Week 9

Lab Session Activities

Homework assignment #9

Team Homework Assignment #9

http://www.csun.edu/~twang/380/HW/HW9.pdf

1. Requirements of the ATM Simulator

The software to be designed will control a simulated automated teller machine (ATM) having a magnetic stripe reader for reading an ATM card, a customer console (keyboard and display) for interaction with the customer, a slot for depositing envelopes, a dispenser for cash (in multiples of $20), a printer for printing customer receipts, and a key-operated switch to allow an operator to start or stop the machine. The ATM will communicate with the bank's computer over an appropriate communication link. (The software on the latter is not part of the requirements for this problem.)

The ATM will service one customer at a time. A customer will be required to insert an ATM card and enter a personal identification number (PIN) - both of which will be sent to the bank for validation as part of each transaction. The customer will then be able to perform one or more transactions. The card will be retained in the machine until the customer indicates that he/she desires no further transactions, at which point it will be returned.

The ATM must be able to provide the following services to the customer:

1. A customer must be able to make a cash withdrawal from any suitable account linked to the card, in multiples of $30.00. Approval must be obtained from the bank before cash is dispensed.
2. A customer must be able to make a deposit to any account linked to the card, consisting of cash and/or checks in an envelope. The customer will enter the amount of the deposit into the ATM, subject to manual verification when the envelope is removed from the machine by an operator. Approval must be obtained from the bank before physically accepting the envelope.
3. A customer must be able to make a transfer of money between any two accounts linked to the card.
4. A customer must be able to make a balance inquiry of any account linked to the card. A customer must be able to abort a transaction in progress by pressing the Cancel key instead of responding to a request from the machine.

The ATM will communicate each transaction to the bank and obtain verification that it was allowed by the bank. Ordinarily, a transaction will be considered complete by the bank once it has been approved. In the case of a deposit, a second message will be sent to the bank indicating that the customer has deposited the envelope. (If the customer fails to deposit the envelope within the timeout period, or presses cancel instead, no second message will be sent to the bank and the deposit will not be credited to the customer.)

If the bank determines that the customer's PIN is invalid, the customer will be required to re-enter the PIN before a transaction can proceed. If the customer is unable to successfully enter the PIN after a certain number of tries, the card will be permanently retained by the machine, and the customer will have to contact the bank to get it back.

If a transaction fails for any reason other than an invalid PIN, the ATM will display an explanation of the problem, and will then ask the customer whether he/she wants to do another transaction. The ATM will provide the customer with a printed receipt for each successful transaction, showing the date, time, machine location, type of transaction, account(s), amount, and ending and available balance (s) of the affected account ("to" account for transfers).

The ATM will have a key-operated switch that will allow an operator to start and stop the service of the ATM System. After turning the switch to the "on" position, the operator will be required to verify and enter the total cash on hand. The machine can only be turned off when it is not servicing a customer. When the switch is moved to the "off" position, the machine will shut down, so that the operator may remove deposit envelopes and reload the machine with cash, blank receipts, etc.

The ATM will also maintain an internal log of transactions to facilitate resolving ambiguities arising from a hardware failure in the middle of a transaction. Entries will be made in the log when the ATM is started up and shut down, for each message sent to the bank (along with the response back, if one is expected), for the

dispensing of cash, and for the receiving of an envelope. Log entries may contain card numbers and dollar amounts, but for security will never contain a PIN.

Functional Requirements

1. System Startup
The system is started up when the operator turns the operator switch to the "on" position. The operator will be asked to enter the amount of money currently in the cash dispenser, and a connection to the bank will be established. Then the servicing of customers can begin.

2. System Shutdown
The system is shut down when the operator makes sure that no customer is using the machine, and then turns the operator switch to the "off" position. The connection to the bank will be shut down. Then the operator is free to remove deposited envelopes, replenish cash and paper, etc.

3. Session
A session is started when a customer inserts an ATM card into the card reader slot of the machine. The ATM pulls the card into the machine and reads it (If the reader cannot read the card due to improper insertion or a damaged stripe, the card is ejected, an error screen is displayed, and the session is aborted). If the machine is able to read the card, then the customer is asked to enter his/her PIN which is sent to the bank (If the bank reports that the customer's PIN is invalid, the Invalid PIN extension will be performed, in which an attempt will be made to continue the session, where the customer will be asked to re-enter the PIN number. This is done 3 times. If the customer enters invalid PIN entries repeatedly, the session is aborted, the customer will not be offered the option of starting another session, and the card will be retained in the machine). If approved, the customer is then allowed to perform one or more transactions, choosing from a menu of possible types of transaction in each case (See Transaction). After the success/failure of each transaction, the customer is asked whether he/she would like to perform another. When the customer is through performing transactions, the card is ejected from the machine and the session ends. The customer may abort the session by pressing the Cancel key when entering a PIN or choosing a transaction type.

3. Transaction
Transaction is an abstract generalization. Each specific concrete type of transaction implements certain operations in the appropriate way. The flow of events given here describes the behavior common to all types of transaction (withdrawal, deposit, transfer, inquiry). The flow of events for the individual types of transactions gives the features that are specific to that type of transaction. A transaction use case is started within a session after the PIN number has been approved by the bank, and the customer chooses a transaction type from a menu of options. The customer will be asked to furnish appropriate details (e.g. account(s) involved, amount). The transaction will then be sent to the bank, along with information from the customer's card and the PIN the customer entered. If the bank approves the transaction, any steps needed to complete the transaction (e.g. dispensing cash or accepting an envelope, etc) will be performed, and then a receipt will be printed. Then the customer will be asked whether he/she wishes to do another transaction.

The customer may cancel a transaction by pressing the Cancel key as described for each individual type of transaction below. If a transaction is cancelled by the customer, or fails for any reason other than repeated entries of an invalid PIN (e.g. insufficient funds, etc), a screen will be displayed informing the customer of the reason for the failure of the transaction, and then the customer will be offered the opportunity to do another. The messages to the bank and responses back are recorded in the ATM's log.

5. Withdrawal Transaction
A withdrawal transaction asks the customer to choose a type of account to withdraw from (e.g. checking) from a menu of possible accounts, and to choose a dollar amount from a menu of possible amounts. The system verifies that it has sufficient money on hand to satisfy the request before sending the transaction to the bank. (If not, the customer is informed and asked to enter a different amount.) If the transaction is approved by the bank, the

appropriate amount of cash is dispensed by the machine before it issues a receipt. (The dispensing of cash is also recorded in the ATM's log.) A withdrawal transaction can be cancelled by the customer pressing the Cancel key any time prior to choosing the dollar amount.

6. Deposit Transaction
A deposit transaction asks the customer to choose a type of account to deposit to (e.g. checking) from a menu of possible accounts, and to type in a dollar amount on the keyboard. The transaction is initially sent to the bank to verify that the ATM can accept a deposit from this customer to this account. If the transaction is approved, the machine accepts an envelope from the customer containing cash and/or checks before it issues a receipt. Once the envelope has been received, a second message is sent to the bank, to confirm that the bank can credit the customer's account - contingent on manual verification of the deposit envelope contents by an operator later. (The receipt of an envelope is also recorded in the ATM's log.) A deposit transaction can be cancelled by the customer pressing the Cancel key any time prior to inserting the envelope containing the deposit. The transaction is automatically cancelled if the customer fails to insert the envelope containing the deposit within a reasonable period of time after being asked to do so.

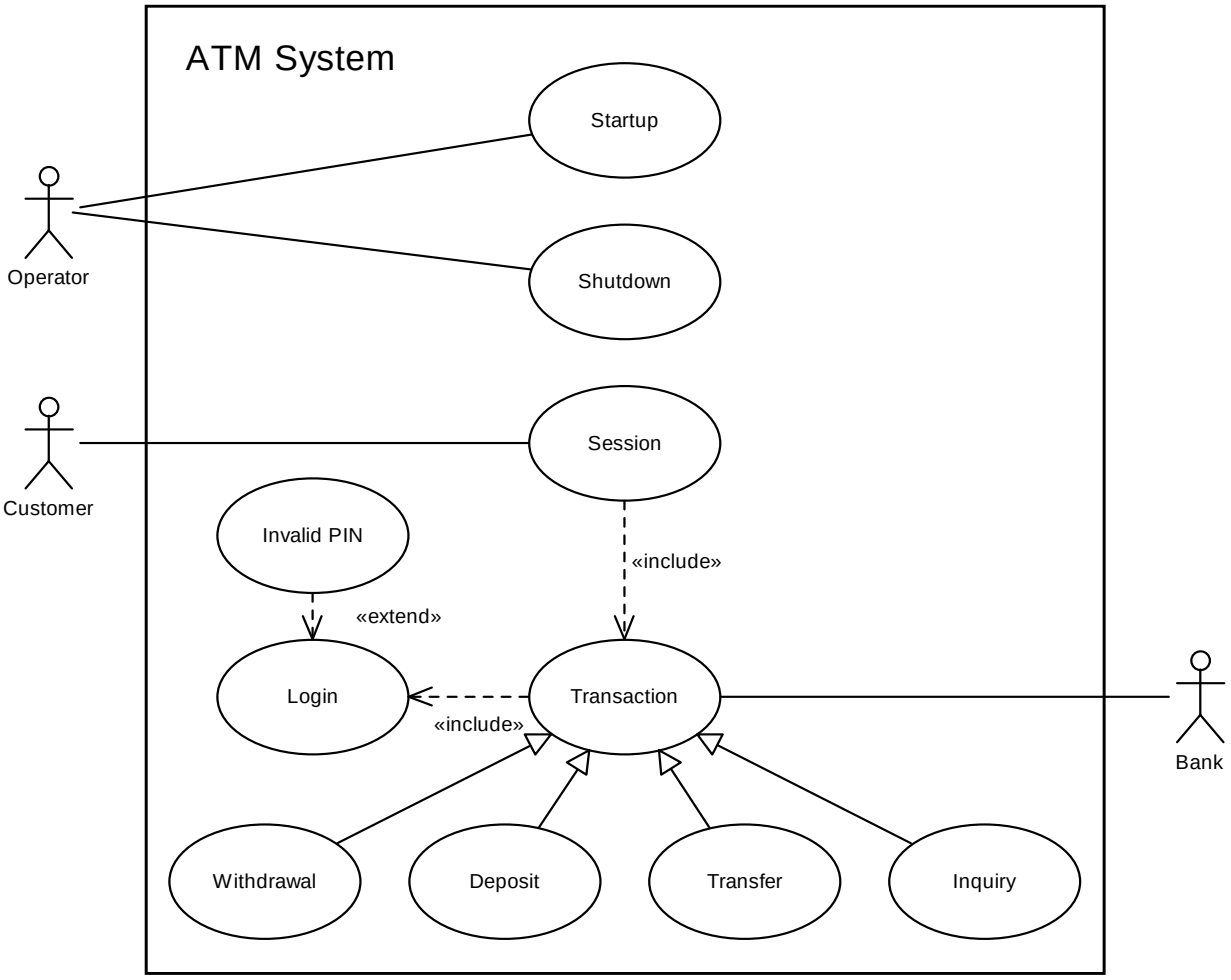
7. Transfer Transaction
A transfer transaction asks the customer to choose a type of account to transfer from (e.g. checking) from a menu of possible accounts, to choose a different account to transfer to, and to type in a dollar amount on the keyboard. No further action is required once the transaction is approved by the bank before printing the receipt. A transfer transaction can be cancelled by the customer pressing the Cancel key any time prior to entering a dollar amount.

8. Inquiry Transaction
An inquiry transaction asks the customer to choose a type of account to inquire about from a menu of possible accounts. No further action is required once the transaction is approved by the bank before printing the receipt. An inquiry transaction can be cancelled by the customer pressing the Cancel key any time prior to choosing the account to inquire about.

9. Invalid PIN Extension
An invalid PIN extension is started from within a transaction when the bank reports that the customer's PIN number is invalid. The customer will be asked to re-enter the PIN number, and the request is sent to the bank again. If the customer fails four times to enter the correct PIN, the card will be retained in the machine, a screen is displayed informing the customer of this and suggesting he/she contact the bank, and the entire customer session is aborted. If the customer presses Cancel instead of re-entering a PIN, the original transaction is cancelled. On the other hand, if the PIN is successfully re-entered, it is used for both the current transaction and all subsequent transactions in the session.

2. Use Case

2.1 Use Case Diagram



2.1 Use Case Descriptions

UC1: Startup

Characteristic Information

Goal	Power-up and initialize the ATM
Pre-Condition	ATM must be in the OFF mode
Success End Condition	ATM is powered up and has been initialized
Primary Actor	Operator

Primary Scenario

Step	Actor/System	Action Description

1	User	Push the power on button
2	ATM	Perform a self-test
3	ATM	Set the ATM in IDLE mode
4	ATM	Run the clock

Alternative Scenario

Step	Condition	Action Description
2a	self-test fails	Set an alarm and notify the operator to correct the problem
3a	Mode setting failure	Set an alarm and notify the operator to correct the problem
4a	Clock failure	Set an alarm and notify the operator to correct the problem

UC3: Session

Characteristic Information

Goal	Verify an ATM card before transactions start
Pre-Condition	ATM is powered up and has been initialized
Success End Condition	The card is ejected from the machine and the session ends
Primary Actor	User

Primary Scenario

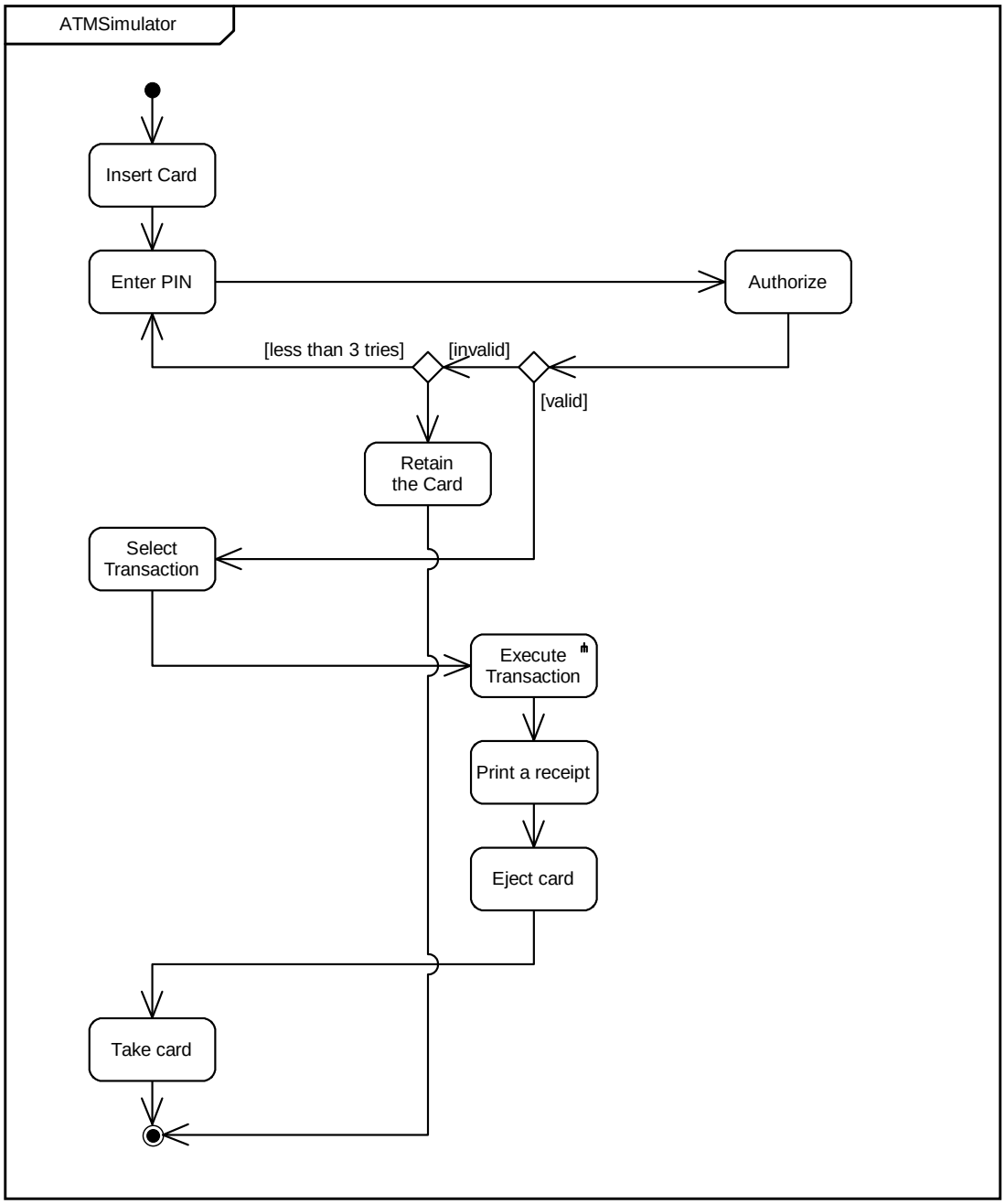
Step	Actor/System	Action Description
1	User	Insert an ATM card into the card reader slot of the machine
2	ATM	Pulls the card into the machine and reads it
3	ATM	Ask the user to enter his/her PIN
4	User	Punch the PIN
5	ATM	Verify the PIN and allow the user to perform one or more transactions

Alternative Scenario

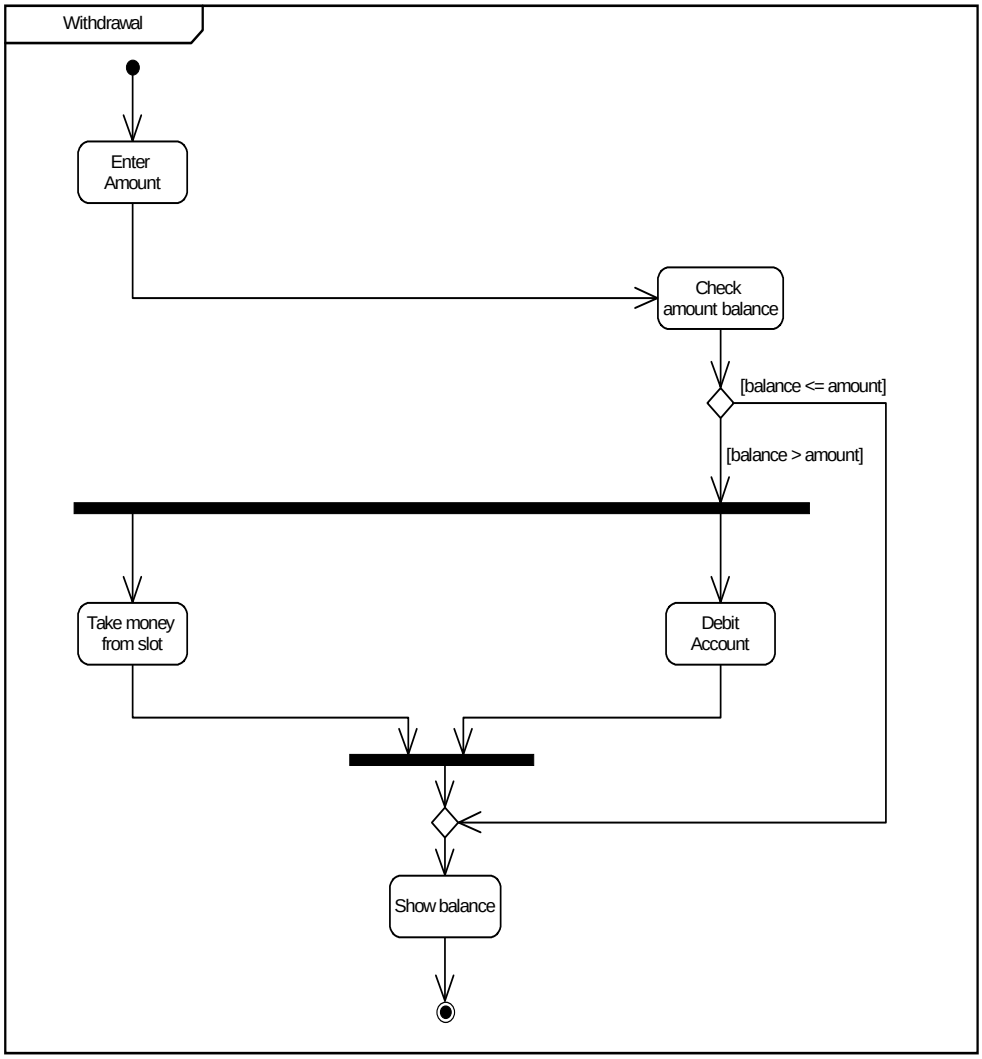
Step	Condition	Action Description
2a	Card reading fails	The session is aborted
4a	Abort the session	The user aborts the session by pressing the Cancel key
5a	Verification failed	The PIN is not valid. Many invalid PIN entries cause the session to be aborted, with the card being retained in the machine.

3. Activity Diagram

3.1 System Activity Diagram

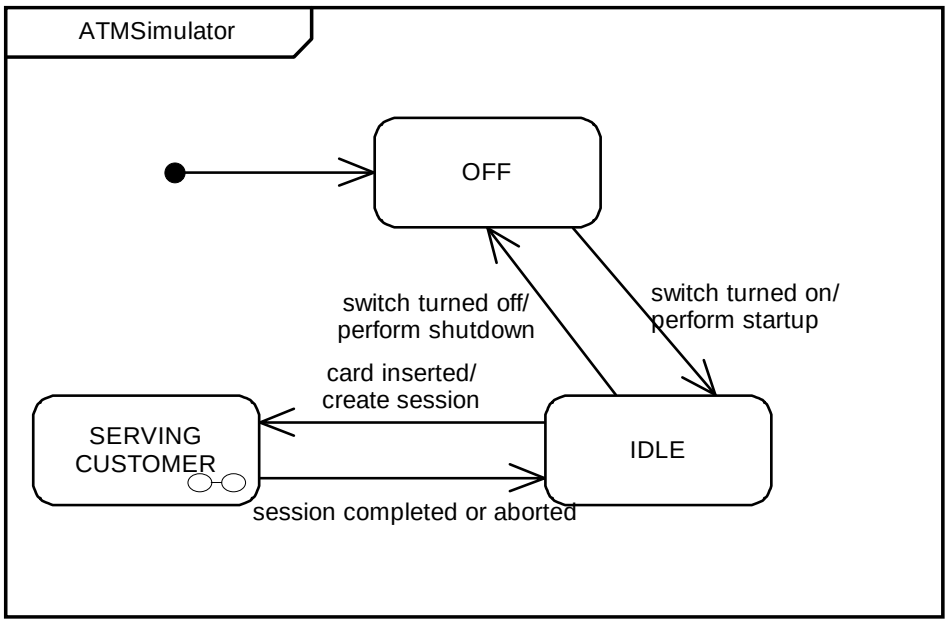


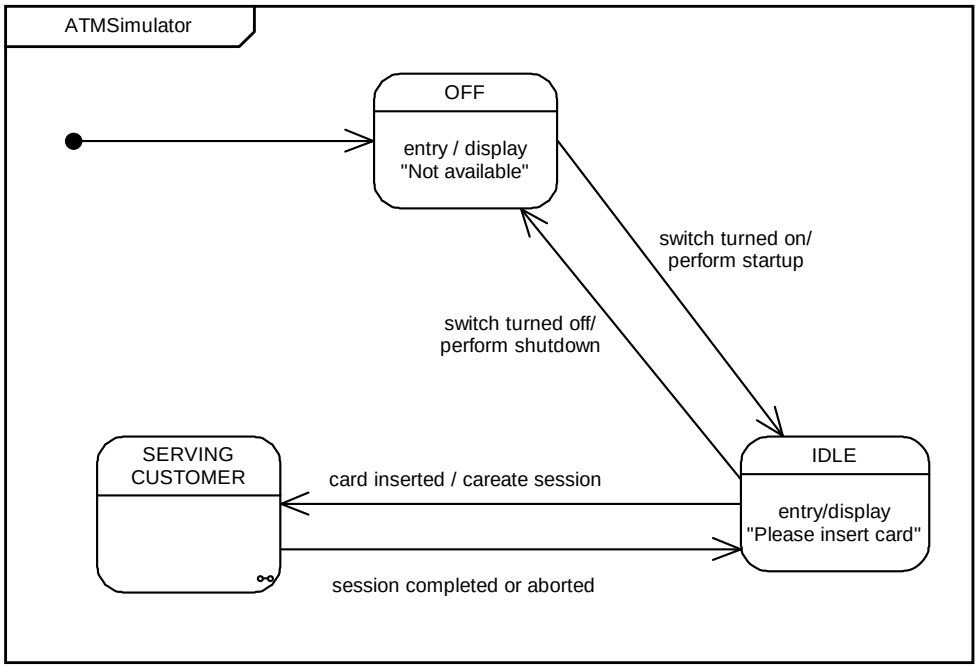
3. 2 Function Activity Diagram



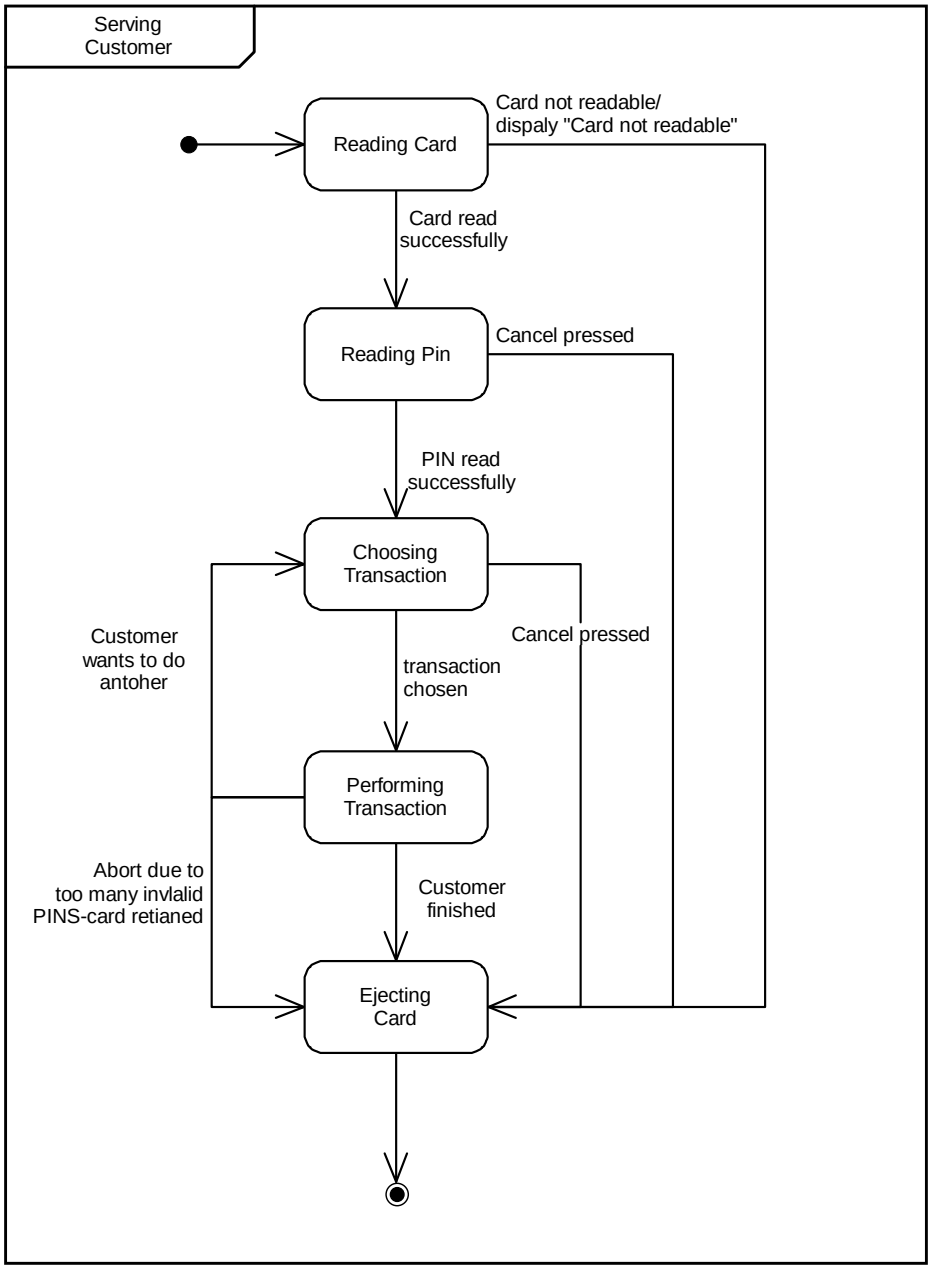
4. Statechart Diagram

4.1 System Statechart Diagram

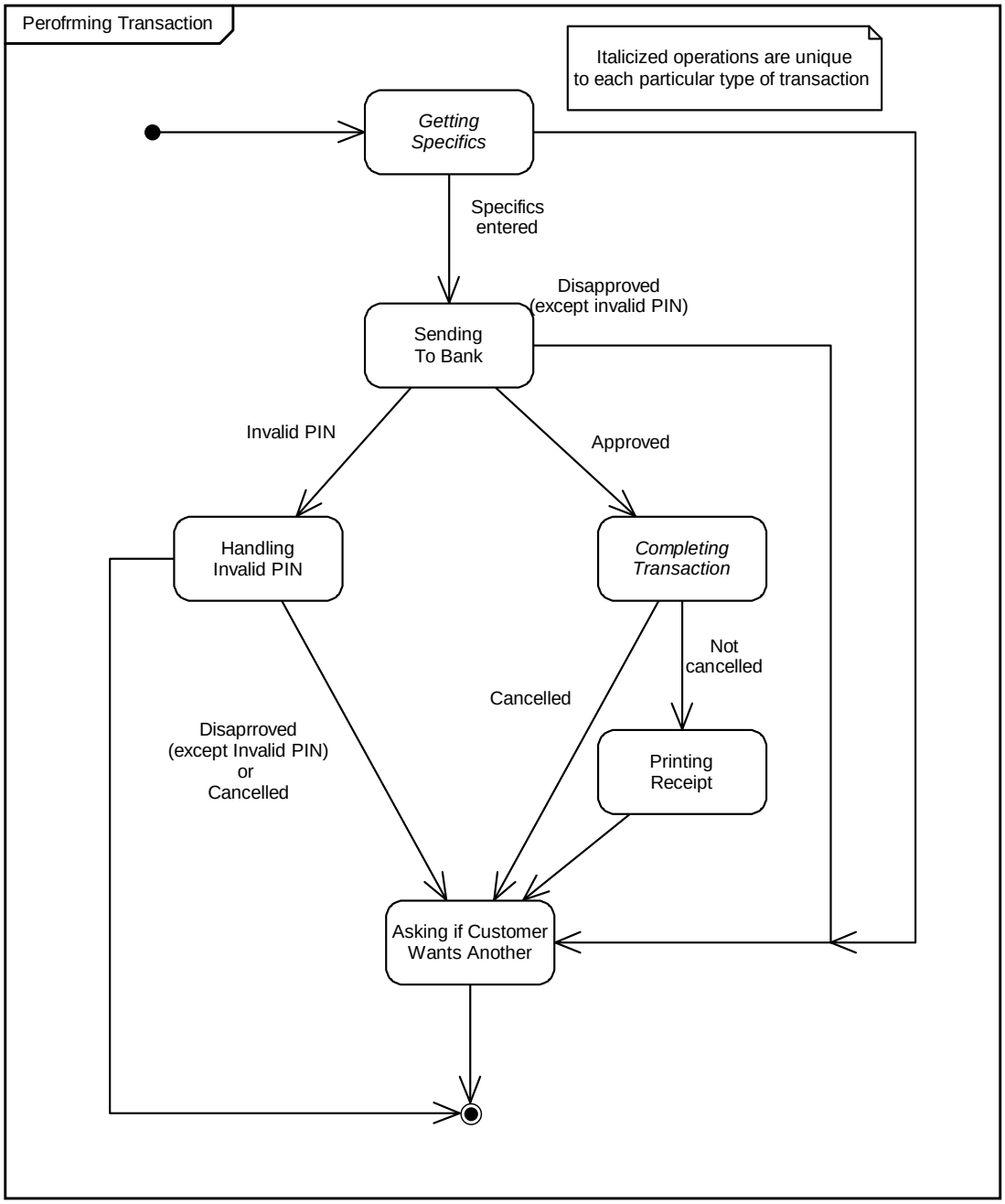




4.2 Substates of “Serving Customer” State



4.3 Substates of “Performing Transaction” State



5. CRC (Class Responsibilities Collaborators) Cards

ATM	
Responsibilities	**Collaborators**
Start up when switch is turned on	OperatorPanel CashDispenser NetworkToBank
Shut down when switch is turned off	NetworkToBank
Start a new session when card is inserted by customer	Session CustomerConsole
Provide access to component parts for sessions and transactions	

CardReader	
Responsibilities	**Collaborators**
Tell ATM when card is inserted ATM	ATM Controller
Read information from card Card	Card
Eject card	
Retain card	

Session	
Responsibilities	**Collaborators**
Perform session use case ATM	CardReader Card CustomerConsole Transaction

Transaction	
Responsibilities	**Collaborators**
Allow customer to choose a type of transaction	CustomerConsole Withdrawal Deposit Transfer Inquiry

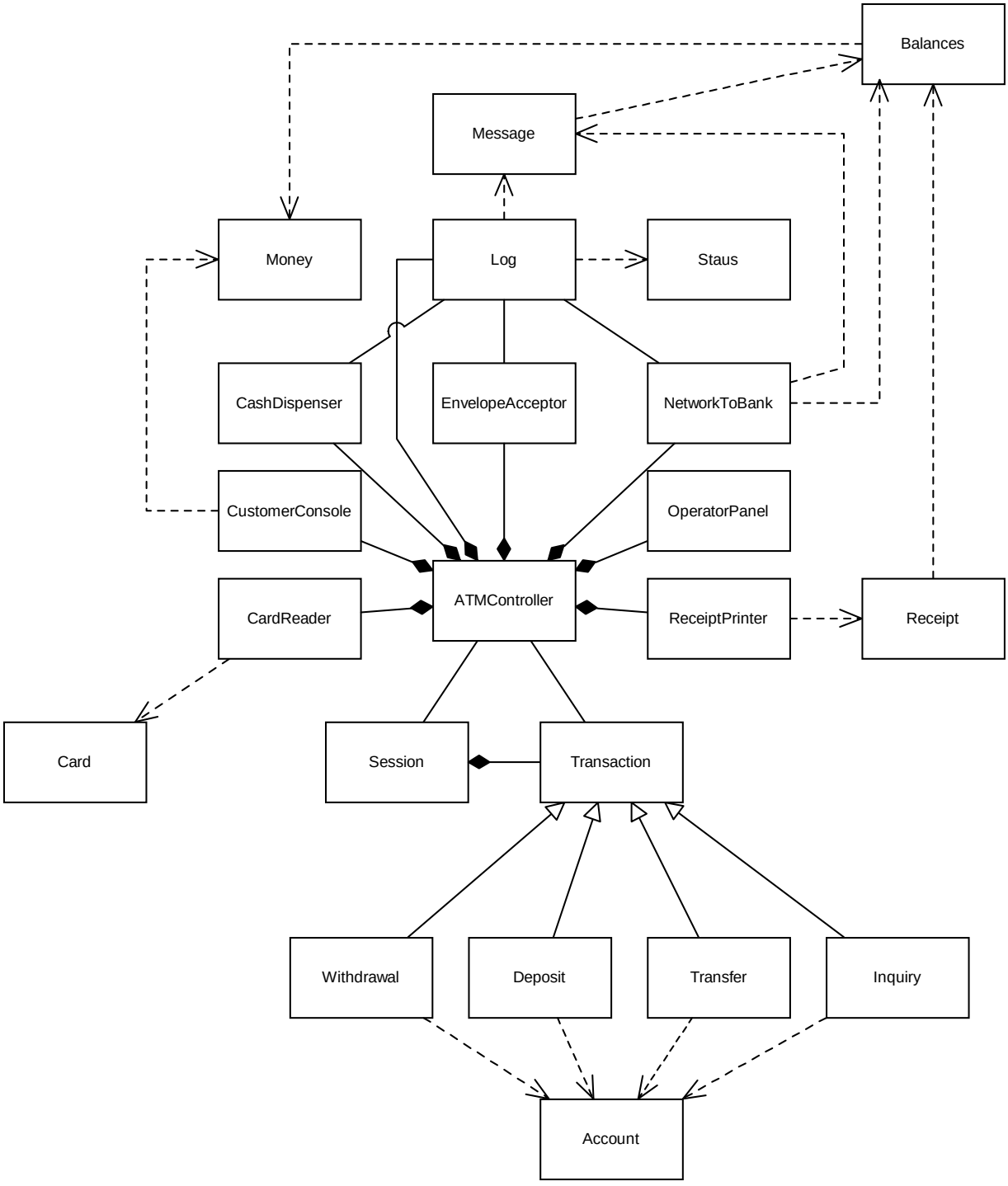
NetworkToBank	
Responsibilities	**Collaborators**
Initiate connection to bank at startup Send message to bank and wait for response	 Message Log Balances Status

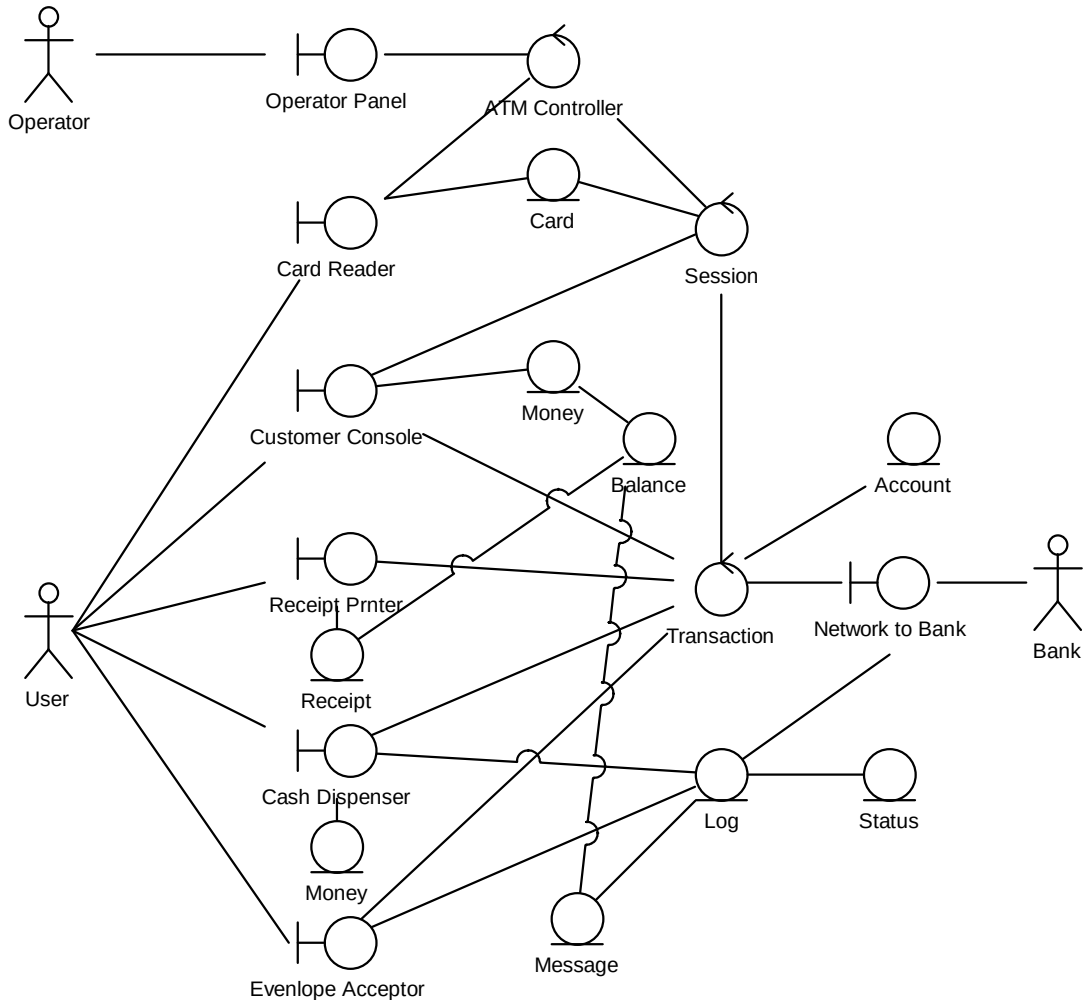
Message	
Responsibilities	**Collaborators**
Represent information to be sent over network to bank	

State	
Responsibilities	**Collaborators**
Represent information to be sent over network to bank	Represent transaction status information returned by bank

Log	
Responsibilities	**Collaborators**
Log messages sent to bank Log responses from bank Log dispensing of cash Log receiving an envelope	

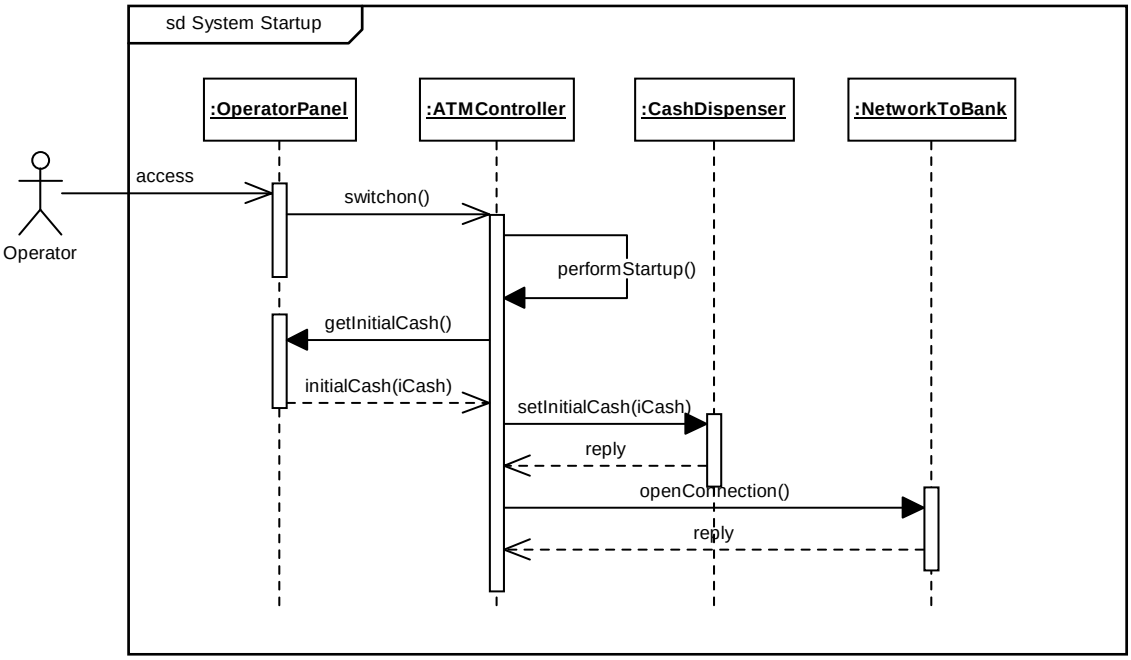
6. Analysis Class Diagram

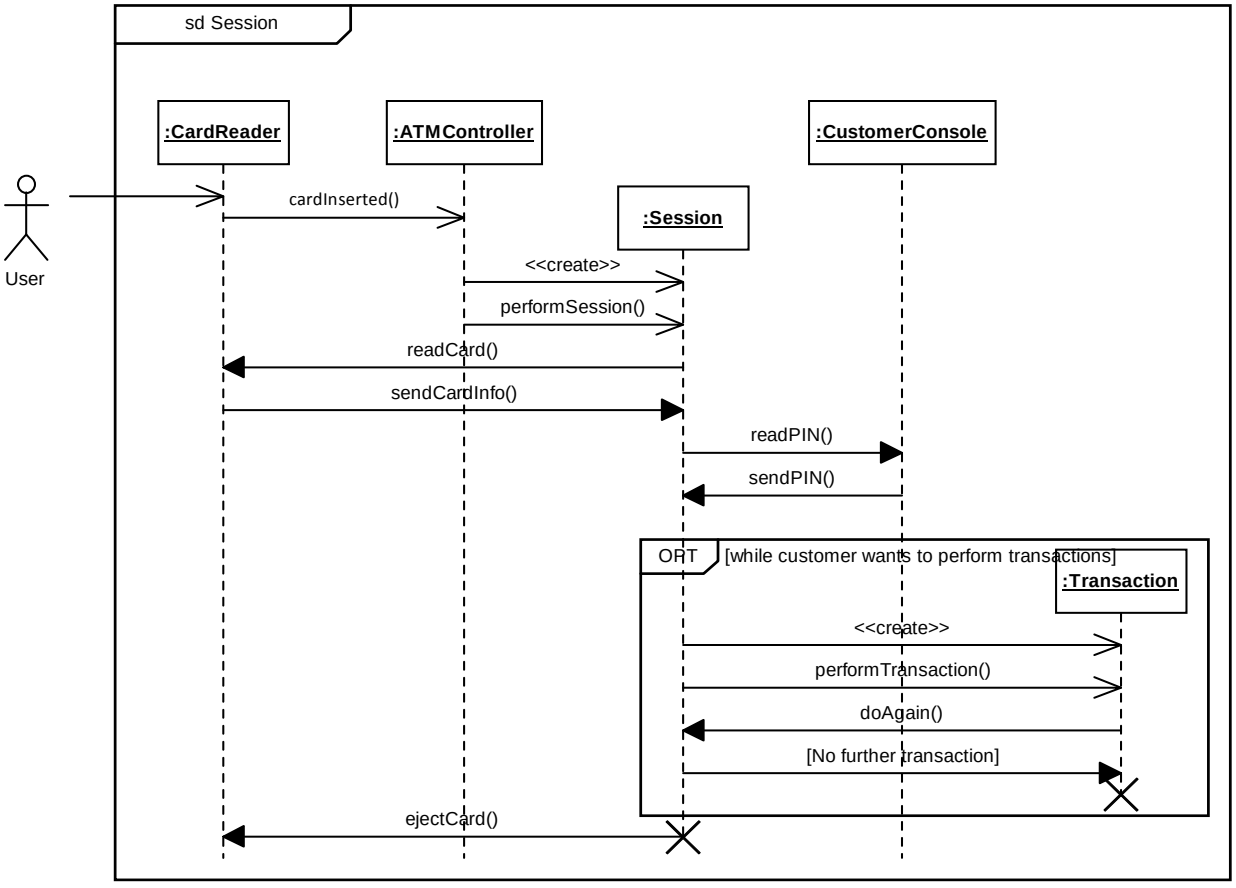




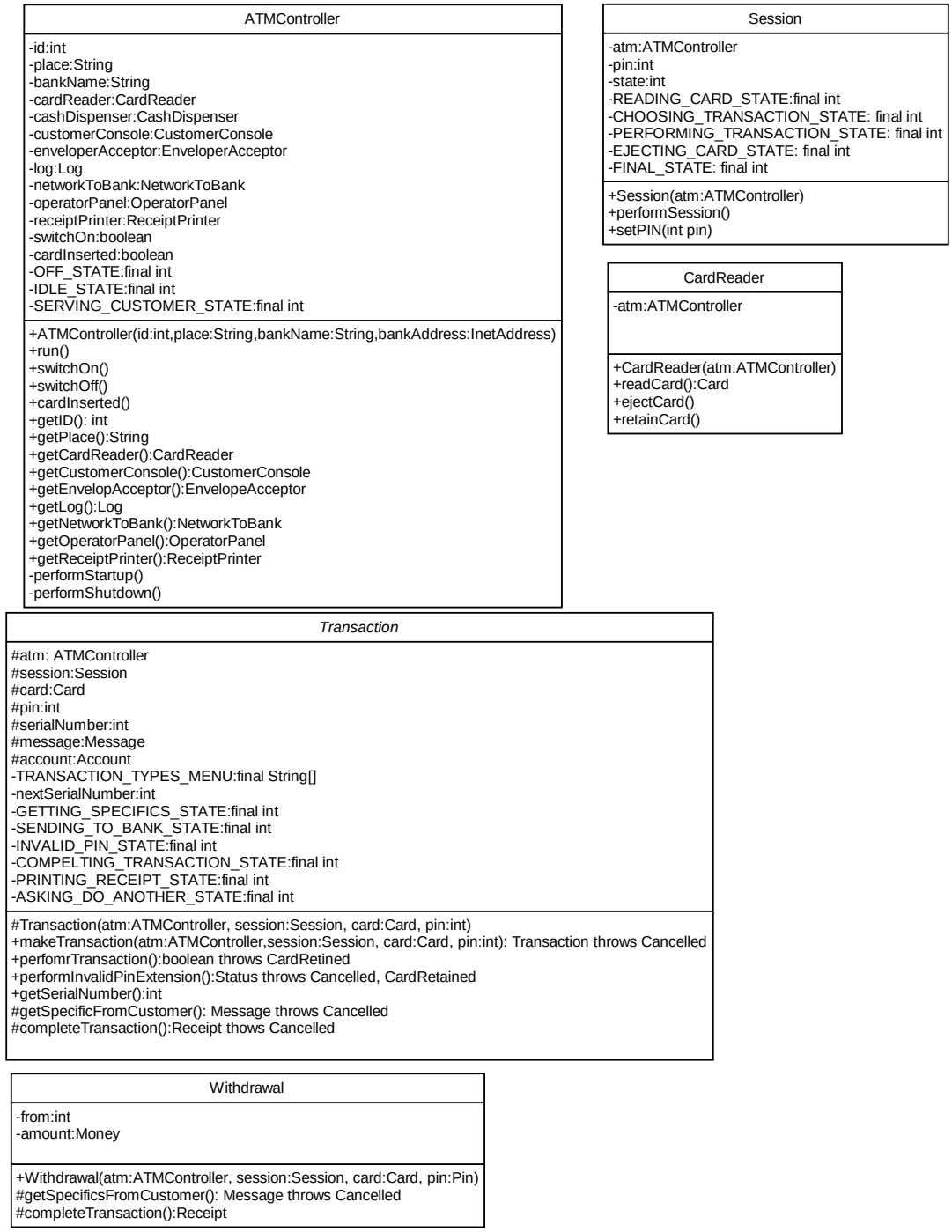
Boundary classes	Controller classes	Entity classes
Class CardReader	Class ATM Controller	Class Account
Class CashDispenser	Class Session	Class Card
Class CustomerConsole	Class Transaction	Class Receipt
Class EnvelopeAcceptor	Class Withdrawal	Class Log
Class NetworkToBank	Class Deposit	Class Money
Class OperatorPanel	Class Transfer	Class Status
Class ReceiptPrinter	Class Inquiry	Class Message

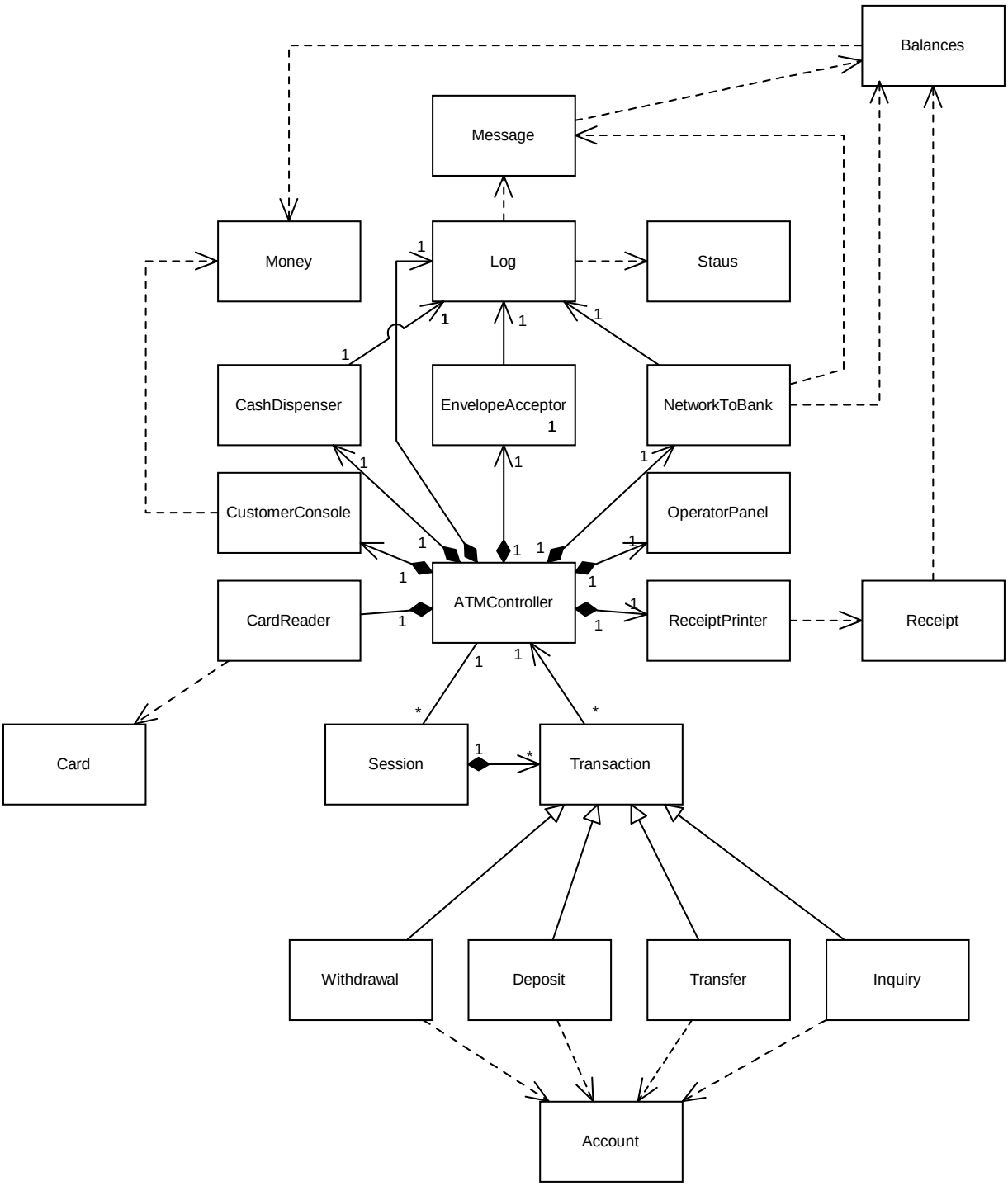
7. Sequence Diagram





8. Detailed Class Diagram




Balances
Message
Money
Log
Staus
CashDispenser
EnvelopeAcceptor
NetworkToBank
CustomerConsole
OperatorPanel
ATMController
CardReader
ReceiptPrinter
Receipt
Card
Session
Transaction
Withdrawal
Deposit
Transfer
Inquiry
Account

9. Implementation

```
/*
 * ATM Example system - file ATMController.java
 *
 */

package atm;
import java.net.InetAddress;
import atm.physical.*;
import banking.Card;
import banking.Money;

public class ATMController implements Runnable
{
   public ATMController(int id, String place, String bankName, InetAddress bankAddress)
   {
      this.id = id;
      this.place = place;
      this.bankName = bankName;
      this.bankAddress = bankAddress;

      // Create objects corresponding to component parts

      log = new Log();
      cardReader = new CardReader(this);
      cashDispenser = new CashDispenser(log);
      customerConsole = new CustomerConsole();
      envelopeAcceptor = new EnvelopeAcceptor(log);
      networkToBank = new NetworkToBank(log, bankAddress);
      operatorPanel = new OperatorPanel(this);
      receiptPrinter = new ReceiptPrinter();

      // Set up initial conditions when ATM first created

      state = OFF_STATE;
      switchOn = false;
      cardInserted = false;
   }

   // Methods corresponding to major responsibilities of the ATM
   public void run()
   {  Session currentSession = null;
      while (true) {
         switch(state) {
            case OFF_STATE:
               customerConsole.display("Not currently available");
               synchronized(this)  {
                  try
                  { wait();
                  }
                  catch(InterruptedException e)
                  { }
               }

               if (switchOn) {
                  performStartup();
```

```
            state = IDLE_STATE;
         }
         break;

      case IDLE_STATE:
         customerConsole.display("Please insert your card");
         cardInserted = false;

         synchronized(this) {
            try {
               wait();
            }
            catch(InterruptedException e)
            { }
         }

         if (cardInserted)  {
            currentSession = new Session(this);
            state = SERVING_CUSTOMER_STATE;
         }
         else if (! switchOn)  {
            performShutdown();
            state = OFF_STATE;
         }

         break;

      case SERVING_CUSTOMER_STATE:

         // The following will not return until the session has
         // completed

         currentSession.performSession();

         state = IDLE_STATE;

         break;

     }
   }
}

/** Inform the ATM that the switch on the operator console has been moved
 *  to the "on" position.
 */
public synchronized void switchOn()
{
   switchOn = true;
   notify();
}

/** Inform the ATM that the switch on the operator console has been moved
 *  to the "off" position.
 */
public synchronized void switchOff()
{  switchOn = false;
   notify();
}
```

```
/** Inform the ATM that a card has been inserted into the card reader.
 */
public synchronized void cardInserted()  {
   cardInserted = true;
   notify();
}

// The following methods allow objects of other classes to access component
// parts of the ATM
public int getID()
{
   return id;
}

public String getPlace()
{
   return place;
}

public String getBankName()
{
   return bankName;
}

public CardReader getCardReader()
{
   return cardReader;
}

public CashDispenser getCashDispenser()
{
   return cashDispenser;
}

public CustomerConsole getCustomerConsole()
{
   return customerConsole;
}

public EnvelopeAcceptor getEnvelopeAcceptor()
{
   return envelopeAcceptor;
}

public Log getLog()
{
   return log;
}

public NetworkToBank getNetworkToBank()
{
   return networkToBank;
}

/** Accessor for operator panel
 *
 *  @return operator panel component of this ATM
```

```
 */
public OperatorPanel getOperatorPanel()
{
   return operatorPanel;
}

public ReceiptPrinter getReceiptPrinter()
{
   return receiptPrinter;
}

// Private methods

/** Perform the System Startup use case when switch is turned on
 */
private void performStartup()
{
   Money initialCash = operatorPanel.getInitialCash();
   cashDispenser.setInitialCash(initialCash);
   networkToBank.openConnection();
}

/** Perform the System Shutdown use case when switch is turned off
 */
private void performShutdown()
{
   networkToBank.closeConnection();
}


// Instance variables recording information about the ATM


/** Unique ID for this ATM
 */
private int id;

/** Physical location of this ATM
 */
private String place;

/** Name of the bank owning this ATM
 */
private String bankName;

/** Internet address of the bank
 */
private InetAddress bankAddress;


// Instance variables referring to the omponent parts of the ATM

/** The ATM's card reader
 */
private CardReader cardReader;

/** The ATM's cash dispenser
 */
```

```
private CashDispenser cashDispenser;

/** The ATM's customer console
 */
private CustomerConsole customerConsole;

/** The ATM's envelope acceptor
 */
private EnvelopeAcceptor envelopeAcceptor;

/** The ATM's log
 */
private Log log;

/** The ATM's network connection to the bank
 */
private NetworkToBank networkToBank;

/** The ATM's operator panel
 */
private OperatorPanel operatorPanel;

/** The ATM's receipt printer
 */
private ReceiptPrinter receiptPrinter;


// State information


/** The current state of the ATM - one of the possible values listed below
 */
private int state;

/** Becomes true when the operator panel informs the ATM that the switch has
 *  been turned on - becomes false when the operator panel informs the ATM
 *  that the switch has been turned off.
 */
private boolean switchOn;

/** Becomes true when the card reader informs the ATM that a card has been
 *  inserted - the ATM will make this false when it has tried to read the
 *  card
 */
private boolean cardInserted;


// Possible values for state


/** The ATM is off.  The switch must be turned on before it can operate
 */
private static final int OFF_STATE = 0;

/** The ATM is on, but idle.  It can service a customer, or it can be shut down
 */
private static final int IDLE_STATE = 1;
```

```
   /** The ATM is servicing a customer.
    */
   private static final int SERVING_CUSTOMER_STATE = 2;
}



/*
 * ATM Example system - file Withdraw.java
 *
*/


package atm.transaction;
import atm.ATMController;
import atm.Session;
import atm.physical.*;
import banking.AccountInformation;
import banking.Card;
import banking.Message;
import banking.Money;
import banking.Status;
import banking.Receipt;

/** Representation for a cash withdrawal transaction
 */
public class Withdrawal extends Transaction
{
   /** Constructor
    *
    *  @param atm the ATM used to communicate with customer
    *  @param session the session in which the transaction is being performed
    *  @param card the customer's card
    *  @param pin the PIN entered by the customer
    */
   public Withdrawal(ATMController atm, Session session, Card card, int pin)
   {
      super(atm, session, card, pin);
   }

   /** Get specifics for the transaction from the customer
    *
    *  @return message to bank for initiating this transaction
    *  @exception CustomerConsole.Cancelled if customer cancelled this transaction
    */
   protected Message getSpecificsFromCustomer() throws CustomerConsole.Cancelled
   {
      from = atm.getCustomerConsole().readMenuChoice(
         "Account to withdraw from",
         AccountInformation.ACCOUNT_NAMES);

      String [] amountOptions = { "$20", "$40", "$60", "$100", "$200" };
      Money [] amountValues = {
                        new Money(20), new Money(40), new Money(60),
                        new Money(100), new Money(200)
                       };

      String amountMessage = "";
```

```
      boolean validAmount = false;

      while (! validAmount)
      {
         amount = amountValues [
            atm.getCustomerConsole().readMenuChoice(
               amountMessage + "Amount of cash to withdraw", amountOptions) ];

         validAmount = atm.getCashDispenser().checkCashOnHand(amount);

         if (! validAmount)
            amountMessage = "Insufficient cash available\n";
      }

      return new Message(Message.WITHDRAWAL,
                  card, pin, serialNumber, from, -1, amount);

   }

   /** Complete an approved transaction
    *
    *  @return receipt to be printed for this transaction
    */
   protected Receipt completeTransaction()
   {
      atm.getCashDispenser().dispenseCash(amount);
      return new Receipt(this.atm, this.card, this, this.balances) {
         {
            detailsPortion = new String[2];
            detailsPortion[0] = "WITHDRAWAL FROM: " +
                        AccountInformation.ACCOUNT_ABBREVIATIONS[from];
            detailsPortion[1] = "AMOUNT: " + amount.toString();
         }
      };
   }

   /** Account to withdraw from
    */
   private int from;

   /** Amount of money to withdraw
   */
   private Money amount;
}


/*
 * ATM Example system - file CardReader.java
 *
*/

package atm.physical;
import atm.ATMController;
import banking.Card;
import simulation.Simulation;

/** Manager for the ATM's card reader.  In a real ATM, this would
 *  manage a physical device; in this simulation, it uses classes
```

```
 *  in package simulation to simulate the device.
 */

public class CardReader
{
   /** Constructor
    *
    *  @param atm the ATM that owns this card reader
    */
   public CardReader(ATMController atm)
   {
      this.atm = atm;
   }

   // In a real ATM, code would be needed to sense insertion of a card into the
   // slot and notify the ATM - simulated in this case by a button in the GUI

   /** Read a card that has been partially inserted into the reader
    *
    *  @return Card object representing information on the card if read
    *          successfully, null if not read successfully
    */
   public Card readCard()
   {
      return Simulation.getInstance().readCard();
   }

   /** Eject the card that is currently inside the reader.
    */
   public void ejectCard()
   {
      Simulation.getInstance().ejectCard();
   }

   /** Retain the card that is currently inside the reader for action by the
    *  bank.
    */
   public void retainCard()
   {
      Simulation.getInstance().retainCard();
   }

   /** The ATM to which this card reader belongs
    */
   private ATMController atm;
}
```

10. Package Diagram