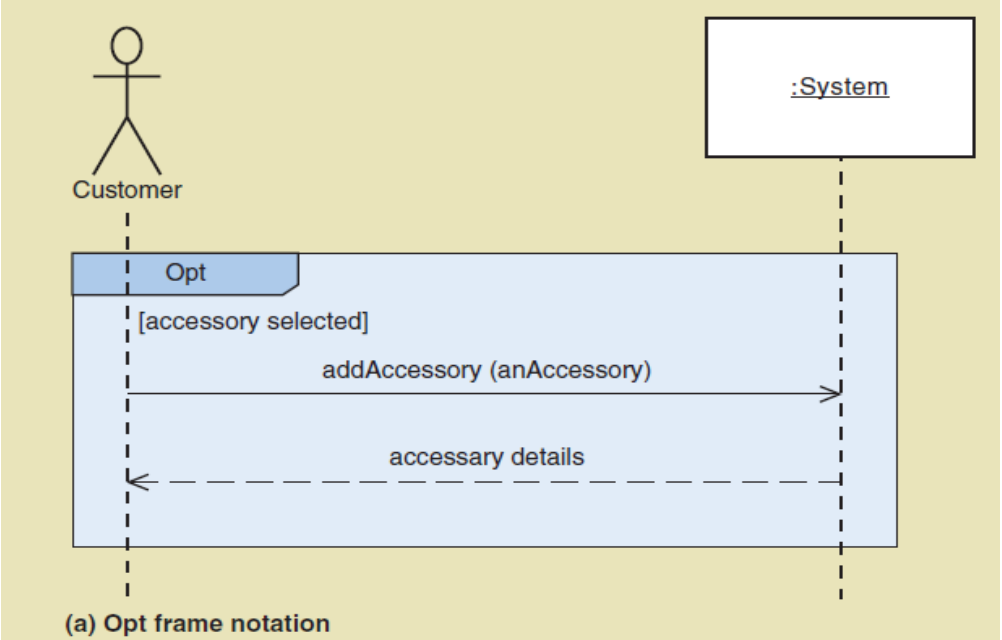
(a) Detailed notation



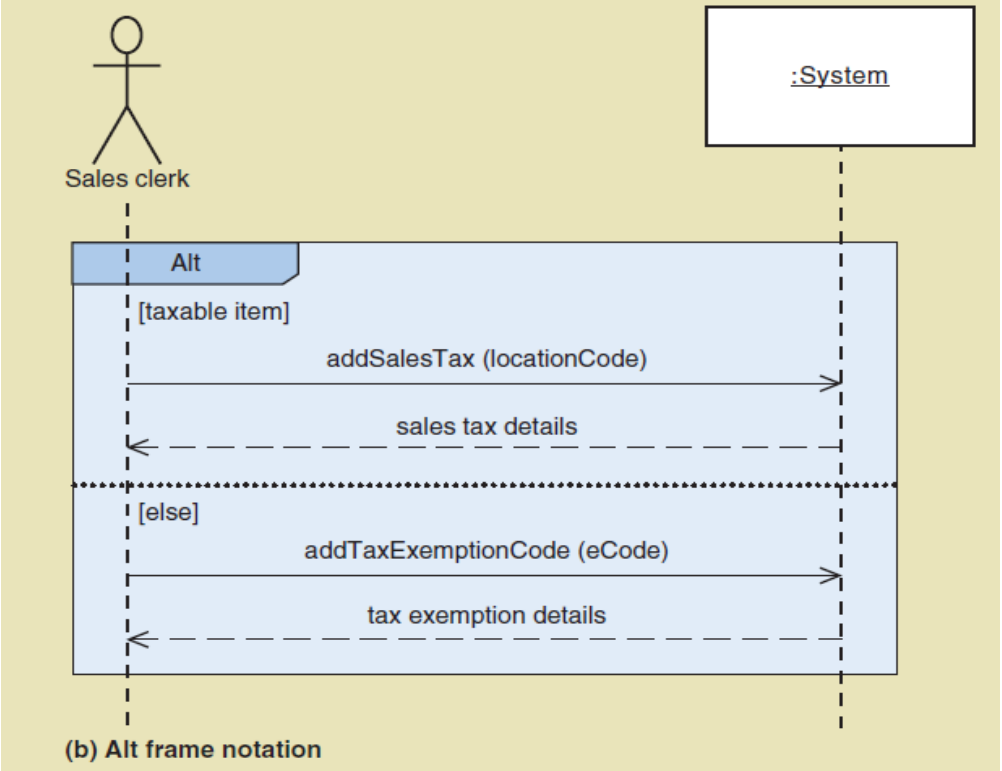
(b) Alternate notation

Sequence diagram notation for opt frame and alt frame

- The **opt frame** in is used when a message or a series of messages is optional or based on some true/false condition.
- The alt frame is used with if-then-else logic. The **alt frame** in the figure indicates that if an item is taxable, then add sales tax; otherwise, add a tax exemption code for a sales tax exemption.



(a) Opt frame notation

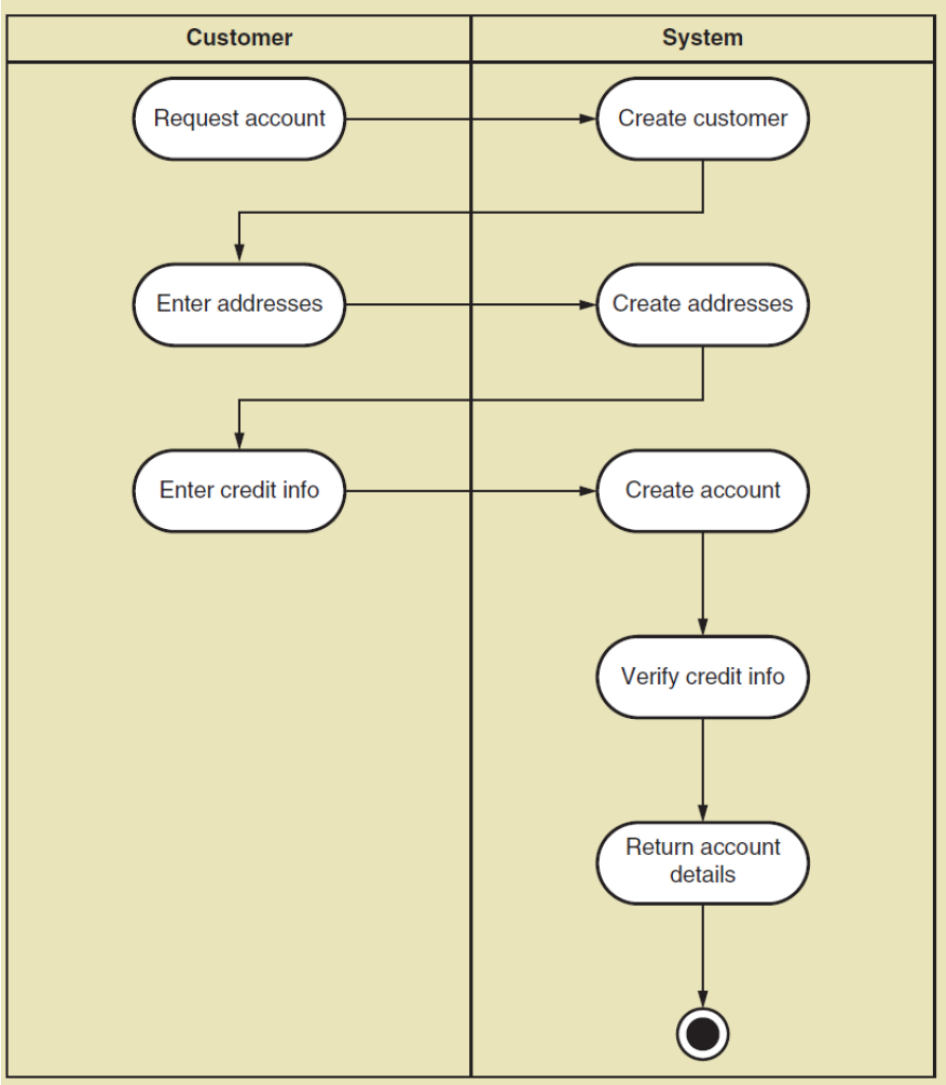


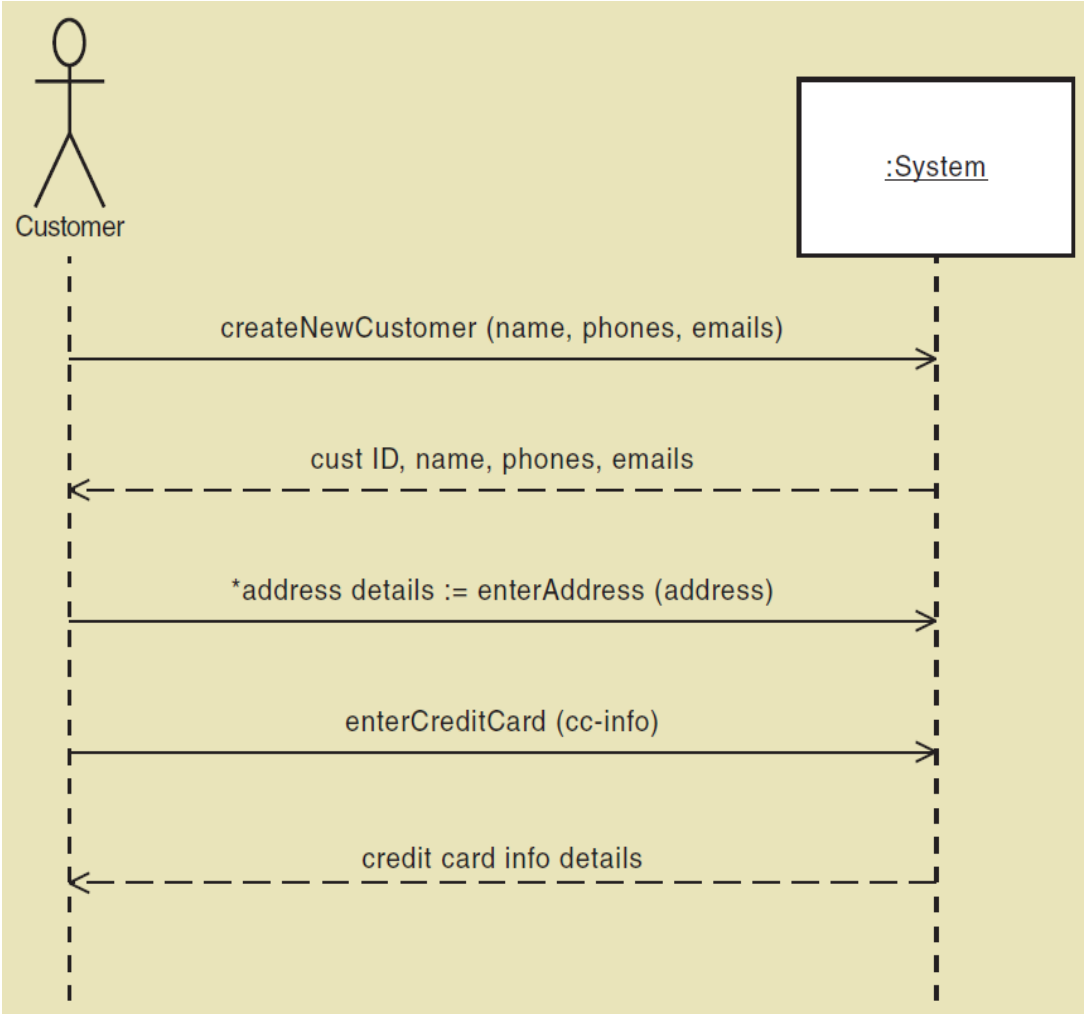
(b) Alt frame notation

The development of an SSD based on an activity diagram

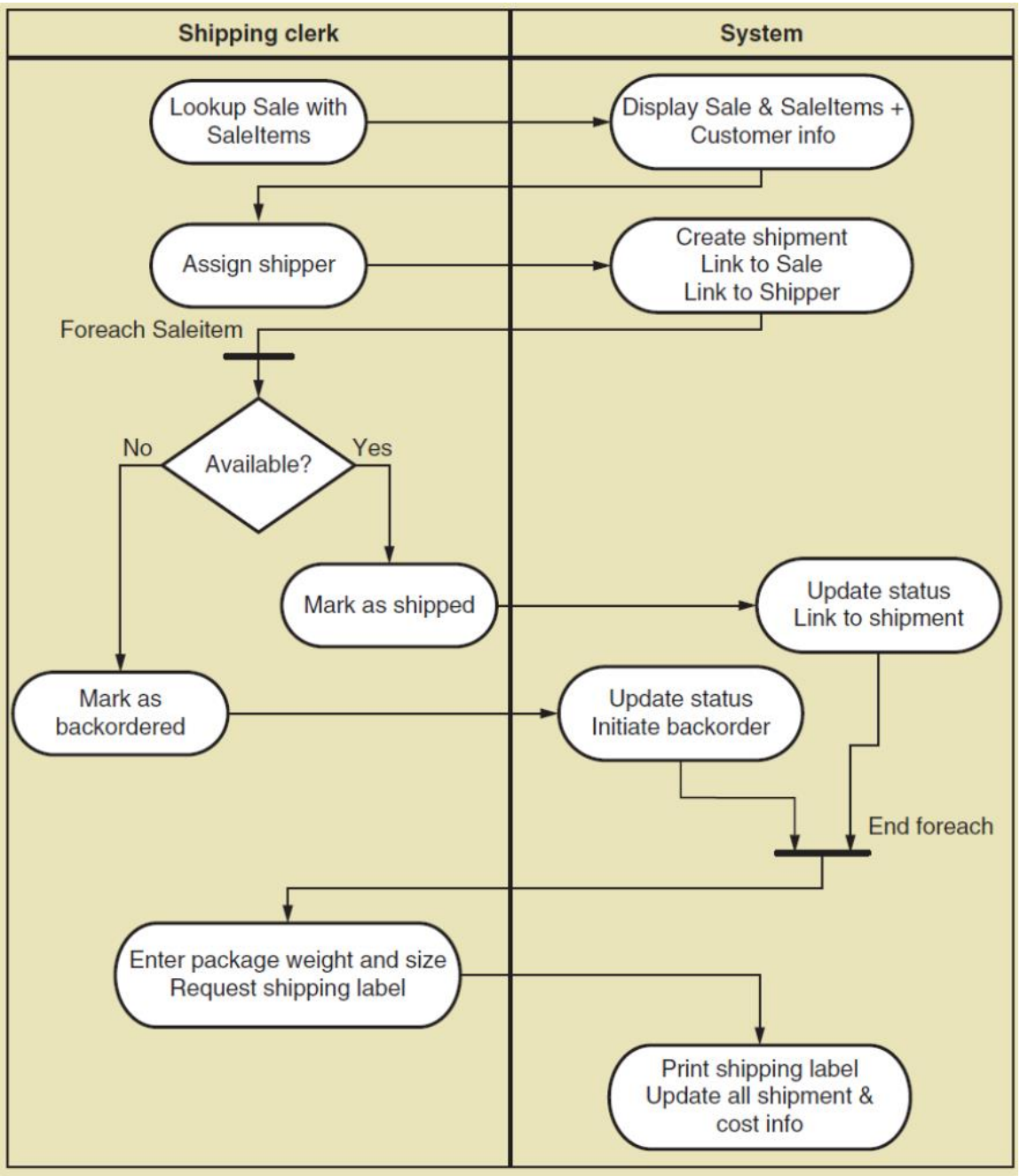
1. Identify the input messages.
   - At each location that the workflow crosses the automation boundary, input data are required; therefore, a message is needed.
2. Describe the message from the external actor to the system by using the message notation described earlier.
   - In most cases, you will need a message name that describes the service requested from the system and the input parameters being passed.
3. Identify and add any special conditions on the input messages, including iteration and true/false conditions.
   - In the next example, the enterAddress message is repeated for each address needed for the customer. The asterisk symbol in front of the message is shown.
4. Identify and add the output return messages.
   - There are two options for showing return information: as a return value on the message itself or as a separate return message with a dashed arrow. The activity diagram can provide some clues about return messages, but there is no standard rule that when a transition arrow in the workflow goes from the system to an external actor that an output always occurs.

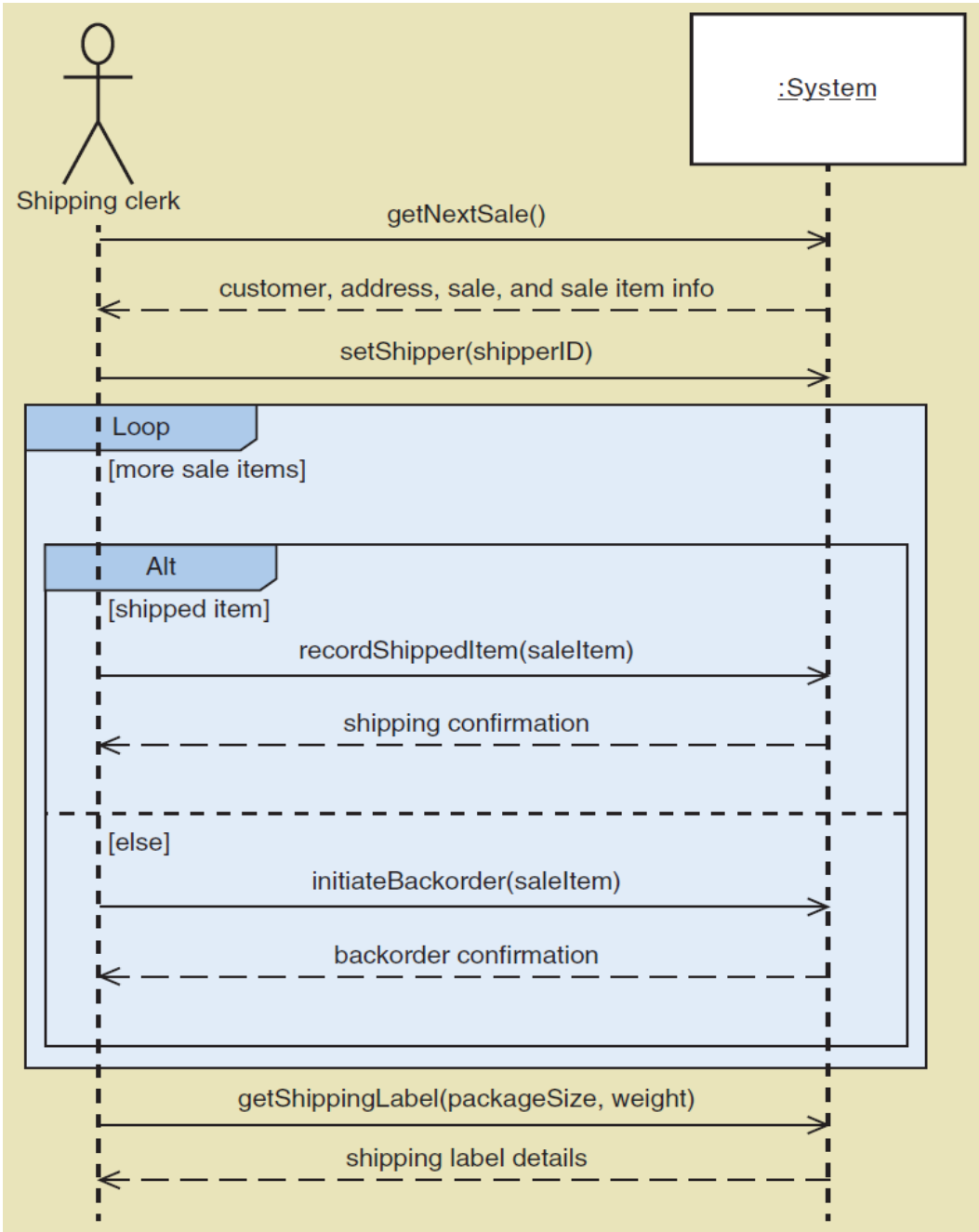
SSD for the “Create customer account use case”





SSD for the Ship items use case





Why use sequence diagrams? Why not code it?

- A good sequence diagram is still above the level of the real code (not all code is drawn on the diagram)
- Sequence diagrams are language-agnostic (can be implemented in many different languages)
- Non-coders can read and write sequence diagrams.
- Easier to do sequence diagrams as a team.
- Can see many objects/classes at a time on the same page (visual bandwidth).

When to use Sequence Diagrams

- Comparing Design Options
  - Shows how objects collaborate to carry out a task
- Assessing Bottlenecks
  - E.g. an object through which many messages pass
- Explaining Design Patterns
  - Enhances structural models.
- Elaborating Use Cases
  - Shows how the user expects to interact with the system
  - Shows how the user interface operates