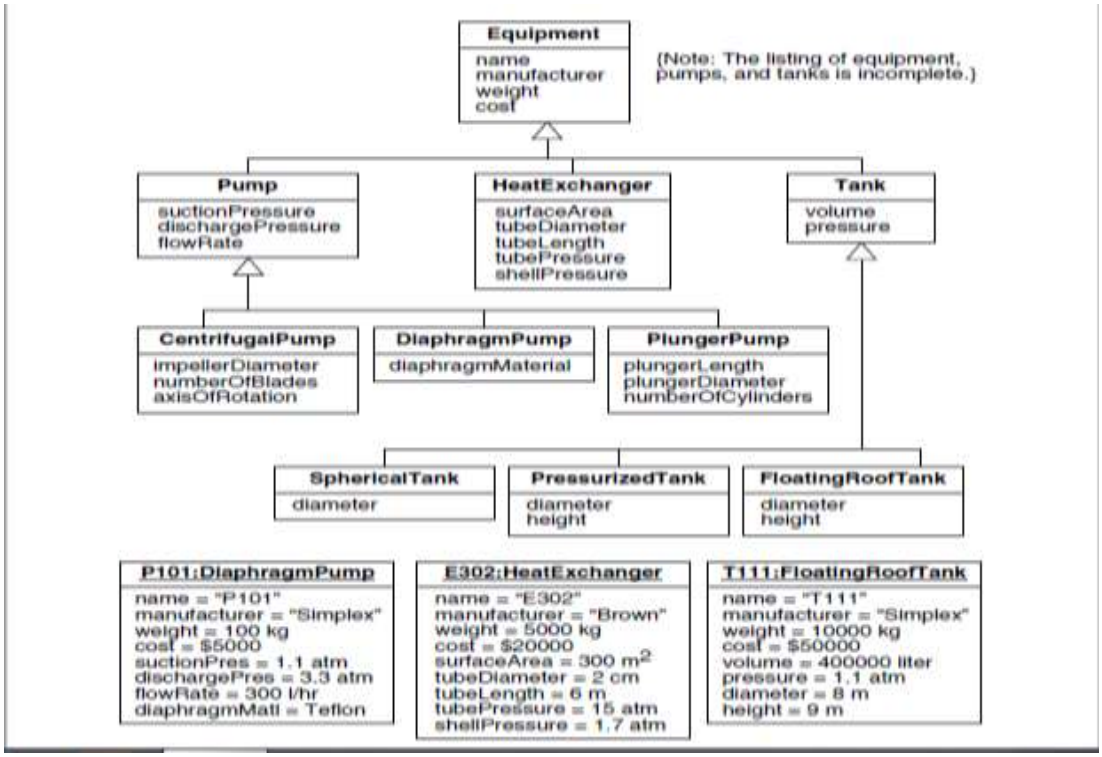
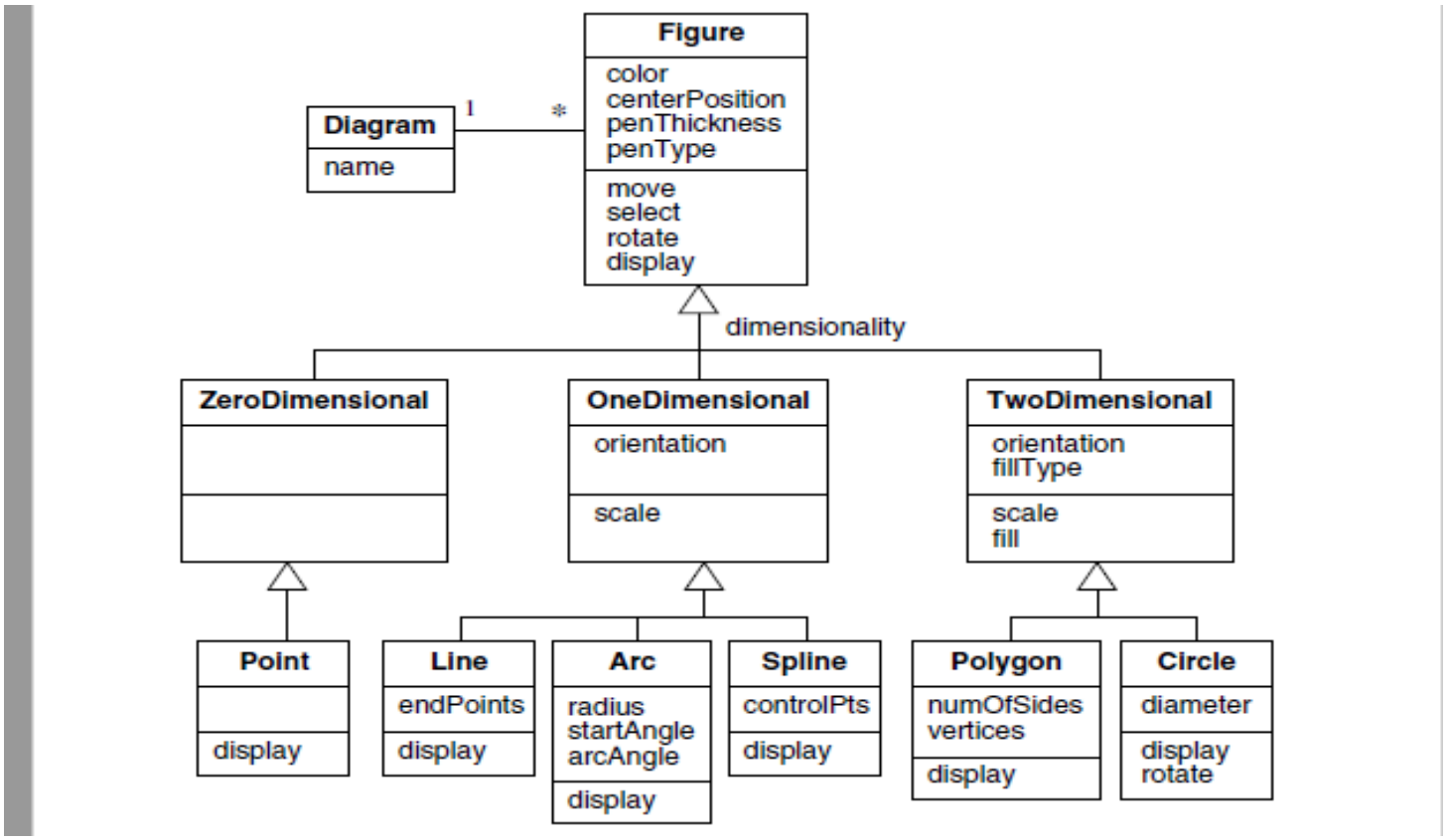
CLASS DIAGRAM

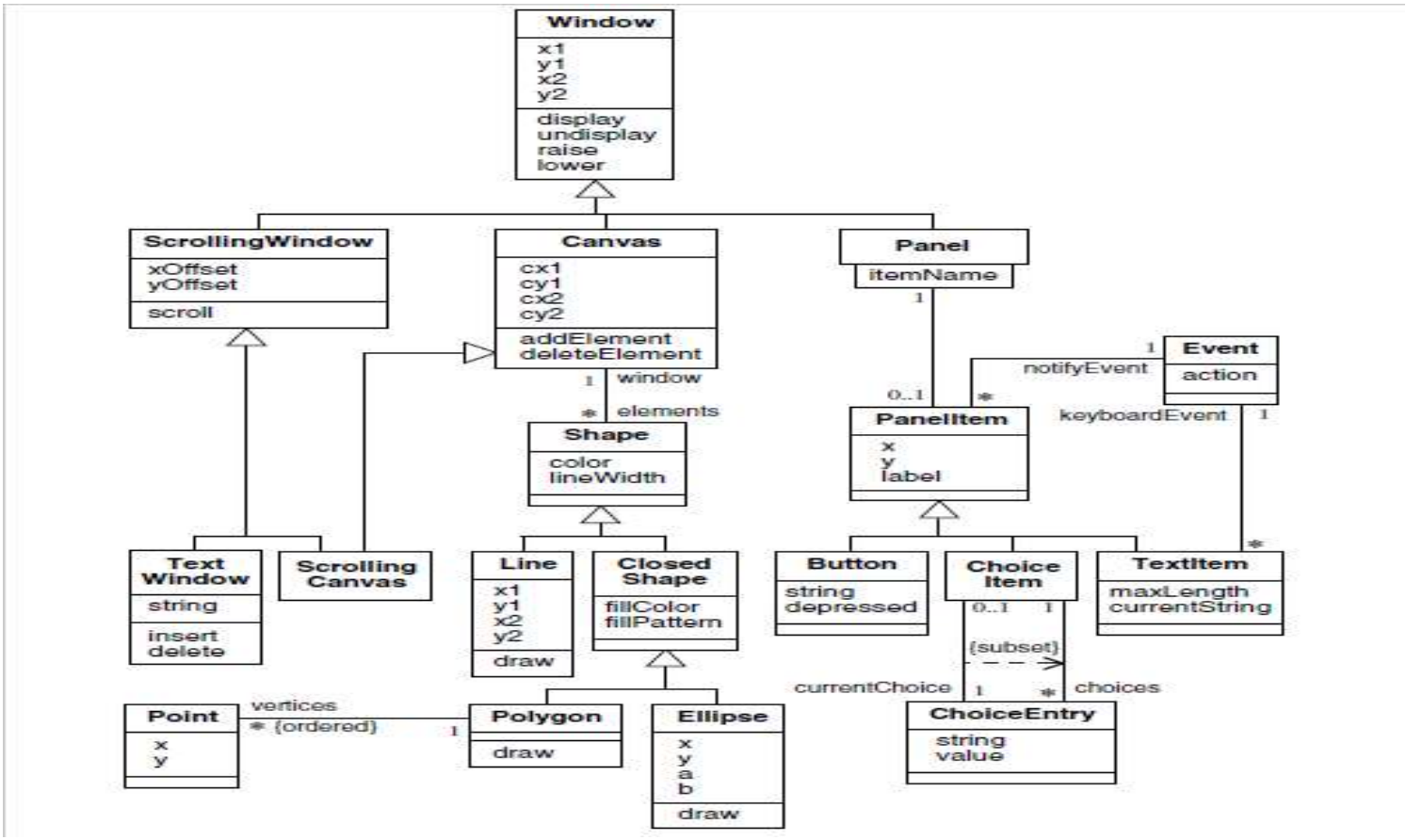
1.



2.



3.



4.

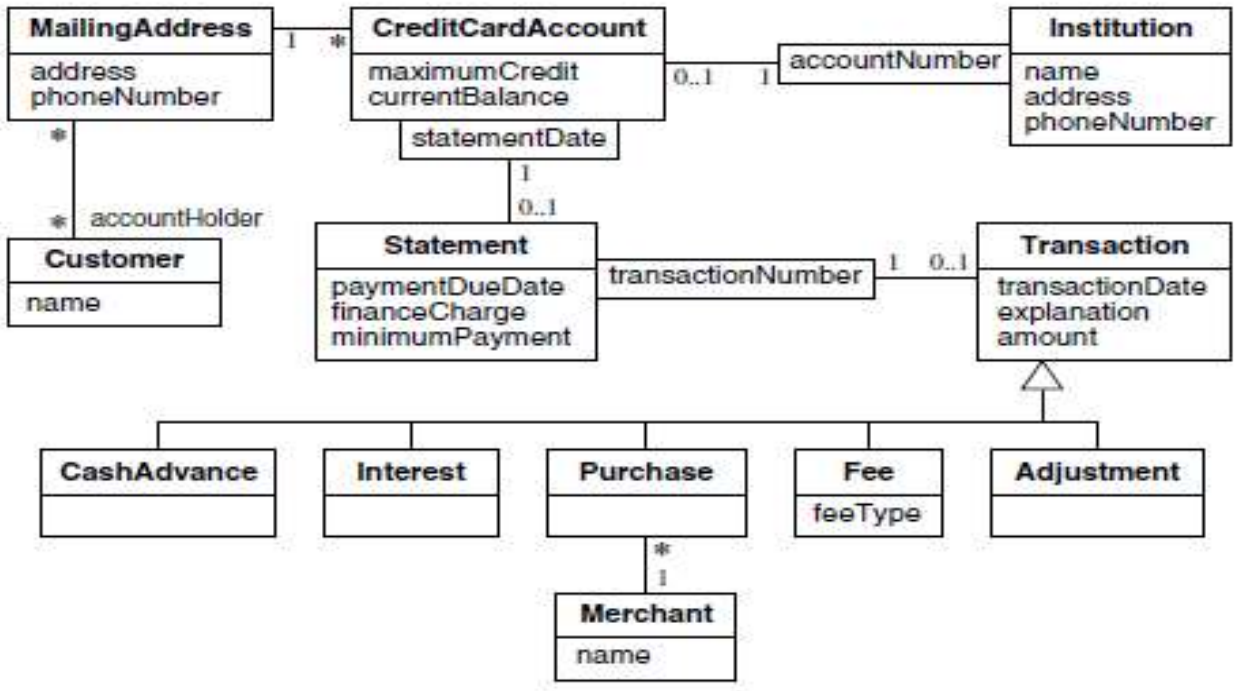


Figure 3.27 Class model for managing credit card accounts

STATE

1.State description

State: *AlarmRinging*

Description: alarm on watch is ringing to indicate target time

Event sequence that produces the state:

setAlarm (*targetTime*)
any sequence not including *clearAlarm*
when (*currentTime* = *targetTime*)

Condition that characterizes the state:

alarm = on, alarm set to *targetTime*, *targetTime* ≤ *currentTime* ≤ *targetTime* + 20 seconds, and no button has been pushed since *targetTime*

Events accepted in the state:

event	response	next state
when (*currentTime* = *targetTime* + 20)	*resetAlarm*	*normal*
buttonPushed (any button)	*resetAlarm*	*normal*

2.

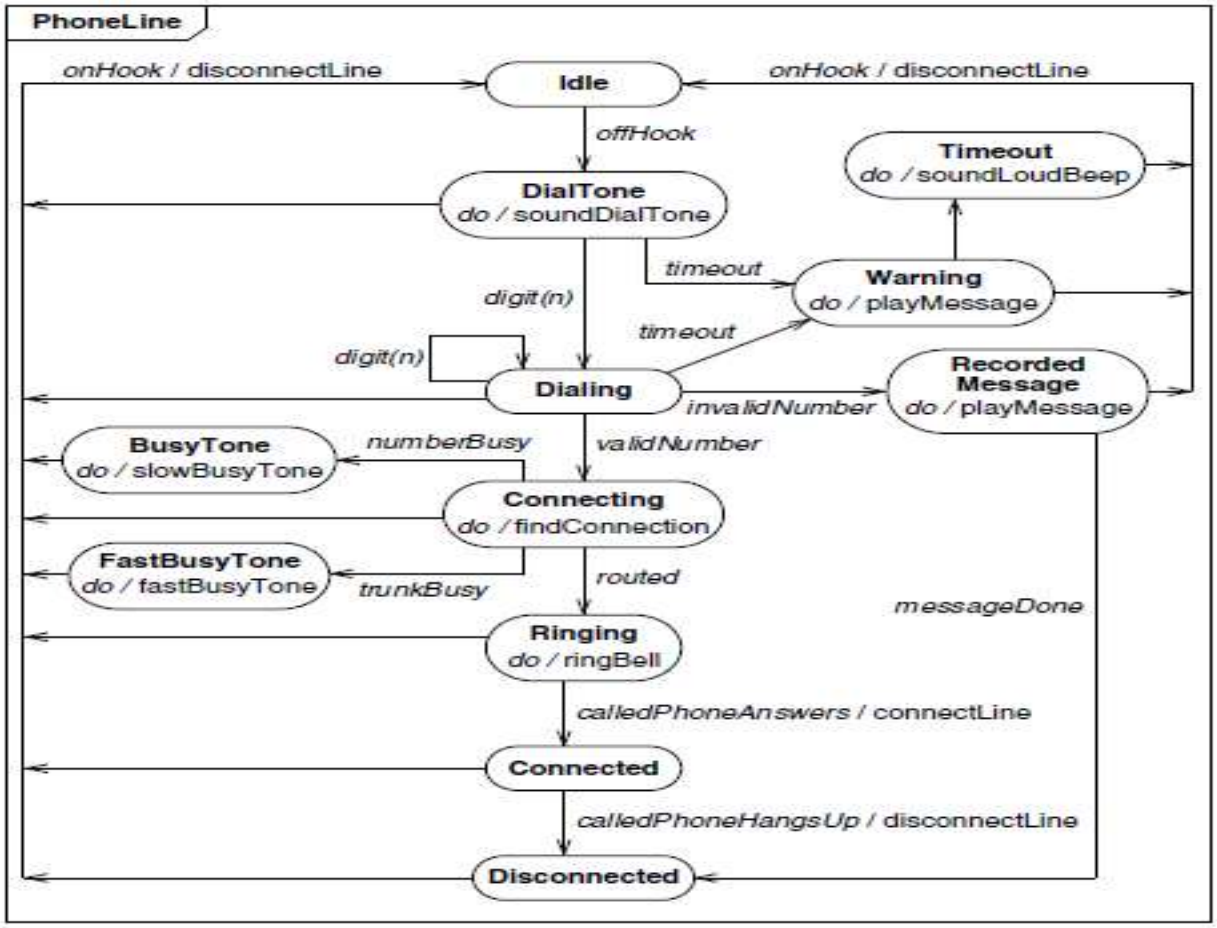


Figure 5.17 State diagram for phone line with activities. State diagrams let you express what objects do in response to events.

3.

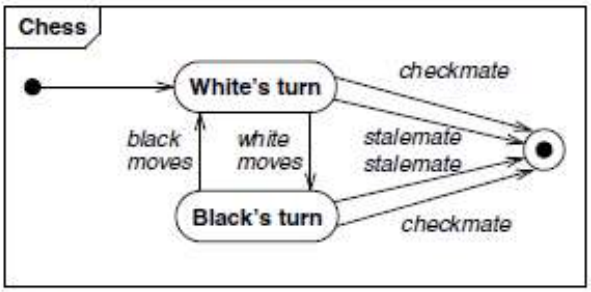


Figure 5.9 State diagram for chess game. One-shot diagrams represent objects with finite lives.

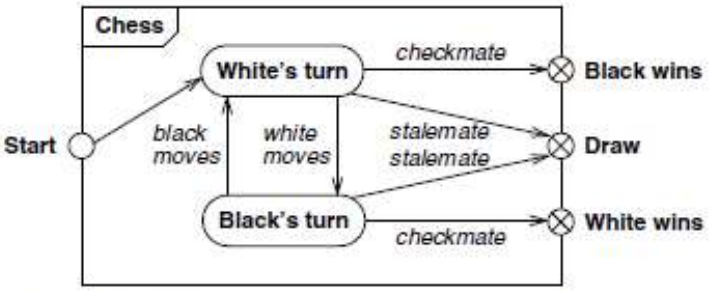


Figure 5.10 State diagram for chess game. You can also show one-shot diagrams by using entry and exit points.

4.

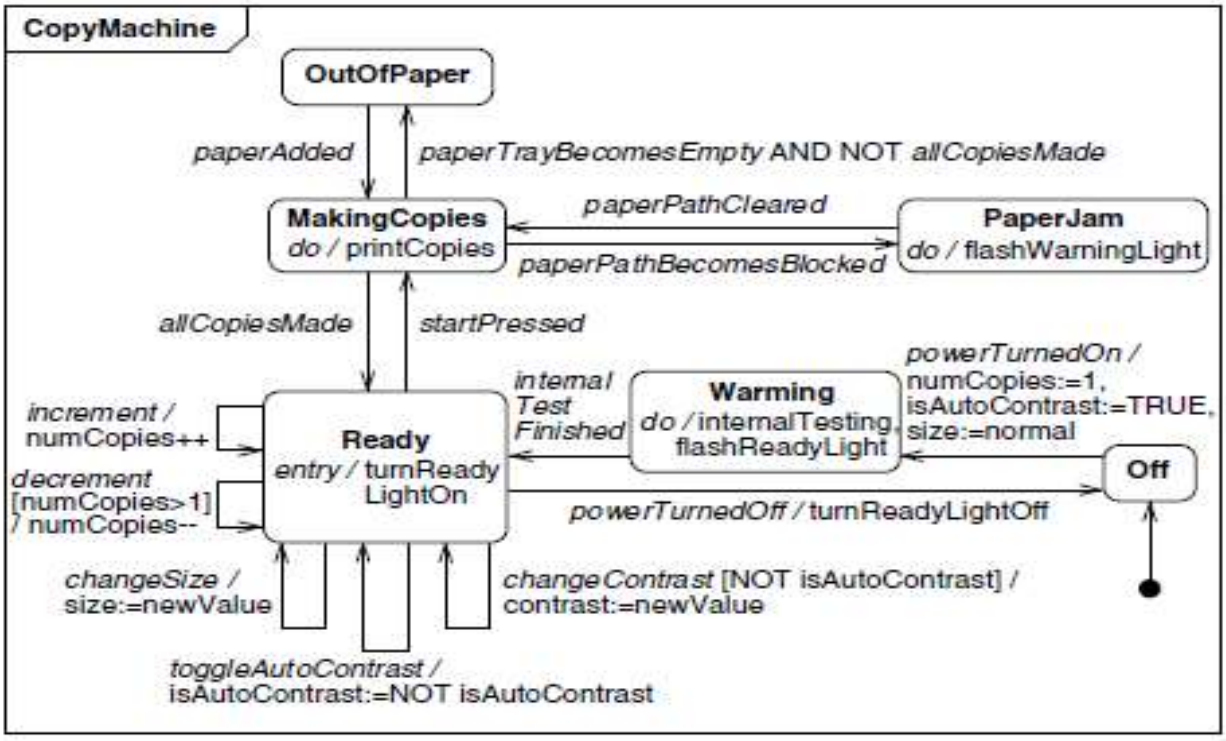


Figure E5.4 State diagram for a copy machine

5.

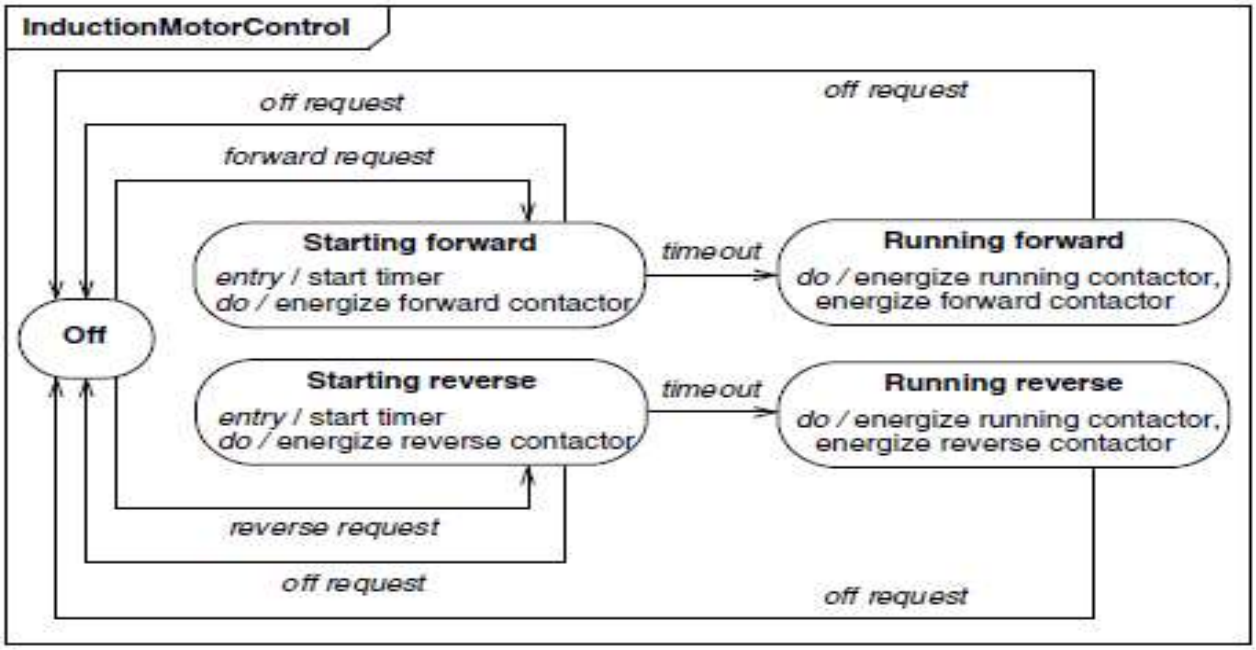


Figure E6.1 State diagram for an induction motor control

USE CASE

1.

- **Buy a beverage**. The vending machine delivers a beverage after a customer selects and pays for it.
- **Perform scheduled maintenance**. A repair technician performs the periodic service on the vending machine necessary to keep it in good working condition.
- **Make repairs**. A repair technician performs the unexpected service on the vending machine necessary to repair a problem in its operation.
- **Load items**. A stock clerk adds items into the vending machine to replenish its stock of beverages.

Figure 7.1 Use case summaries for a vending machine. A use case is a coherent piece of functionality that a system can provide by interacting with actors.

2.

Use Case: Buy a beverage

Summary: The vending machine delivers a beverage after a customer selects and pays for it.

Actors: Customer

Preconditions: The machine is waiting for money to be inserted.

Description: The machine starts in the waiting state in which it displays the message "Enter coins." A customer inserts coins into the machine. The machine displays the total value of money entered and lights up the buttons for the items that can be purchased for the money inserted. The customer pushes a button. The machine dispenses the corresponding item and makes change, if the cost of the item is less than the money inserted.

Exceptions:

Canceled: If the customer presses the cancel button before an item has been selected, the customer's money is returned and the machine resets to the waiting state.

Out of stock: If the customer presses a button for an out-of-stock item, the message "That item is out of stock" is displayed. The machine continues to accept coins or a selection.

Insufficient money: If the customer presses a button for an item that costs more than the money inserted, the message "You must insert *$nn.nn* more for that item" is displayed, where *nn.nn* is the amount of additional money needed. The machine continues to accept coins or a selection.

No change: If the customer has inserted enough money to buy the item but the machine cannot make the correct change, the message "Cannot make correct change" is displayed and the machine continues to accept coins or a selection.

Postconditions: The machine is waiting for money to be inserted.

Figure 7.2 Use case description. A use case brings together all of the behavior relevant to a slice of system functionality.

3.

7.1 Use Case Models

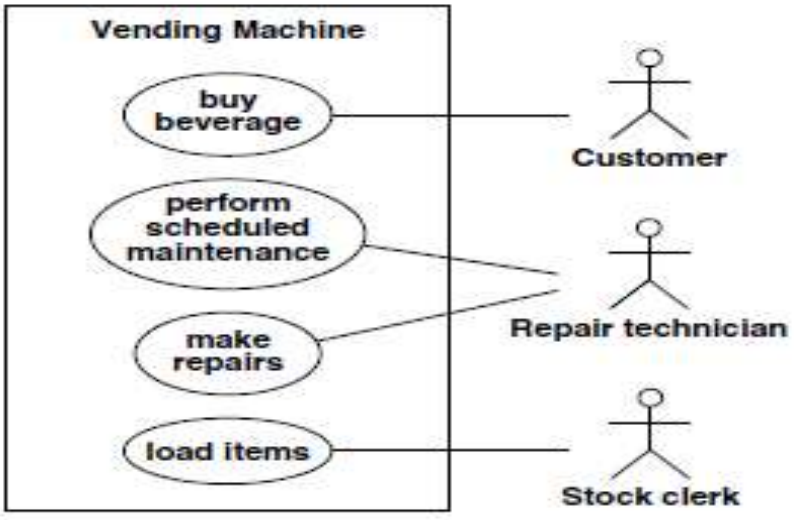


Figure 7.3 Use case diagram for a vending machine. A system involves a set of use cases and a set of actors.

SEQUENCE DIAGRAMS

1.

John Doe logs in.
System establishes secure communications.
System displays portfolio information.
John Doe enters a buy order for 100 shares of GE at the market price.
System verifies sufficient funds for purchase.
System displays confirmation screen with estimated cost.
John Doe confirms purchase.
System places order on securities exchange.
System displays transaction tracking number.
John Doe logs out.
System establishes insecure communication.
System displays good-bye screen.
Securities exchange reports results of trade.

Figure 7.4 Scenario for a session with an online stock broker. A scenario is a sequence of events that occurs during one particular execution of a system.

2.

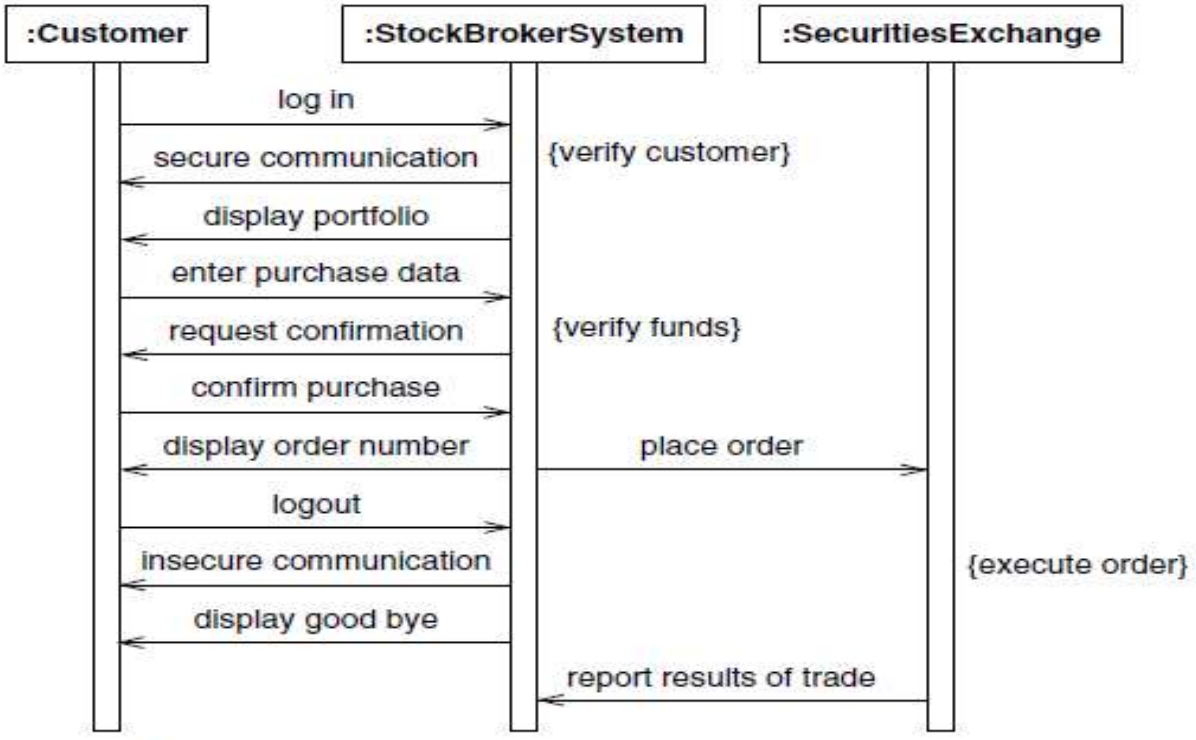


Figure 7.5 Sequence diagram for a session with an online stock broker. A sequence diagram shows the participants in an interaction and the sequence of messages among them.

3.

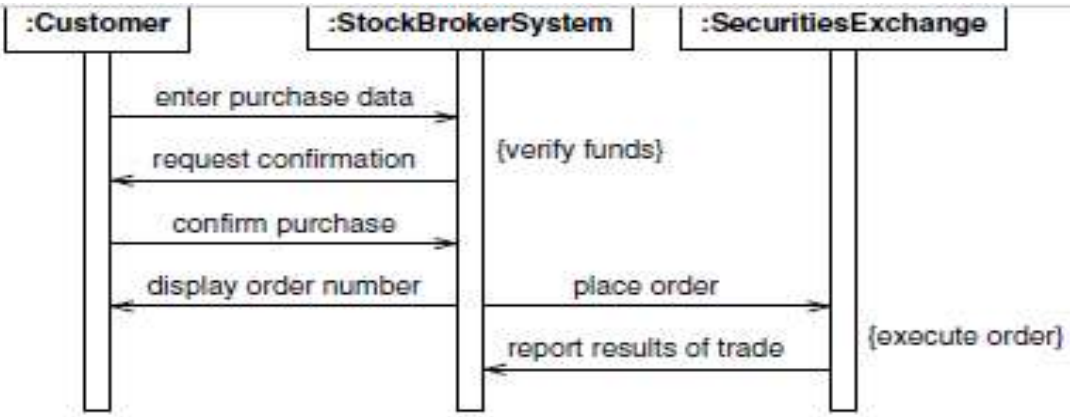


Figure 7.6 Sequence diagram for a stock purchase. Sequence diagrams can show large-scale interactions as well as smaller, constituent tasks.

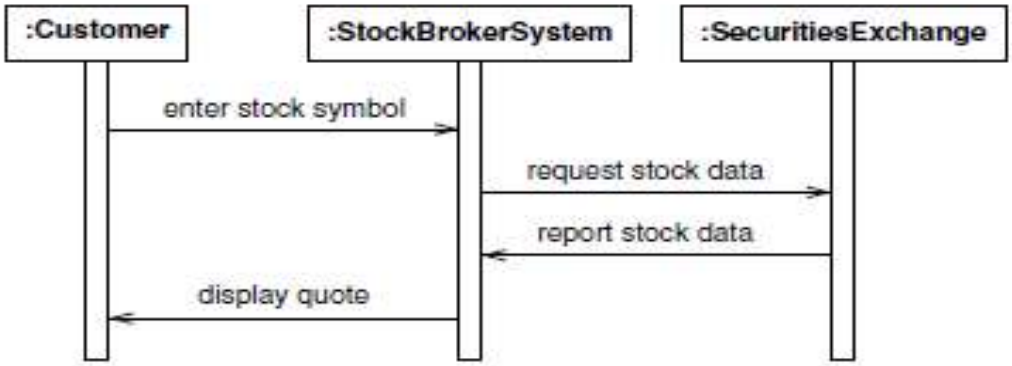


Figure 7.7 Sequence diagram for a stock quote

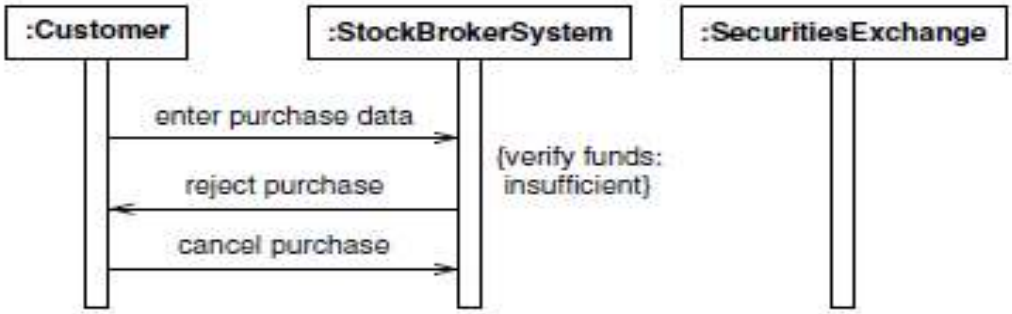


Figure 7.8 Sequence diagram for a stock purchase that fails

ACTIVITY DIAGRAMS

1.

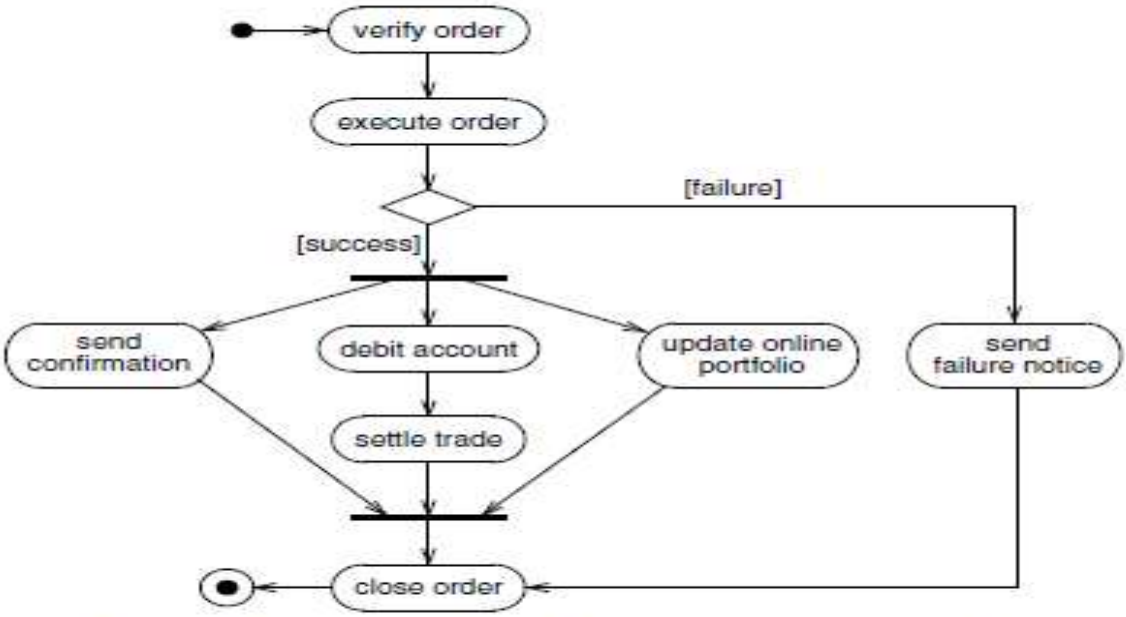


Figure 7.9 Activity diagram for stock trade processing. An activity diagram shows the sequence of steps that make up a complex process.

2.

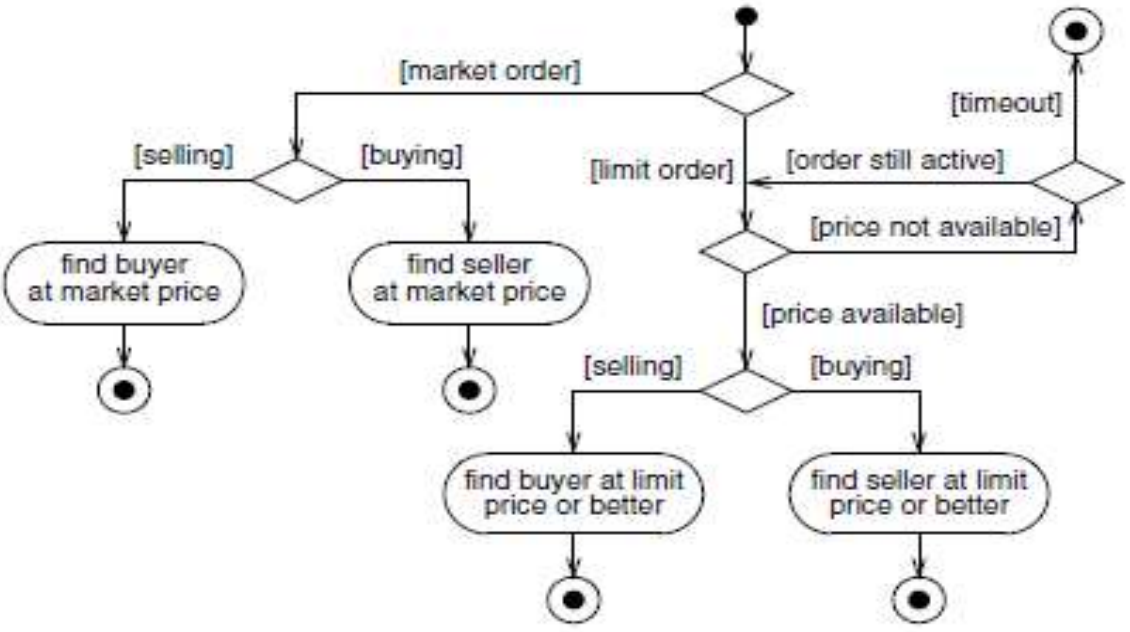


Figure 7.10 Activity diagram for *execute order*. An activity may be decomposed into finer activities.

ATM SYSTEM

Class diagram

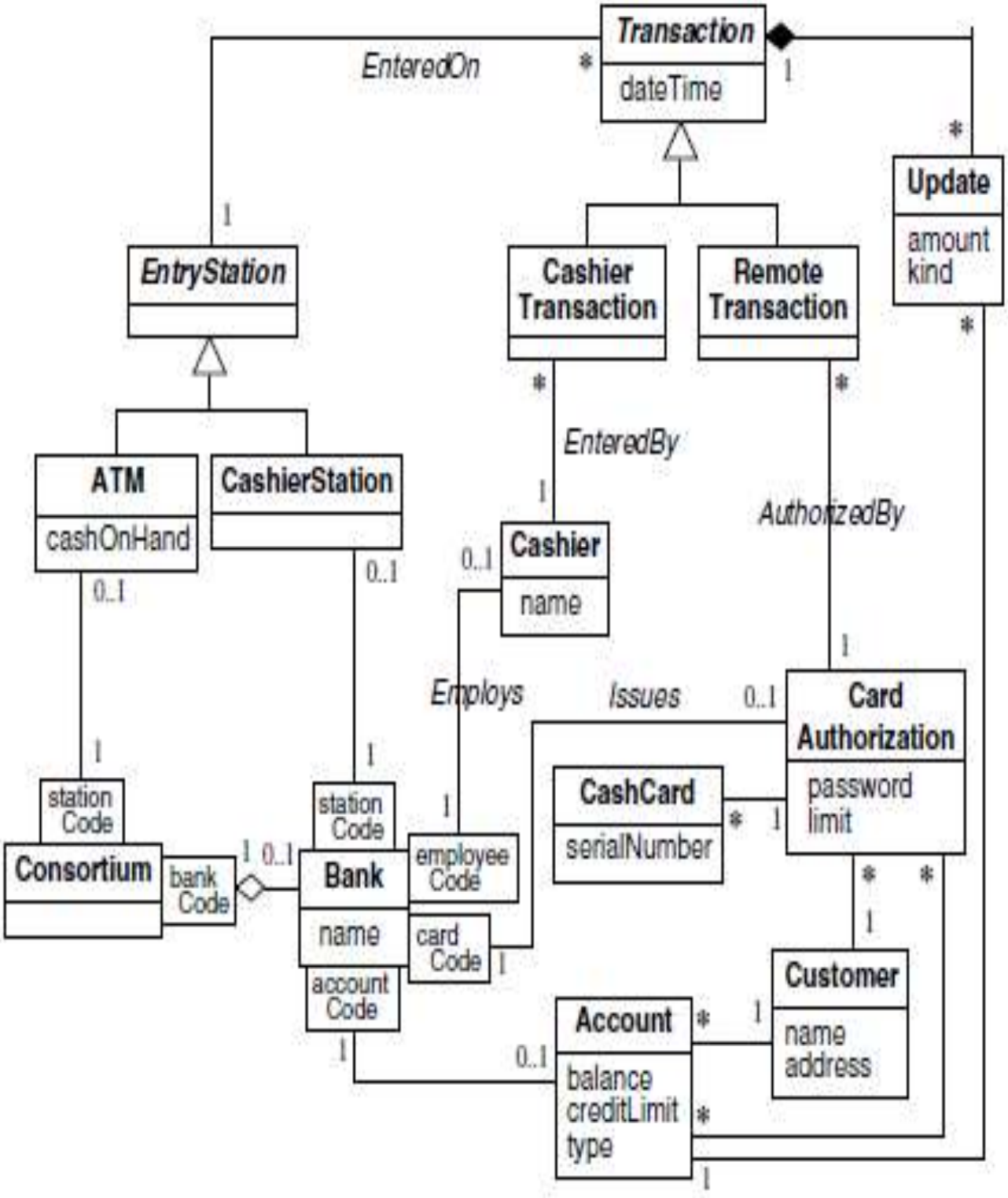


Figure 12.12 ATM class model after further revision

Use case

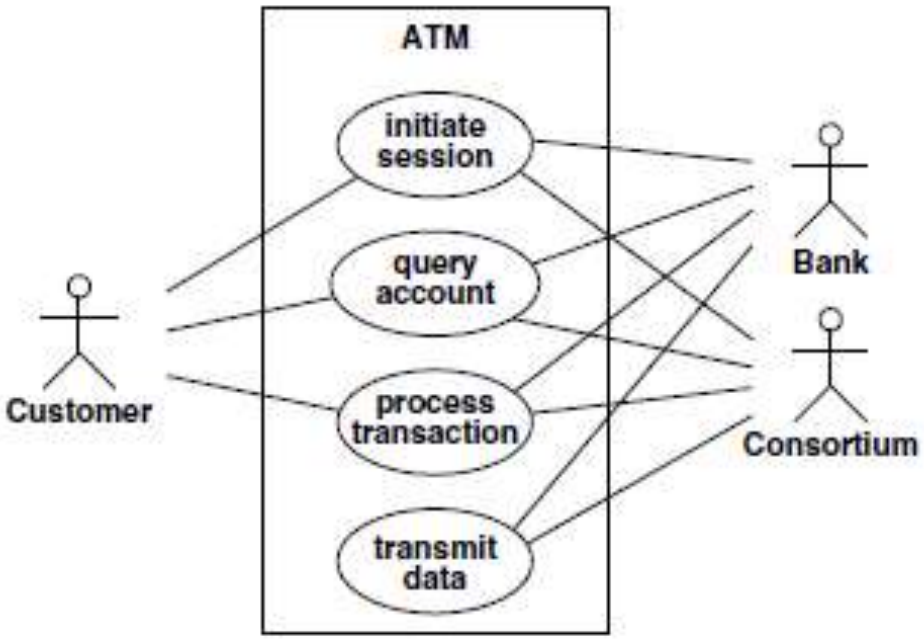


Figure 13.1 Use case diagram for the ATM. Use cases partition the functionality of a system into a small number of discrete units that cover its behavior.

Sequence diagram of transaction

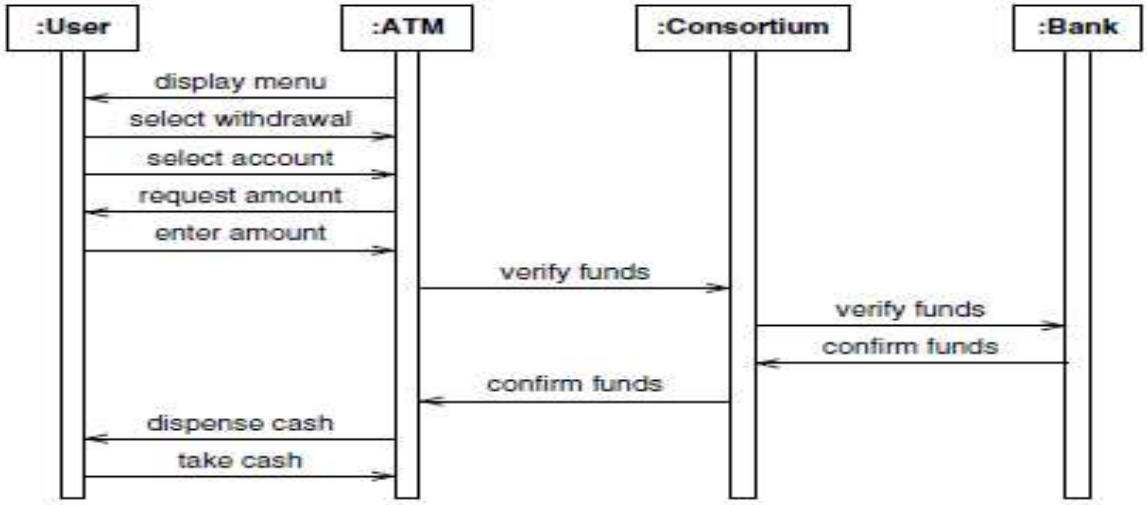


Figure 13.3 Sequence diagram for the *process transaction* scenario. A sequence diagram clearly shows the sender and receiver of each event.

Activity for card verification

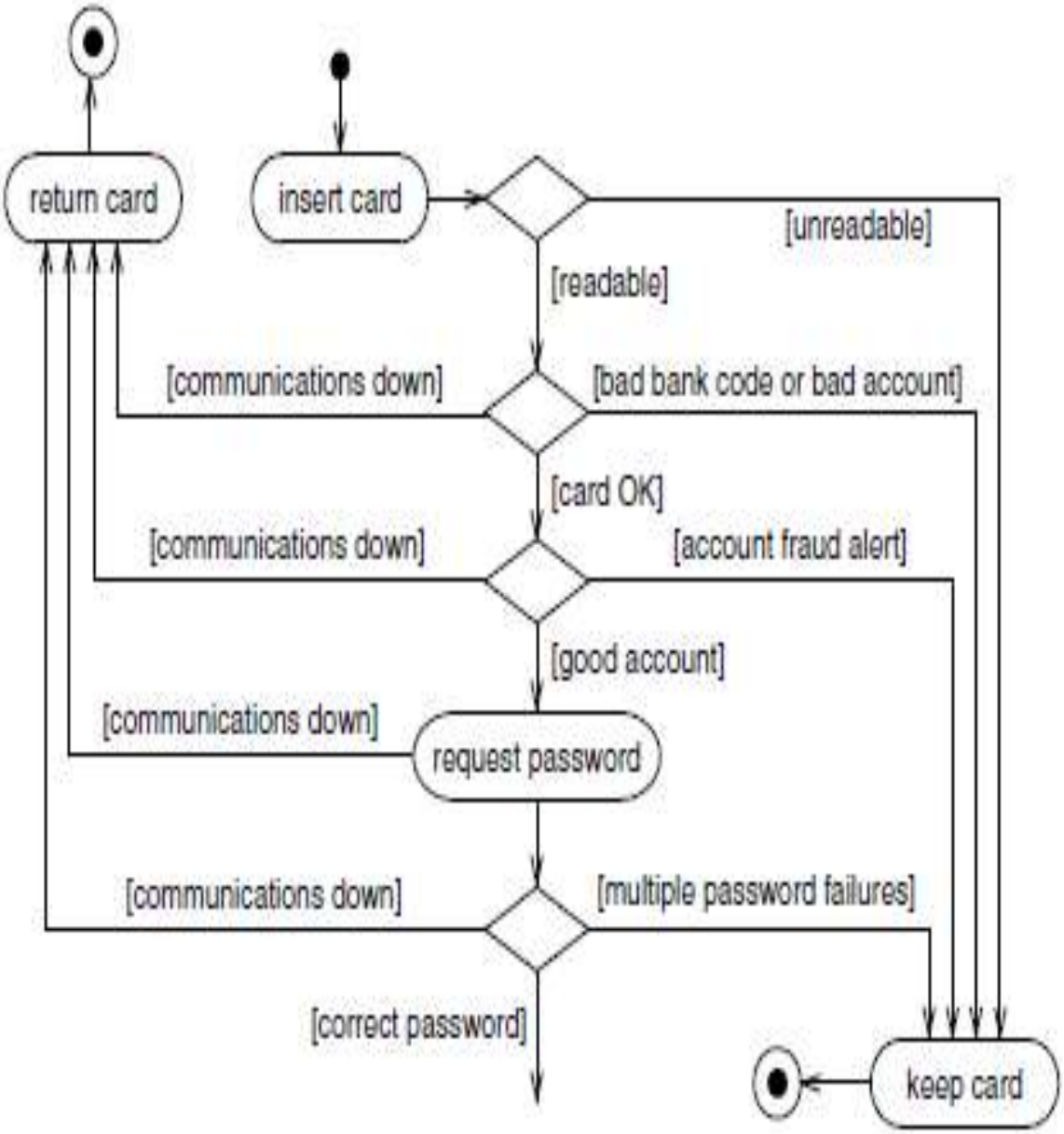


Figure 13.5 Activity diagram for card verification. You can use activity diagrams to document business logic, but do not use them as an excuse to begin premature implementation.

State machine diagrams

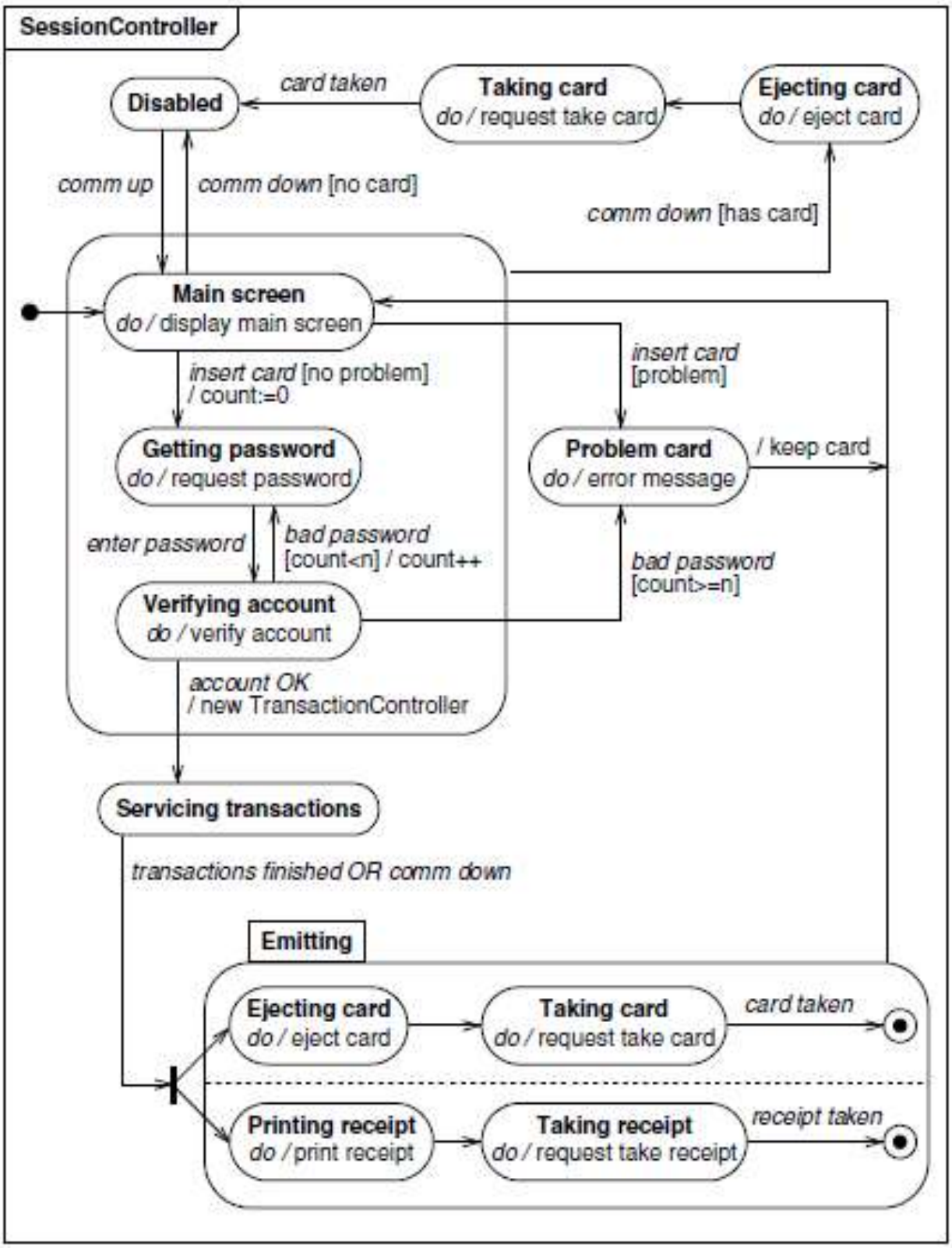


Figure 13.9 State diagram for *SessionController*. Build a state diagram for each application class with temporal behavior.

2.

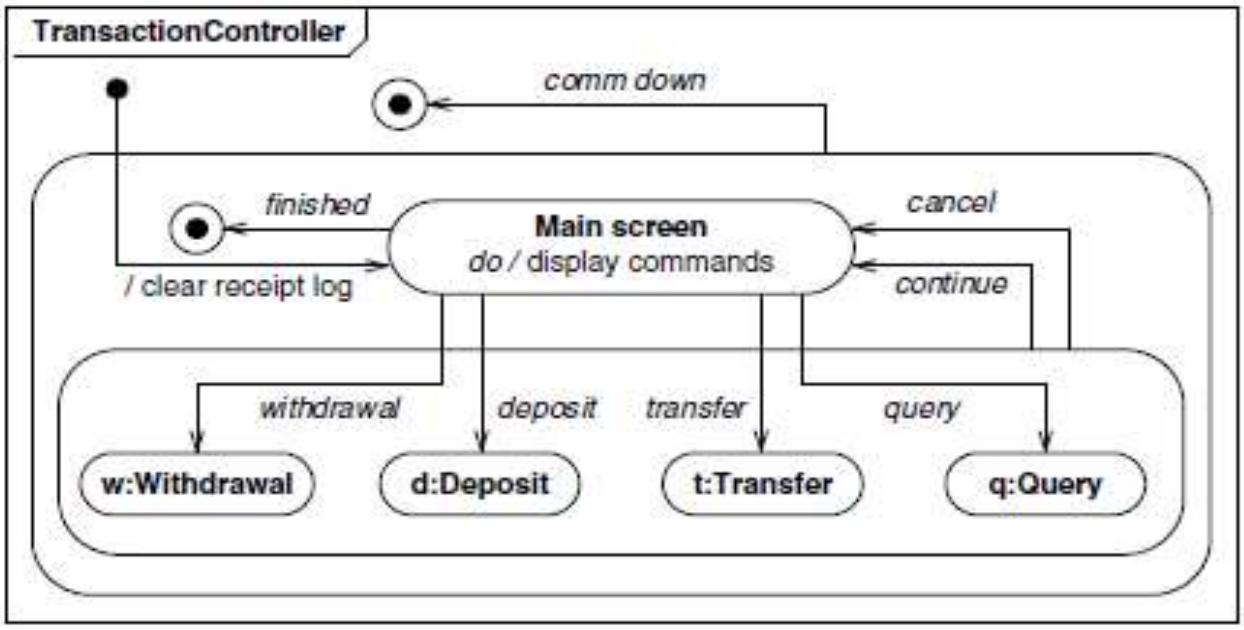


Figure 13.10 State diagram for *TransactionController*. Obtain information from the scenarios of the interaction model.

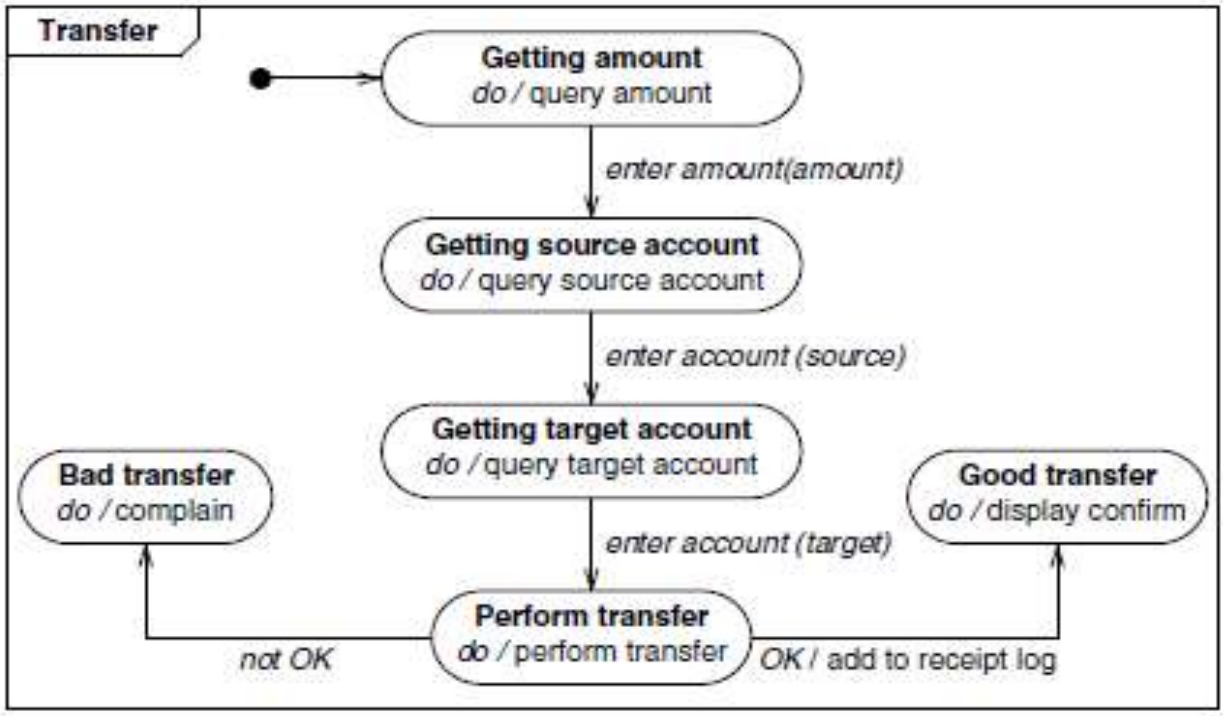


Figure 13.11 State diagram for *Transfer*. This diagram elaborates the *Transfer* state in Figure 13.10.

TABLES FOR SCHEMA OF ATM

Bank table

bankID	name (ck1)

Customer table

customerID	name	tempAmount

Account table

account ID	balance	credit Limit	bankID (ck1) (references Bank)	accountCode (ck1)	account Type	customerID (references Customer)

CardAuthorization table

card AuthorizationID	password	limit	bankID (ck1) (references Bank)	cardCode (ck1)	customerID (references Customer)

CashCard table

cashCardID	serialNumber	cardAuthorizationID (references CardAuthorization)

Account_CardAuthorization table

accountID (references Account)	**cardAuthorizationID** (references CardAuthorization)

SavingsAccount table

savingsAccountID (references Account)

CheckingAccount table

checkingAccountID (references Account)	protectFrom Overdraft

Address table

address ID	address	customerID (references Customer)

Figure 19.15 RDBMS tables for the abbreviated ATM model