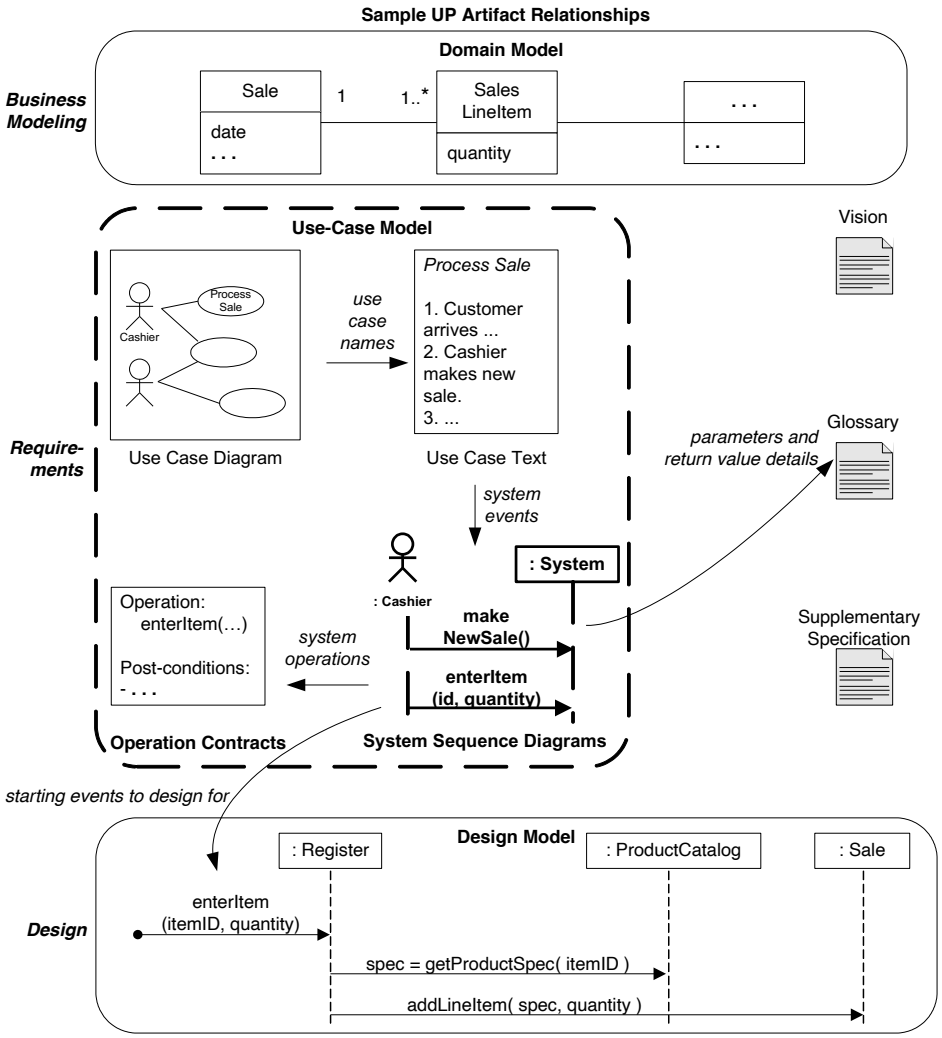
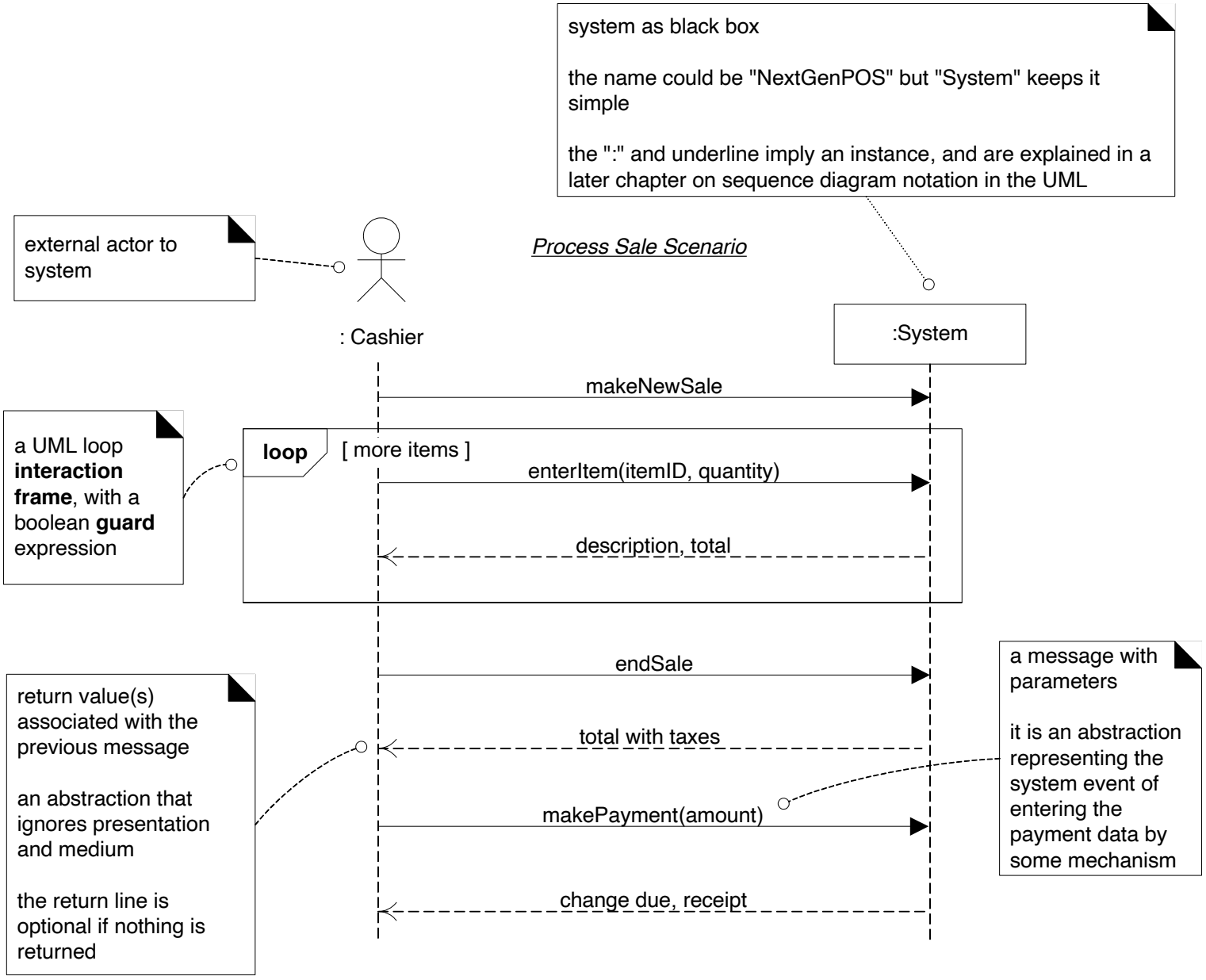
Chapter 10

System sequence diagrams

Fig. 10.1


Sample UP Artifact Relationships
Business Modeling
Domain Model
Sale
date
. . .
1
1..*
Sales LineItem
quantity
. . .
. . .
Requirements
Use-Case Model
Process Sale
Cashier
use case names
Process Sale
1. Customer arrives ...
2. Cashier makes new sale.
3. ...
Use Case Diagram
Use Case Text
Vision
Glossary
parameters and return value details
system events
: System
: Cashier
make NewSale()
enterItem (id, quantity)
Operation:
enterItem(…)
Post-conditions:
- . . .
system operations
Supplementary Specification
Operation Contracts
System Sequence Diagrams
starting events to design for
Design
Design Model
: Register
: ProductCatalog
: Sale
enterItem (itemID, quantity)
spec = getProductSpec(itemID)
addLineItem(spec, quantity)

A System Sequence Diagram



System Sequence Diagrams

- Simple diagrams that illustrate input and output events to the system
- Serve as input to
  - operation contracts
  - object design

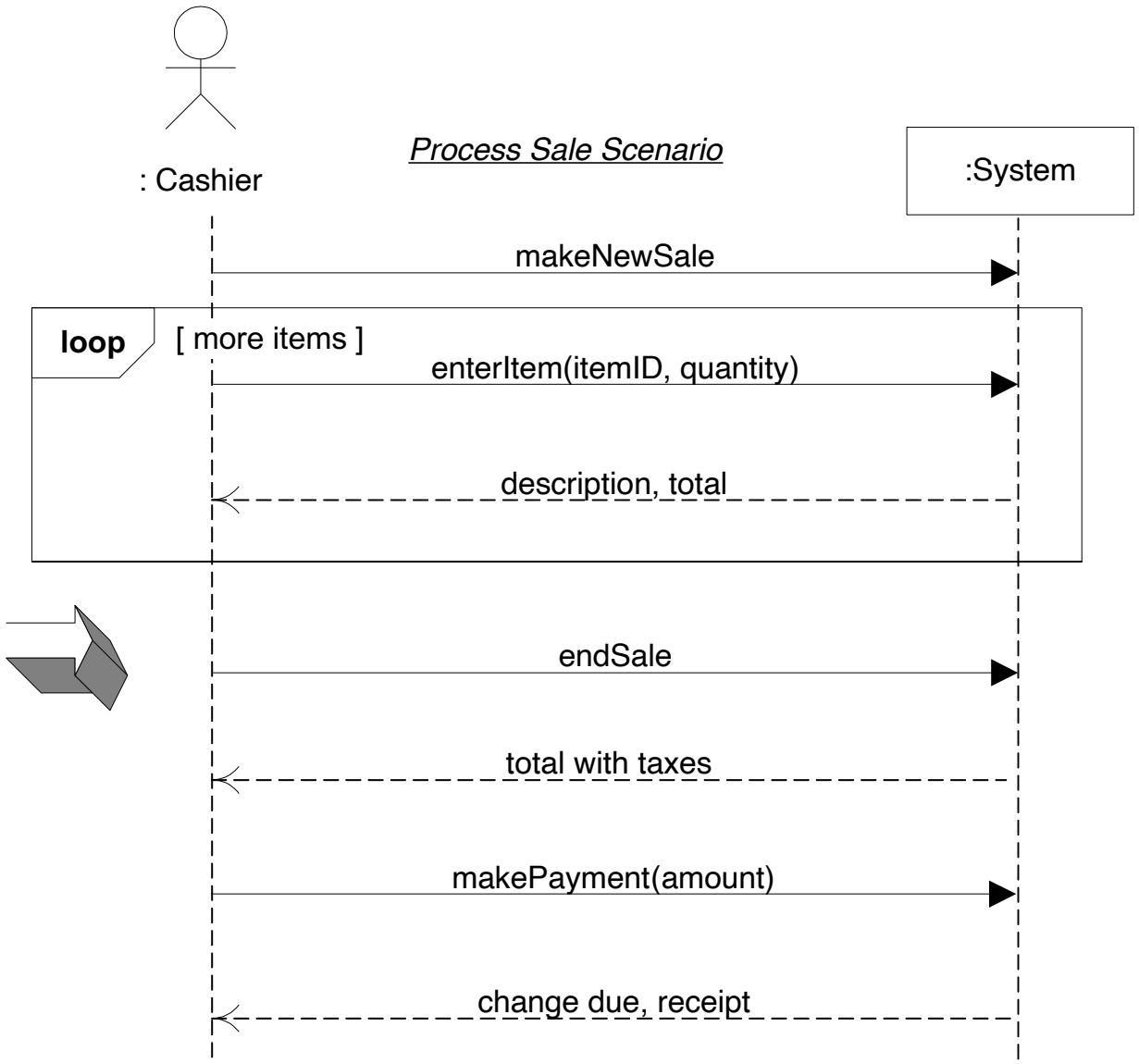
to be covered later

- SSDs are ONE particular use of UML Sequence diagrams
- System treated as black box
  - Interactions with external actors detailed
- Guideline:
  - One SSD per “main success scenario” of a use case
  - Frequent or complex alternative scenarios

SSD for main success scenario of the ProcessSale Use Case

Simple cash-only *Process Sale* scenario:

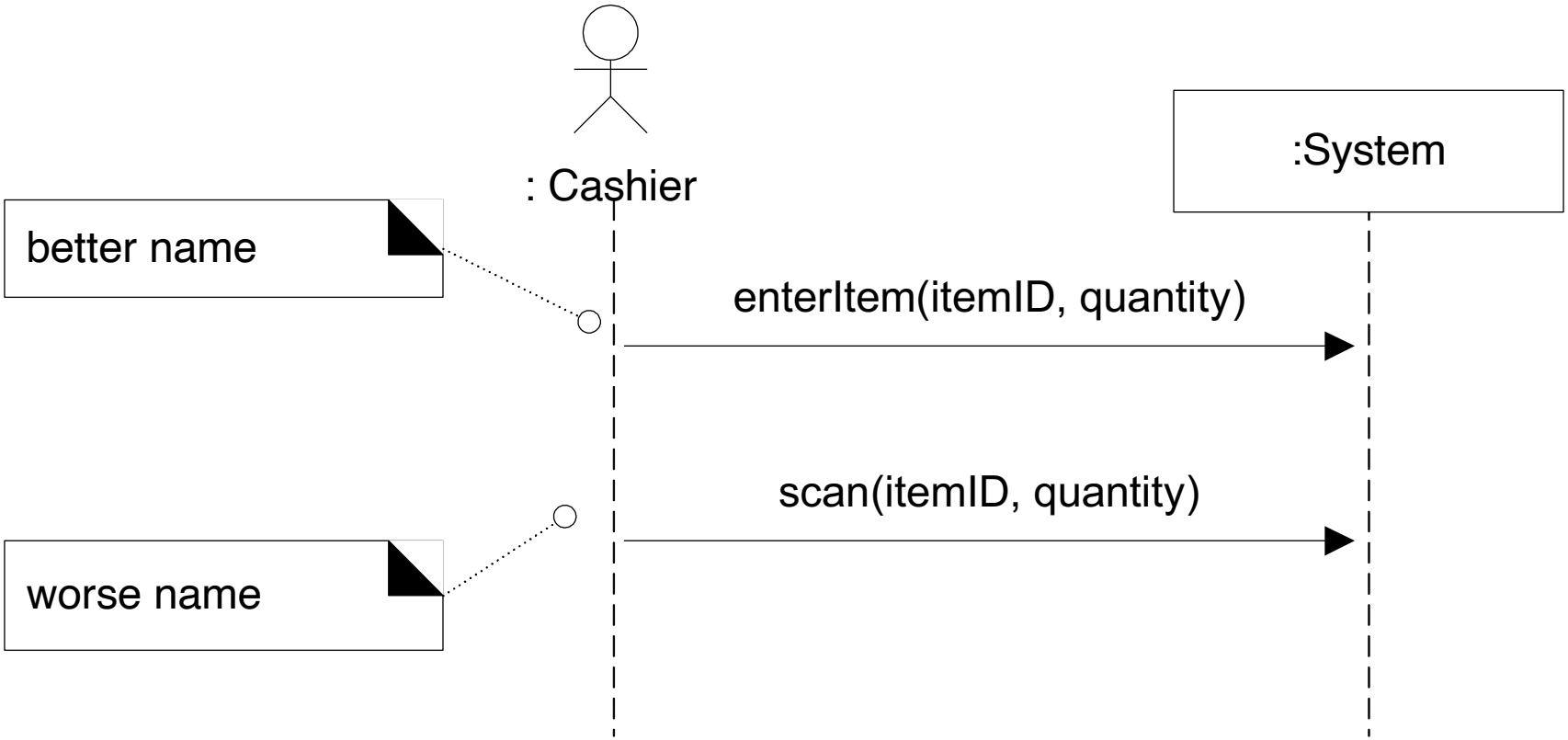
1. Customer arrives at a POS checkout with goods and/or services to purchase.
2. Cashier starts a new sale.
3. Cashier enters item identifier.
4. System records sale line item and presents item description, price, and running total.
Cashier repeats steps 3-4 until indicates done.
5. System presents total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



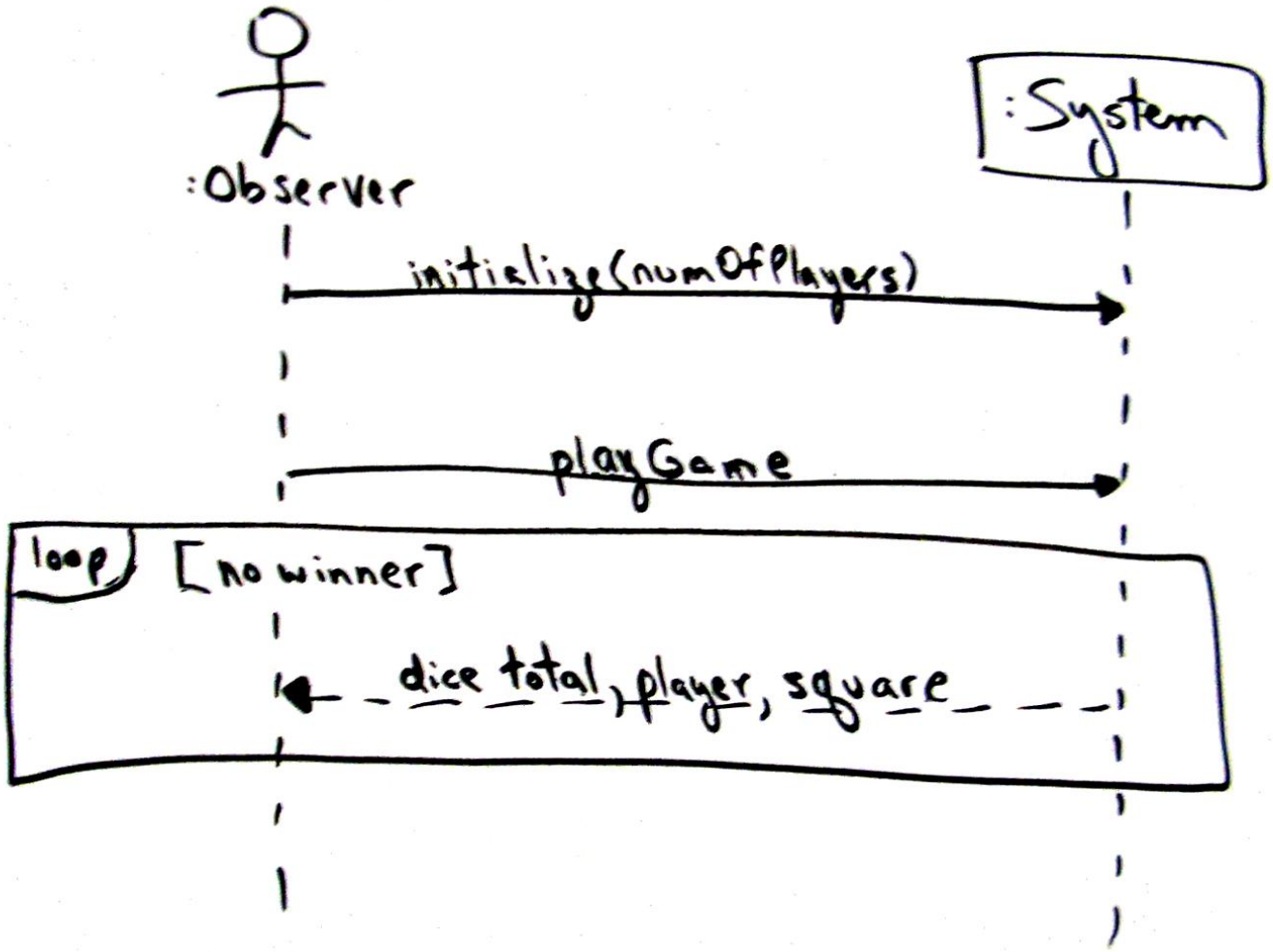
Why draw an SSD?

- Isn't all of this already in the use case narrative?
  - Input and output events along with their arguments (associated information) are made explicit
- Put detailed explanations of certain event names, arguments in the Glossary

Naming Events



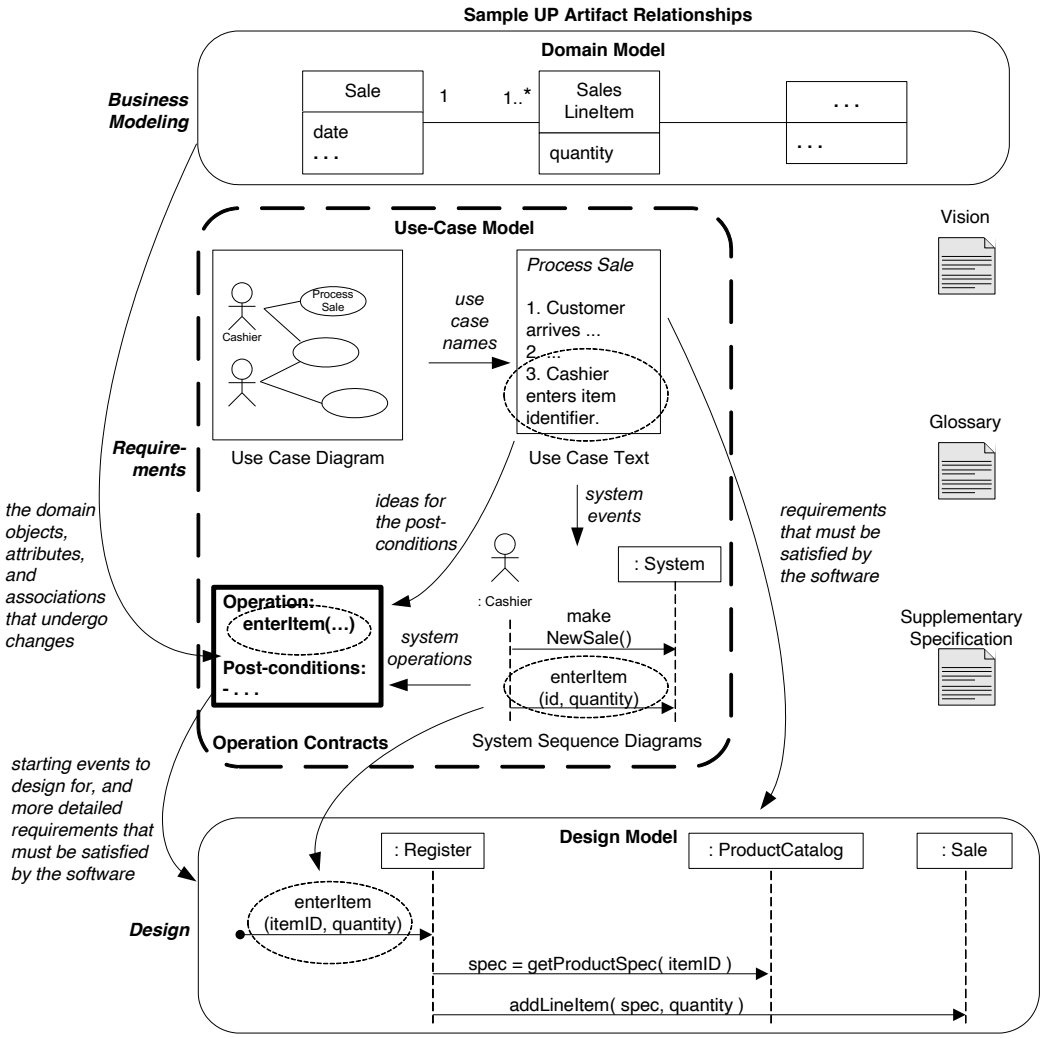
Monopoly SSD example

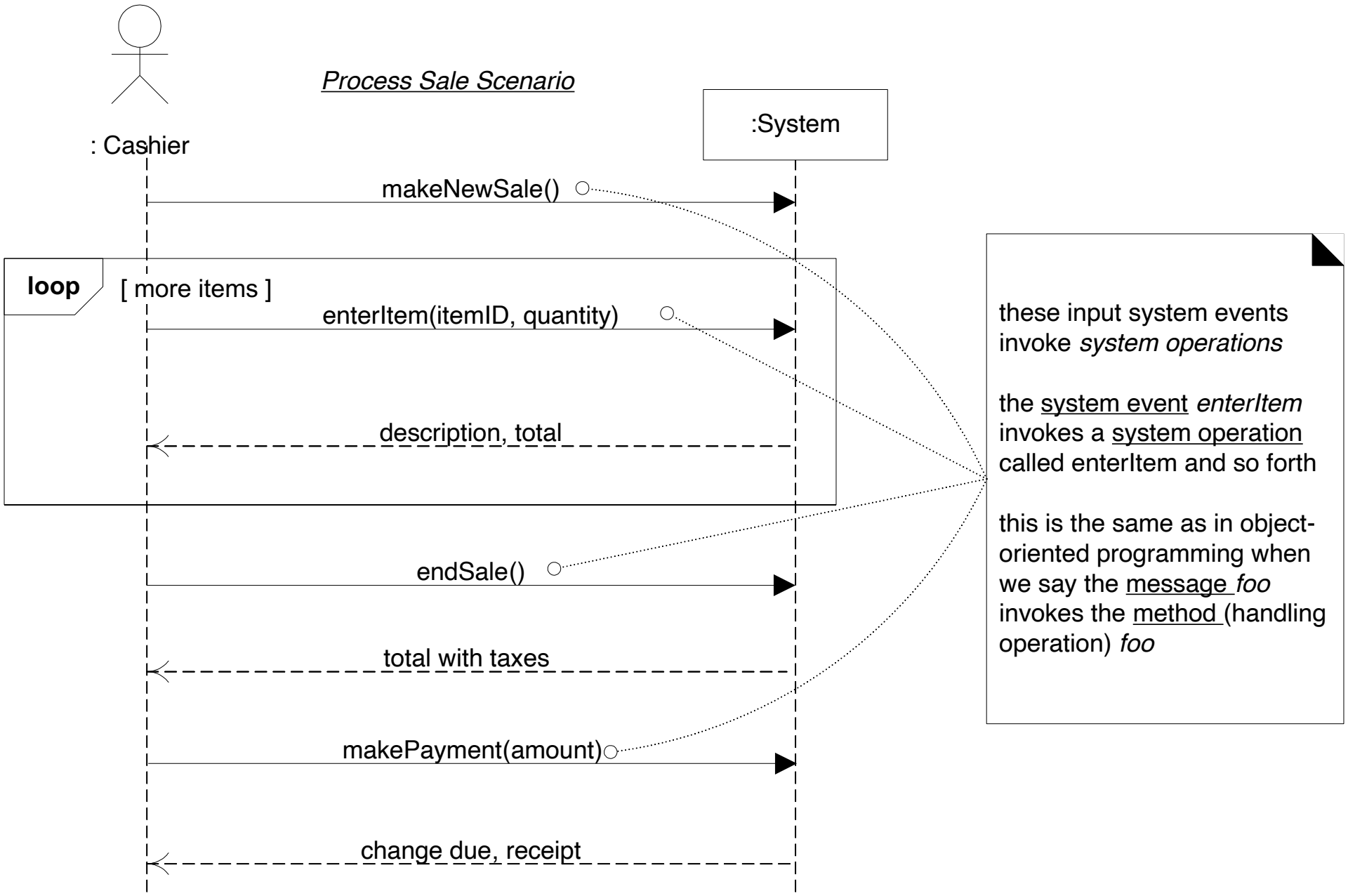


Chapter 11

Operation Contracts

Fig. 11.1


Sample UP Artifact Relationships
Domain Model
Business Modeling
Sale
date
. . .
1
1..*
Sales LineItem
quantity
. . .
. . .
Use-Case Model
Vision
Process Sale
Cashier
use case names
Process Sale
1. Customer arrives ...
2. ...
3. Cashier enters item identifier.
Glossary
Require-ments
Use Case Diagram
Use Case Text
the domain objects, attributes, and associations that undergo changes
ideas for the post-conditions
system events
requirements that must be satisfied by the software
: System
: Cashier
Operation:
enterItem(...)
Post-conditions:
- . . .
system operations
make NewSale()
enterItem (id, quantity)
Supplementary Specification
Operation Contracts
System Sequence Diagrams
starting events to design for, and more detailed requirements that must be satisfied by the software
Design Model
: Register
: ProductCatalog
: Sale
Design
enterItem (itemID, quantity)
spec = getProductSpec(itemID)
addLineItem(spec, quantity)


Process Sale Scenario
: Cashier
:System
makeNewSale()
loop
[more items]
enterItem(itemID, quantity)
description, total
endSale()
total with taxes
makePayment(amount)
change due, receipt
these input system events invoke system operations
the system event enterItem invokes a system operation called enterItem and so forth
this is the same as in object-oriented programming when we say the message foo invokes the method (handling operation) foo

- Operation contracts: Detailed description of changes to objects in domain model in response to events
  - Use pre- and post- conditions
- What is the difference from a use-case narrative?
  - There the pre- and post- conditions are described for the entire use case
  - Here pre- and post- conditions are described per system event

Example contract

Contract CO2: enterItem

Operation: Cross	enterItem(itemID : ItemID, quantity : integer) Use
References:	Cases: Process Sale There is a sale underway.
Preconditions:	
Postconditions:	- A SalesLineItem instance sli was created (instance cre ation). - sli was associated with the current Sale (association formed). -sli.quantity became quantity (attribute modification). - sli was associated with a ProductSpecification, based on itemID match (association formed).

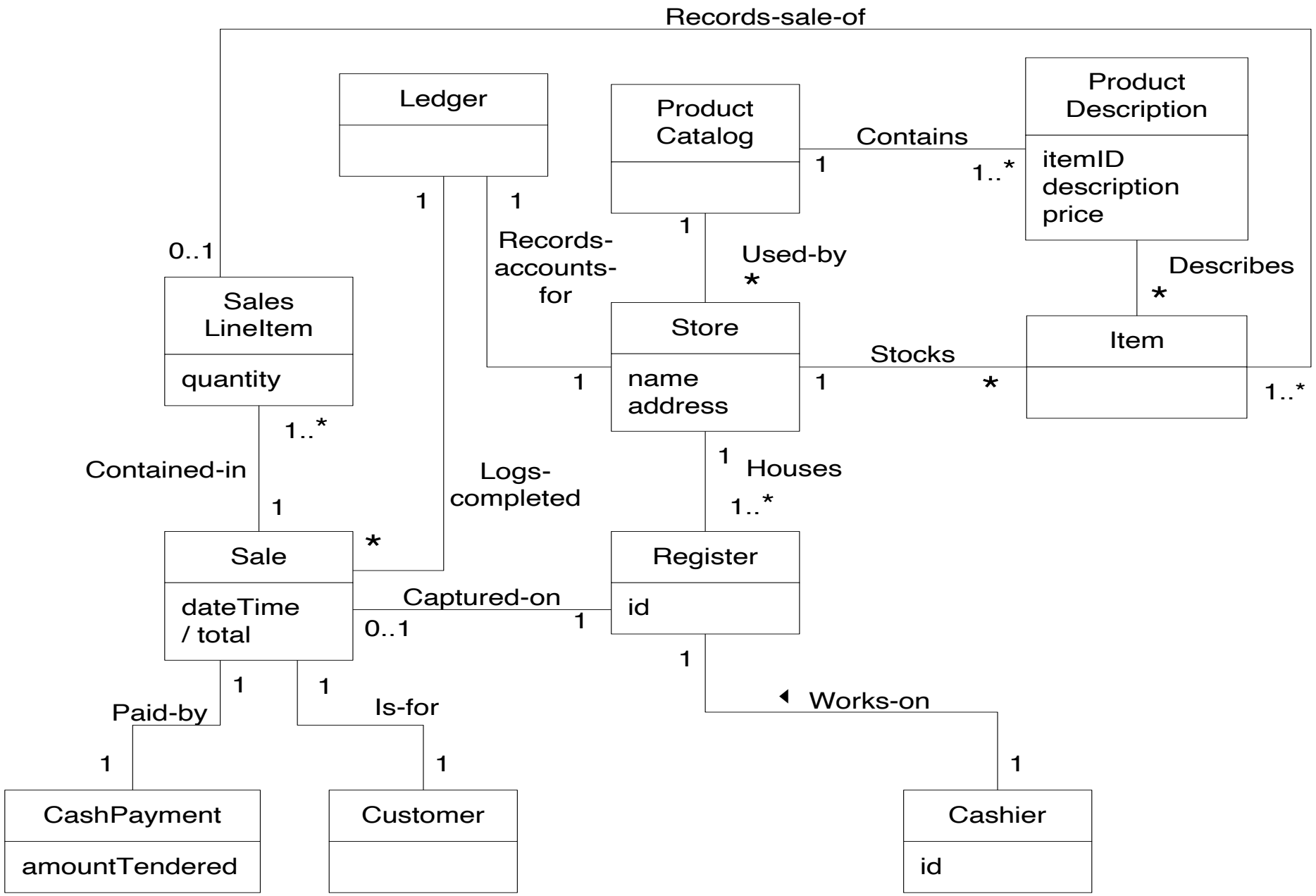
Sections of a contract

Operation: Cross	Name of operation, and parameters
References:	(optional) Use cases this operation can occur within
Preconditions:	Noteworthy *assumptions* about the state of the system or objects in the Domain Model before execution of the operation. These will not be tested within the logic of this operation, are assumed to be true, and are non-trivial assumptions the reader should know were made.
Postconditions:	-The state of objects in the Domain Model after completion of the operation. Discussed in detail in a following section.

Postconditions

- For important, non-trivial system events, describe changes in the state of the objects in the domain model
- Changes include
  - instances created
  - associations (UML links) formed or broken
  - attributes changed
- May result in updates to the domain model
  - Additions of new attributes and associations

POS Domain Model Example:



Example contract

Contract CO4: makePayment

Operation: Cross	makePayment(amount: Money) Use
References:	Cases: Process Sale There is a sale
Preconditions:	underway.
Postconditions:	- *A Payment instance p was created (instance creation).* - *p.amountTendered became amount (attribute modification).* - *p was associated with the current Sale (association formed).* - *The current Sale was associated with the Store (associa tion formed); (to add it to the historical log of completed sales)*