System Sequence Diagrams

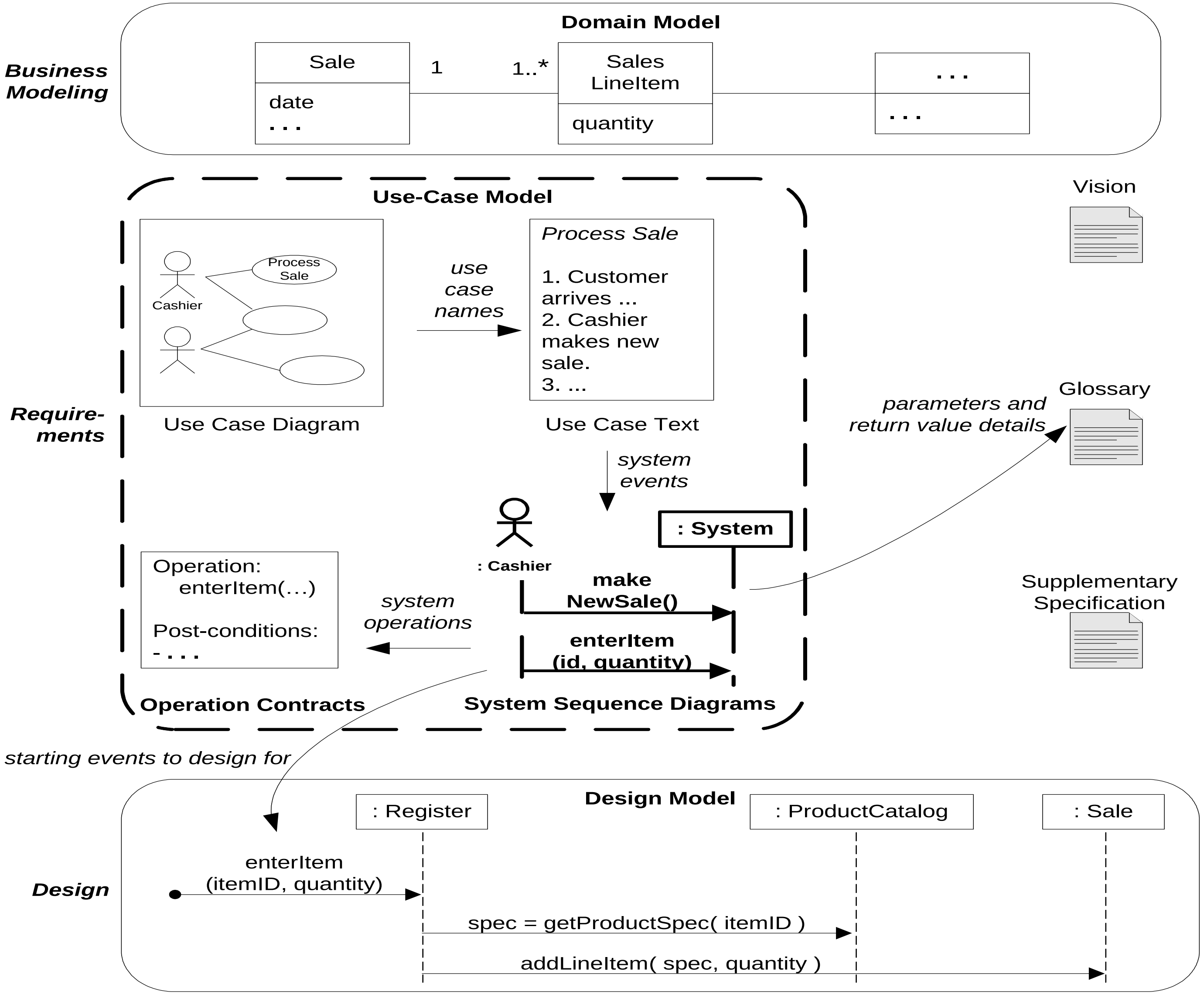
Larman

Chapter 10

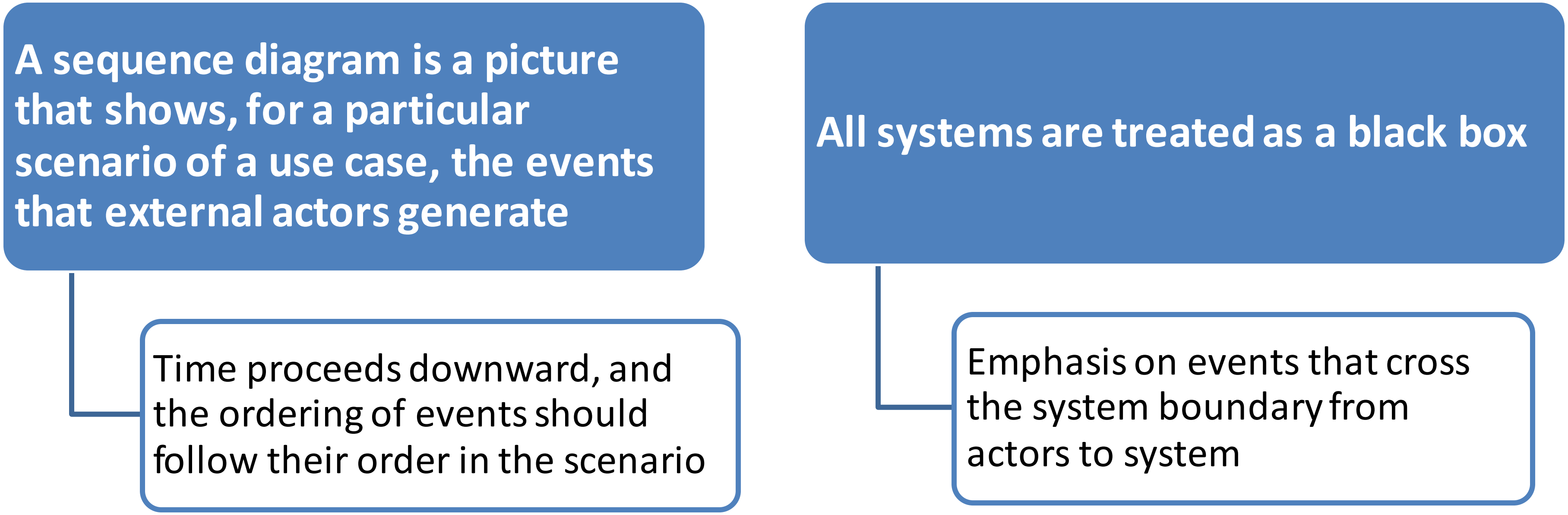
System Sequence Diagrams

- Use cases describe how external actors interact with the software system. During this interaction, an actor generates events.
- Used to model input and output messaging requirements for a use case or scenario
- A system sequence diagram (SSD) is a fast and easily created artifact that illustrates input and output events related to the system.
- Shows sequence of interactions as messages during flow of activities
- System is shown as one object: a “black box”
- SSDs are input to operation contracts and object design.

Sample UP Artifact Relationships



System Sequence Diagrams (SSDs)



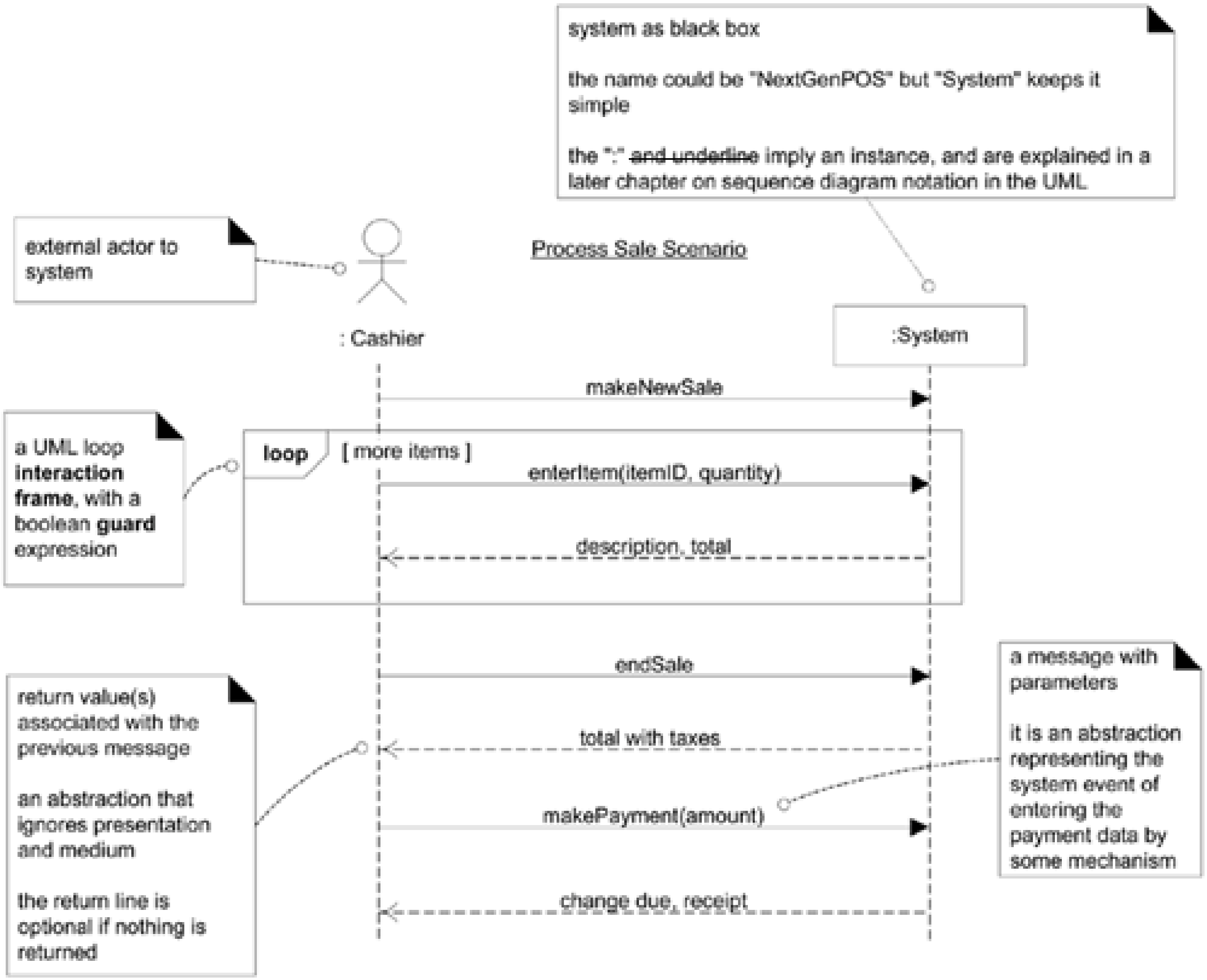
A sequence diagram is a picture that shows, for a particular scenario of a use case, the events that external actors generate

Time proceeds downward, and the ordering of events should follow their order in the scenario

All systems are treated as a black box

Emphasis on events that cross the system boundary from actors to system

Example



system as black box

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

external actor to system

a UML loop **interaction frame**, with a boolean **guard** expression

return value(s) associated with the previous message

an abstraction that ignores presentation and medium

the return line is optional if nothing is returned

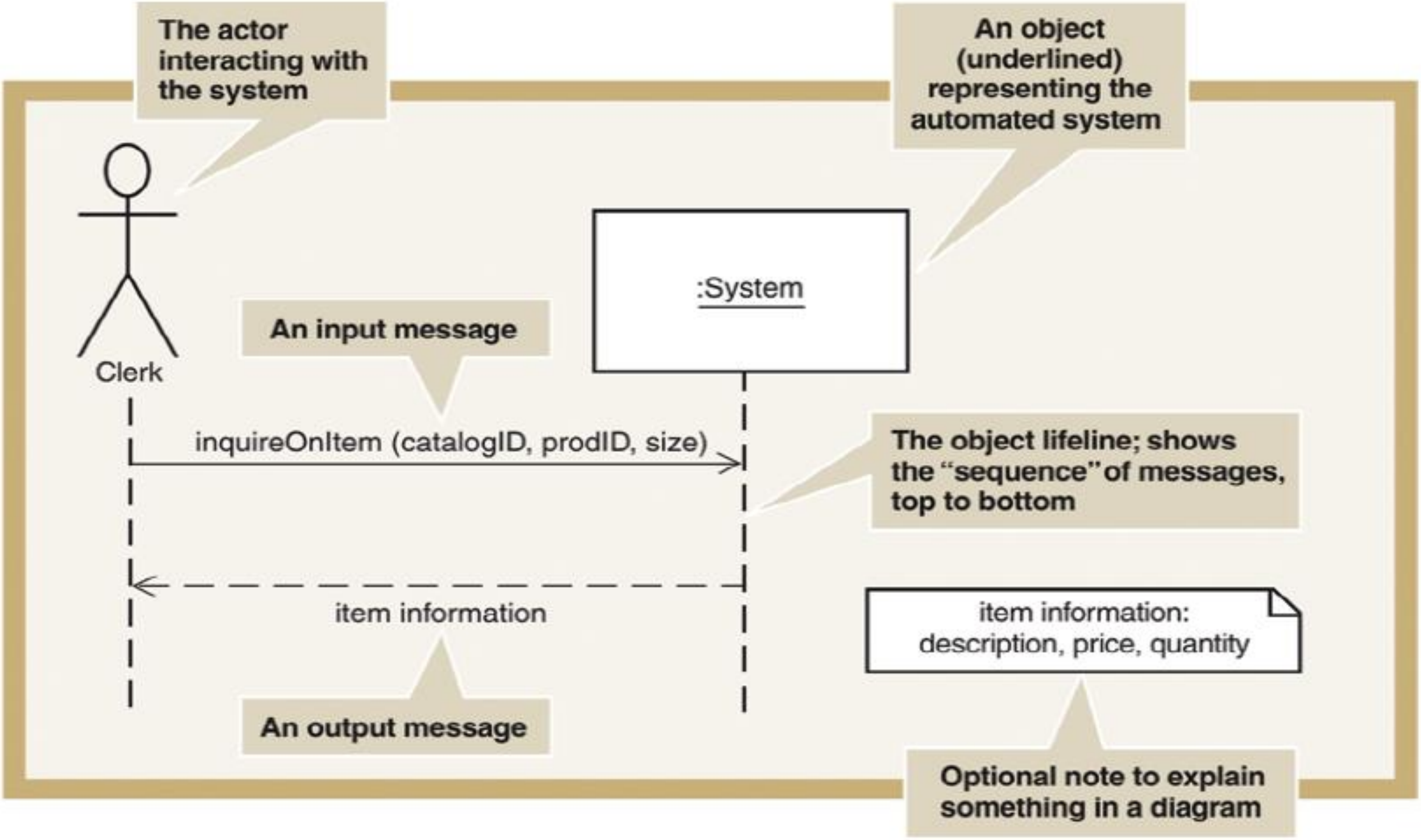
a message with parameters

it is an abstraction representing the system event of entering the payment data by some mechanism

System Sequence Diagrams (SSDs)

- Lifeline or object lifeline is a vertical line under object or actor to show passage of time for object
  - Vertical line under object or actor
    - Shows passage of time
- Message is labeled on arrows to show messages sent to or received by actor or system
- Actor is role interacting with the system with messages

System Sequence Diagram (SSD) Notation


Clerk
:System
inquireOnItem (catalogID, prodID, size)
item information
item information:
description, price, quantity

The actor interacting with the system

An object (underlined) representing the automated system

An input message

The object lifeline; shows the “sequence” of messages, top to bottom

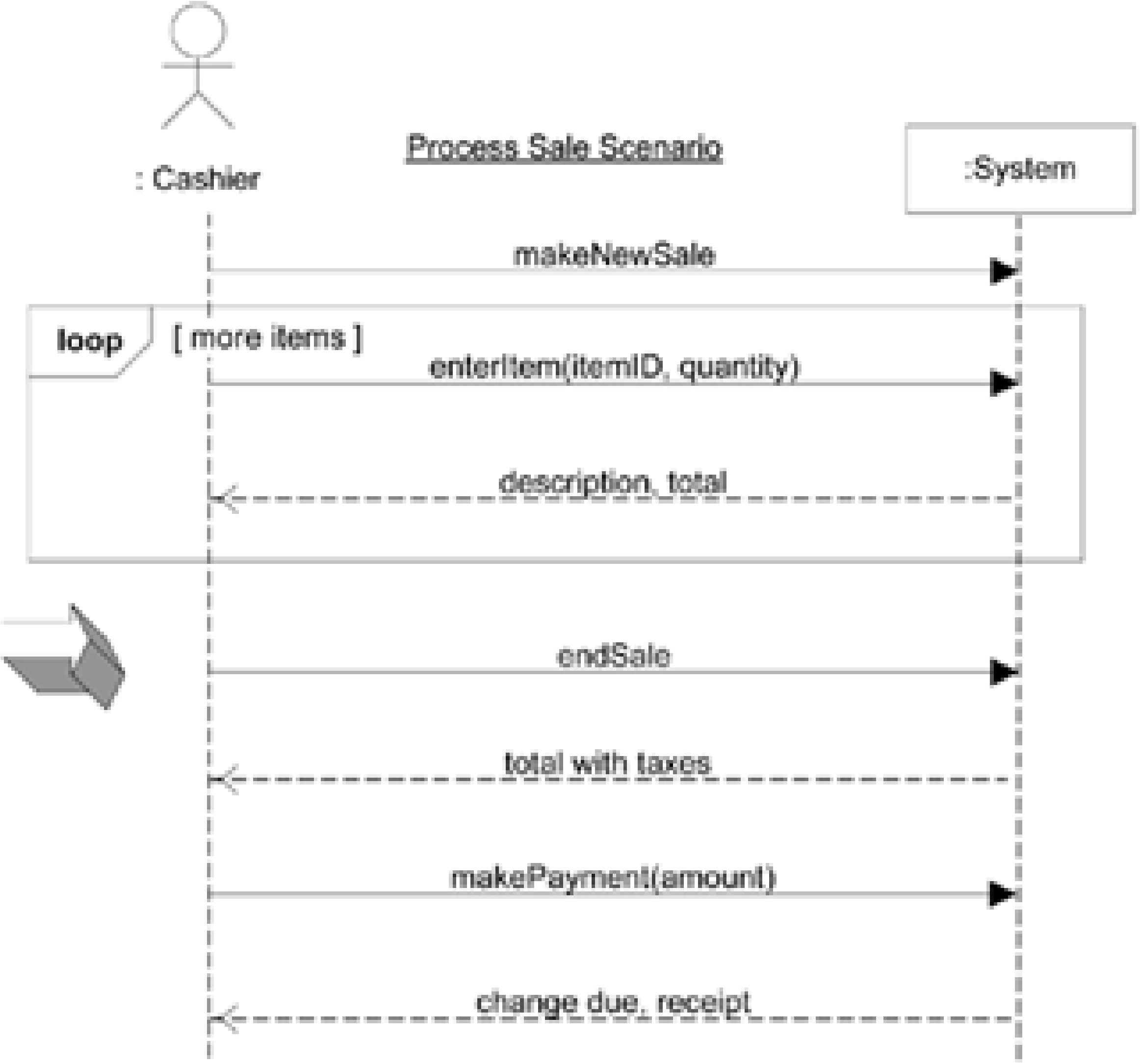
An output message

Optional note to explain something in a diagram

Relationship Between SSDs and Use Cases

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.
...



Naming System Events and Operations

- The set of all required system operations is determined by identifying the system events.
  - makeNewSale()
  - enterItem(itemID, quantity)
  - endSale()
  - makePayment(amount)

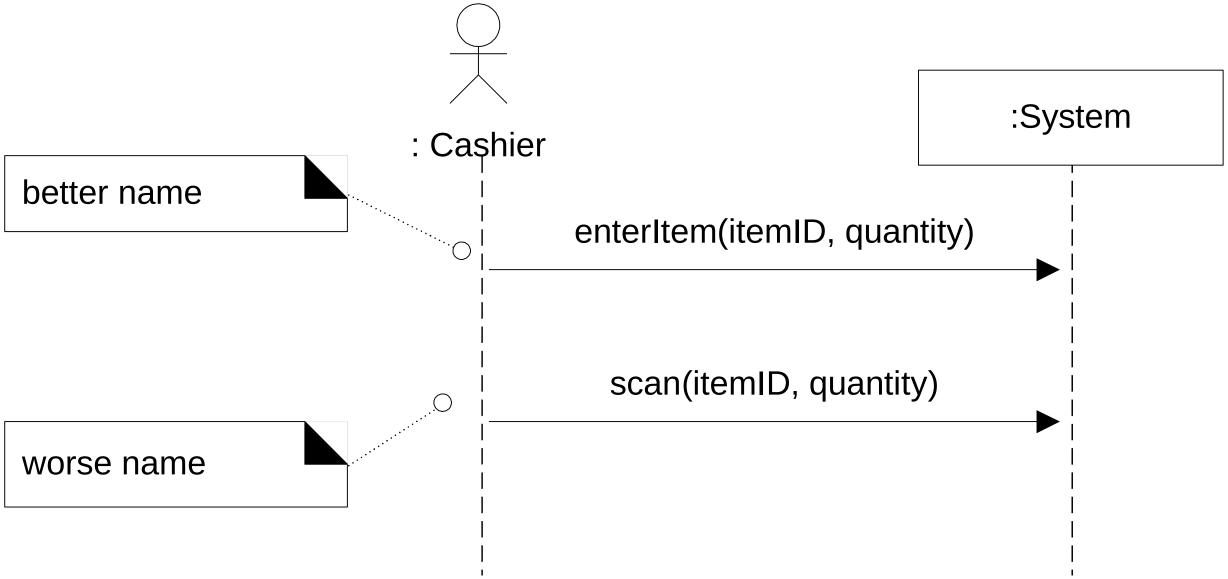
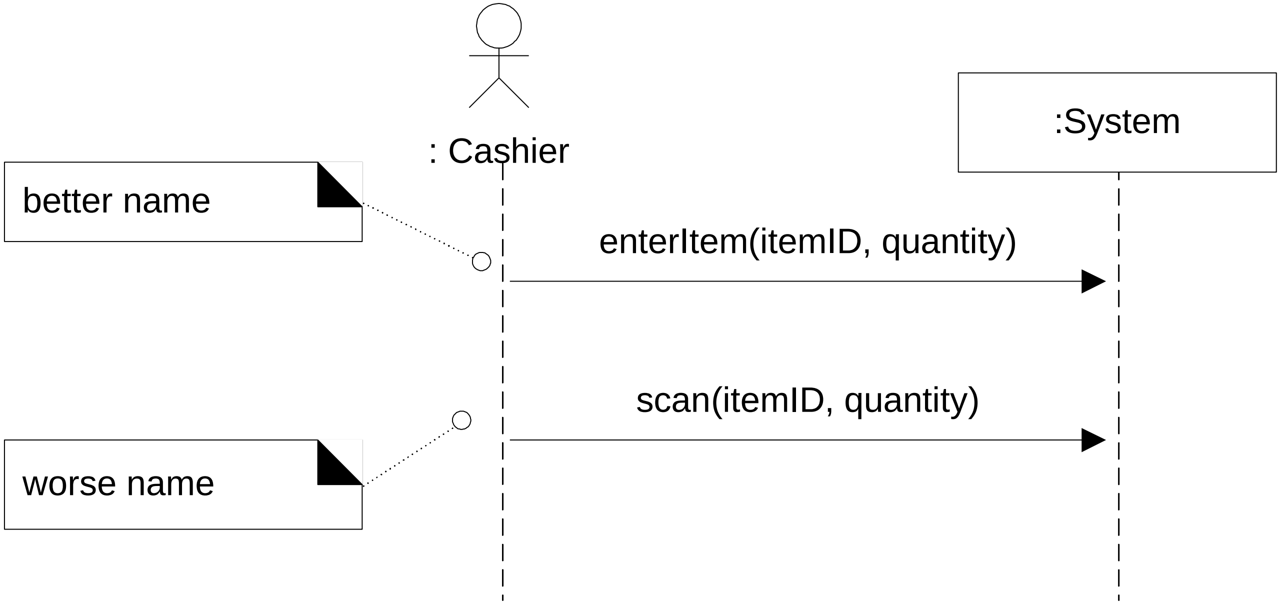


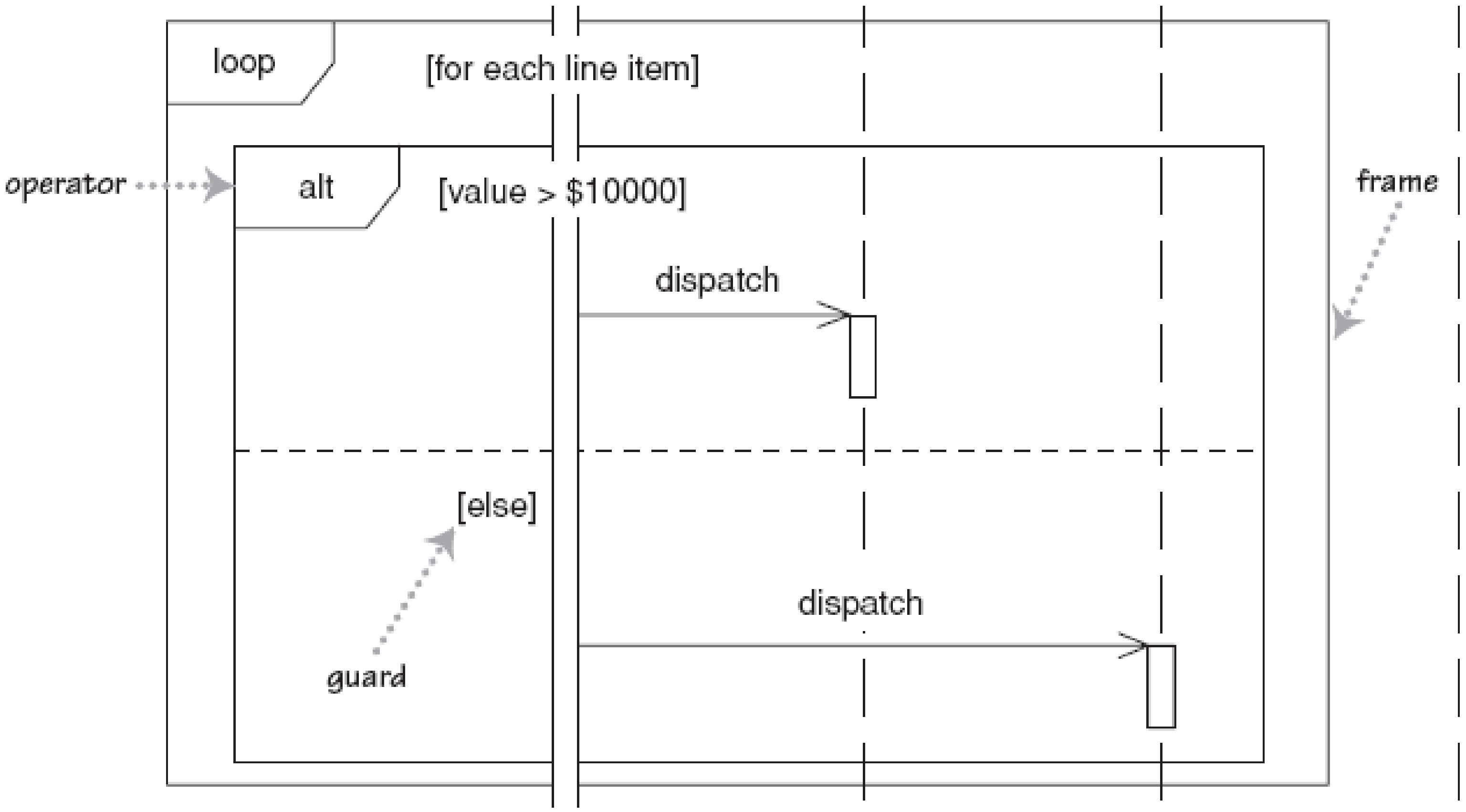
Fig. 10.4


: Cashier
:System
better name
enterItem(itemID, quantity)
scan(itemID, quantity)
worse name

Developing a System Sequence Diagram

- Begin with detailed description of use case from fully developed form or activity diagram
- Identify input messages
- Describe message from external actor to system using message notation
- Identify and add any special conditions on input message, including iteration and true/false conditions
- Identify and add output return messages

Loop and if condition



Conclusions

- We can draw an SSD for a main success scenario of each use case, and frequent or complex alternative scenarios.
- SSDs help to validate, clarify and refine use cases.
- Don't create SSDs for all scenarios, rather draw them only for the scenarios chosen for the next iteration.
- SSDs represent visually the essential aspects of a use case.
- For each event that the system receives from an actor, the system is expected to implement an operation, of the same name, to perform something of value to the actors.