Session 6

UML-Unified Modeling Language

Case-ATM machine

Types Of UML

- Sketching
  - Main goal just to communication an idea for layman
  - Not Specific just basic idea
- Blue print
  - Detail design decision which a programmer might be able to directly translate into code

Model driven architecture

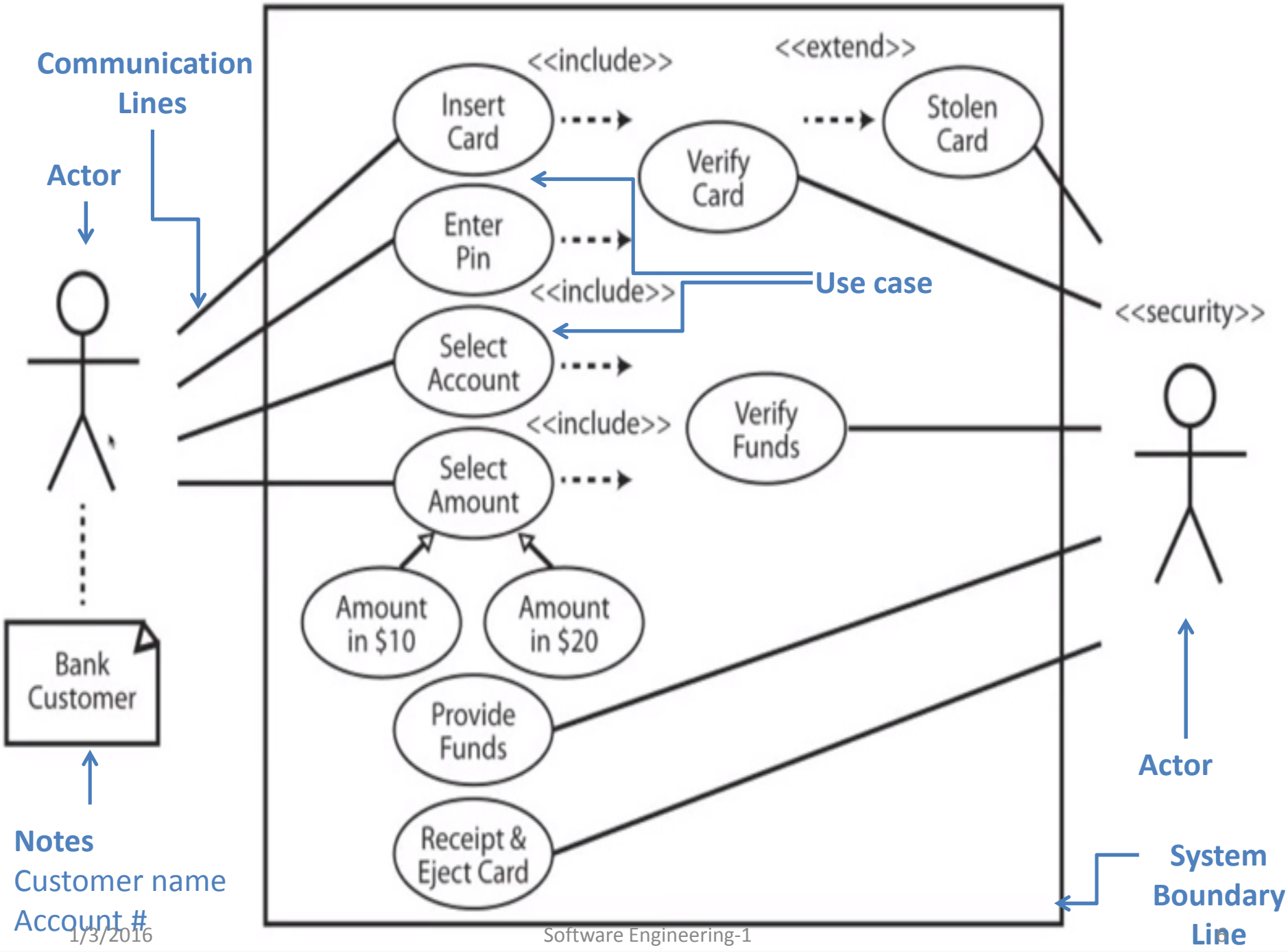
- Platform dependent ……UML… model independent of any specific technology.
- Platform dependent …….model specific to a execution environment or a programming language.

UML development process

- Water fall
- Iterative
  - Mostly for iterative

Use case

- List of steps a system needs to follow to reach a goal…..
- How your program will solve the problem……
- What are requirement and how they are or will be met….
- Shall vs. should requirements……..
- No code in use case diagram…….
- These steps include interaction with actors----humans and external systems…Machines……..


Communication Lines
Actor
<<include>>
<<extend>>
Insert Card
Stolen Card
Verify Card
Enter Pin
Use case
<<include>>
<<security>>
Select Account
<<include>>
Verify Funds
Select Amount
Amount in $10
Amount in $20
Bank Customer
Provide Funds
Actor
Receipt & Eject Card
Notes
Customer name
Account #
System Boundary Line

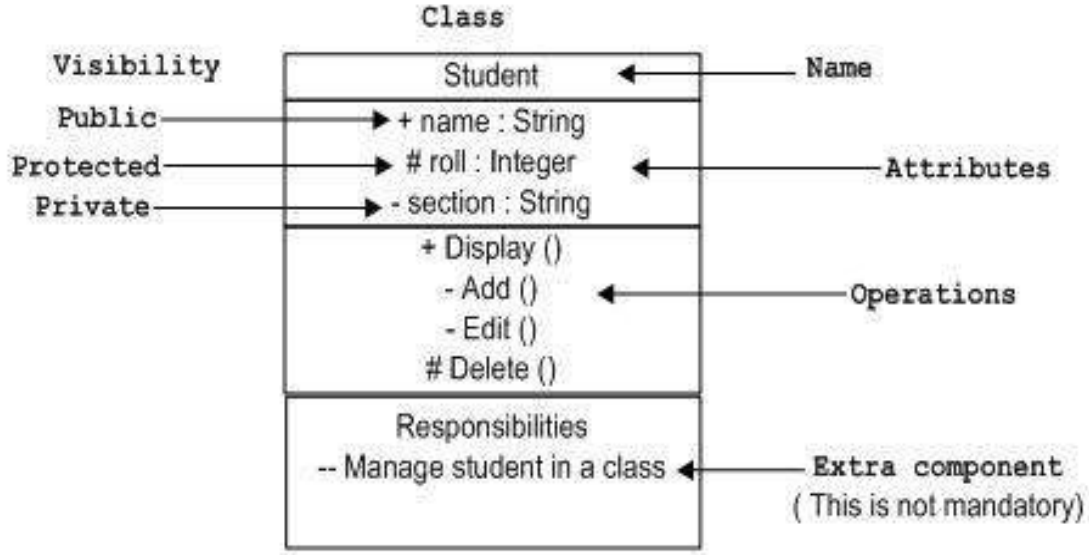
UML planning process

- **Predictive**
  - if all the requirements can be listed easily—100 % sure requirement would not be changed----all the team on the same page
- **Adaptive and Agile**.
  - Differences are inevitable, additional functionality addition

UML-Notations

Class Notation:

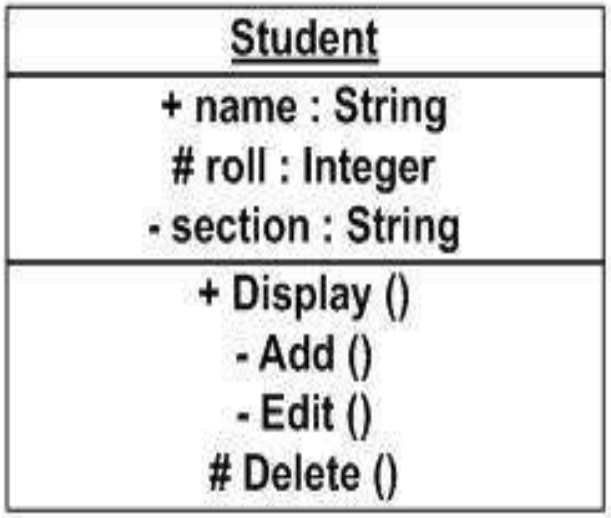
- UML *class* is represented by the diagram shown below. The diagram is divided into four parts.
- The top section is used to name the class.
- The second one is used to show the attributes of the class.
- The third section is used to describe the operations performed by the class.
- The fourth section is optional to show any additional components.



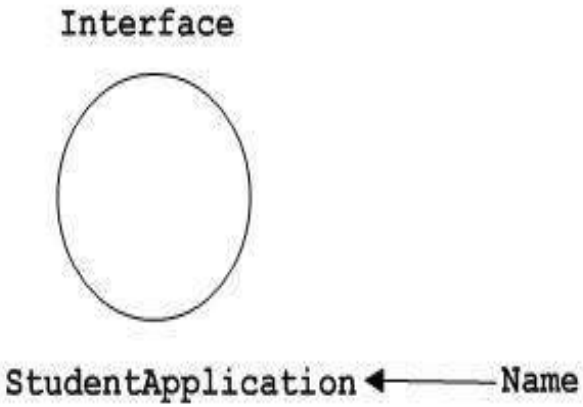
Object

Notation:

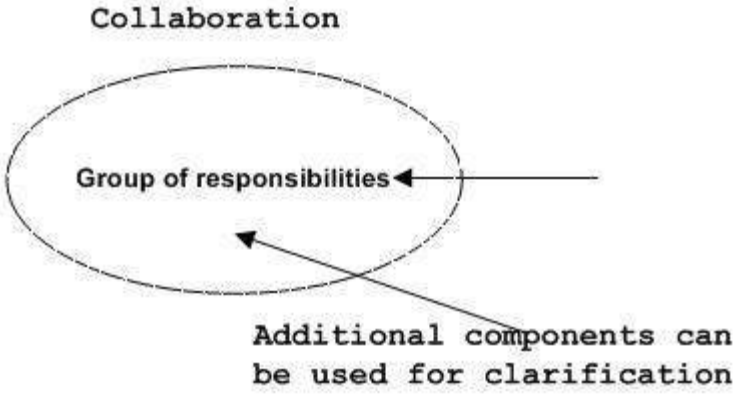
- The *object* is represented in the same way as the class. The only difference is the *name* which is underlined as shown below.



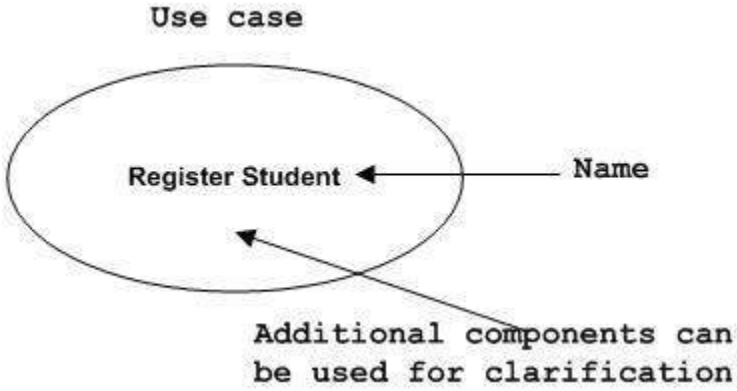
- Describe functionality without implementation.



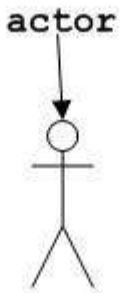
- Represents responsibilities or group of responsibilities.



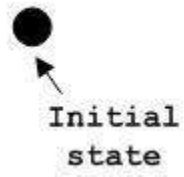
Use case notation represents high level functionalities



An actor can be defined as some internal or external entity that interacts with the system.

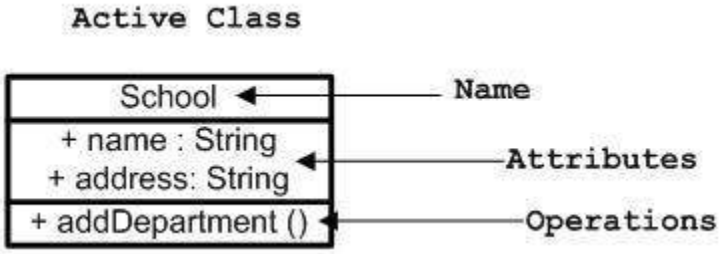


Initial state is defined to show the start of a process.

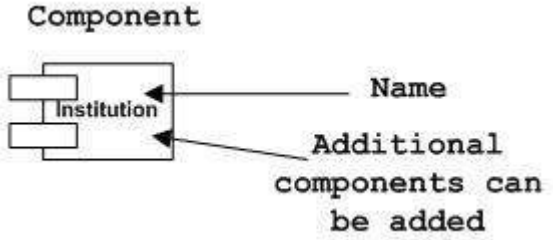


Represents end of the process

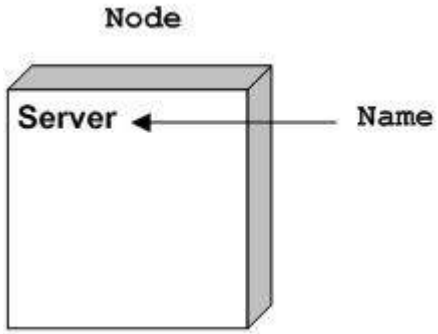
Active class looks similar to a class with a solid border. Active class is generally used to describe concurrent behavior of a system.



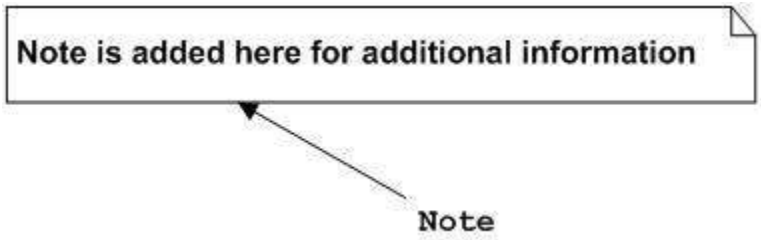
Component is used to represent any part of a system for which UML diagrams are made.



Node is used to represent physical part of a system like server, network etc.

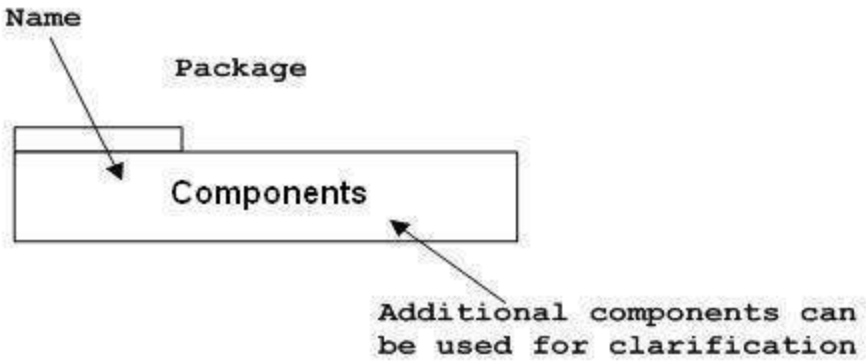


provide necessary information of a system.

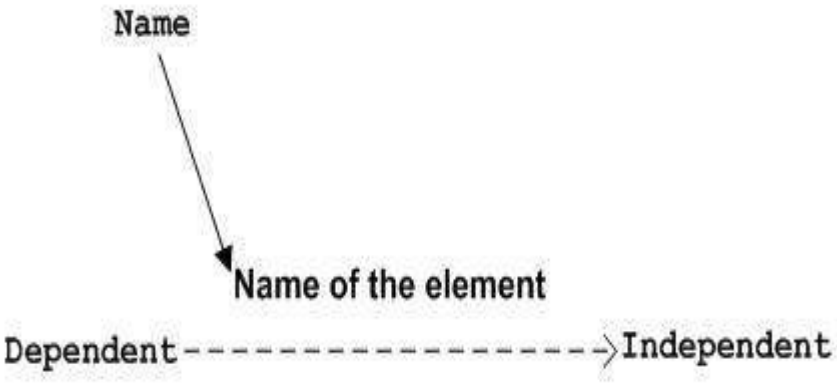


Represents end of the process

Package notation is shown below and this is used to wrap the components of a system.

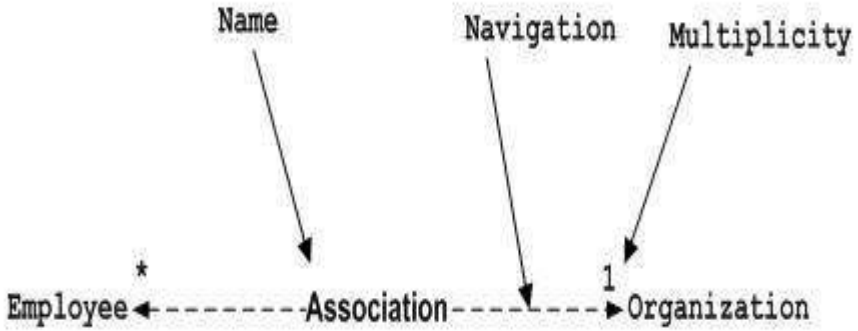


Dependency is used to represent dependency between two elements of a system.

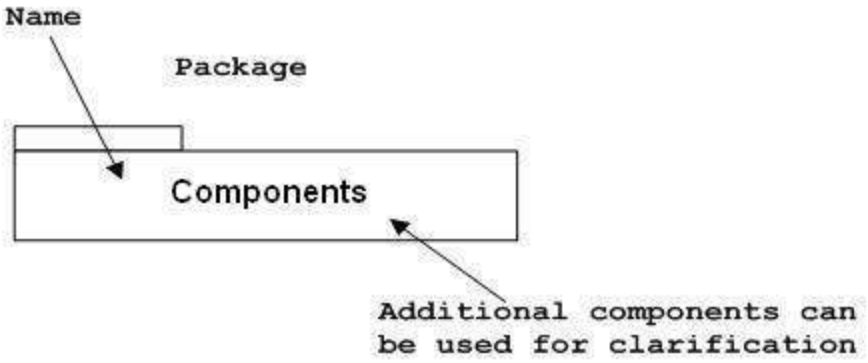


Generalization is represented by an arrow with hollow arrow head as shown below. One end represents the parent element and the other end child element.

describes how many elements are taking part in an interaction.
The two ends represent two associated elements

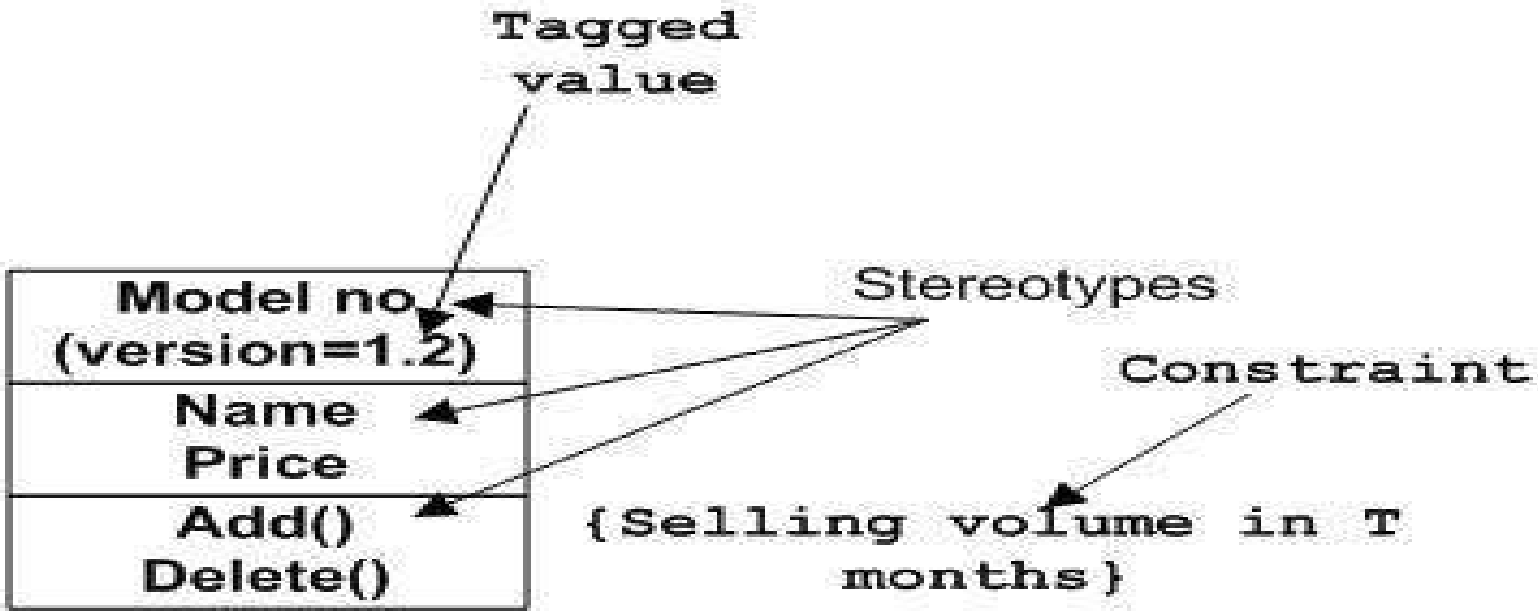


Package notation is shown below and this is used to wrap the components of a system.



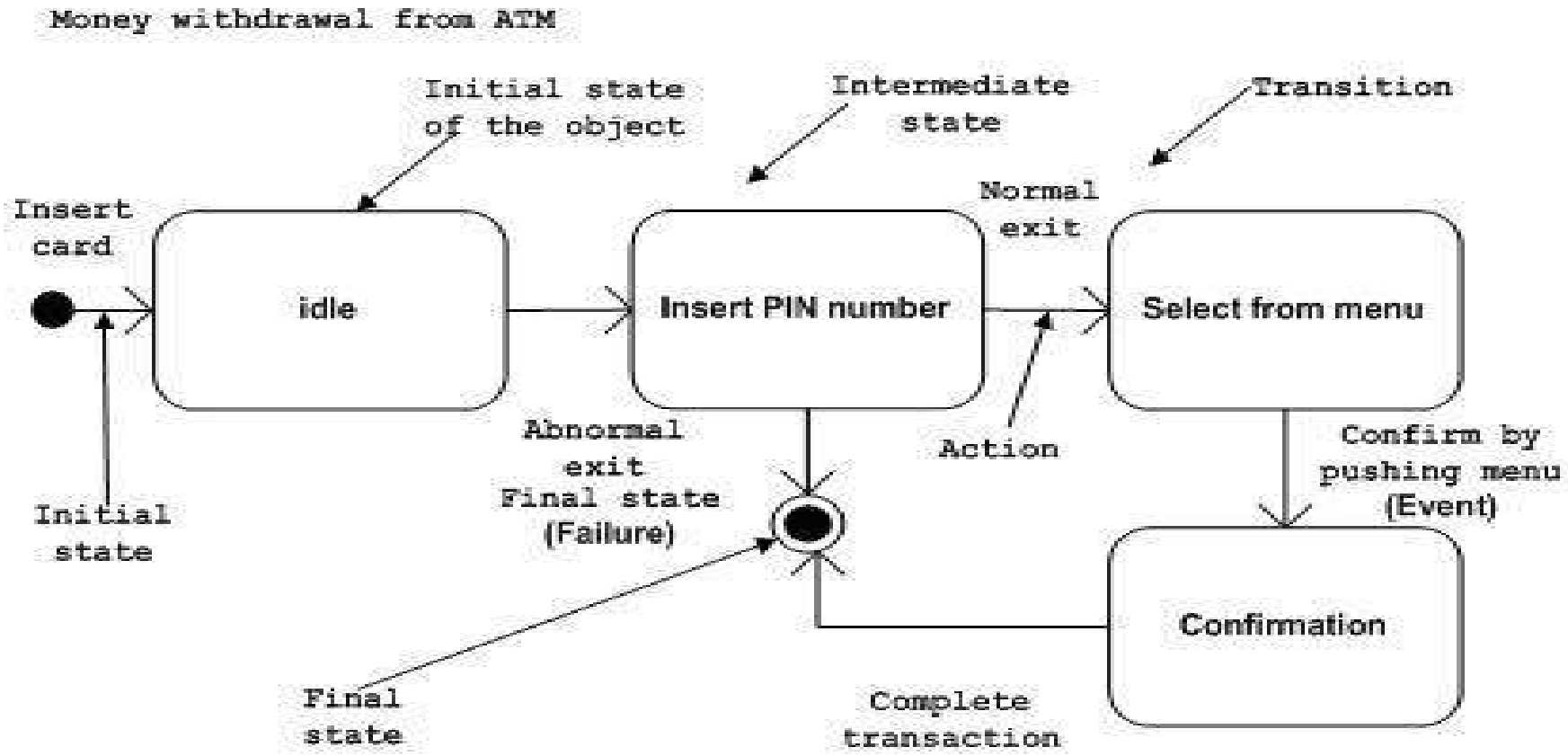
Extensibility Notation:

- All the languages (programming or modeling) have some mechanism to extend its capabilities like syntax, semantics etc. UML is also having the following mechanisms to provide extensibility features.
- Stereotypes (Represents new elements)
- Tagged values (Represents new attributes)
- Constraints (Represents the boundaries)



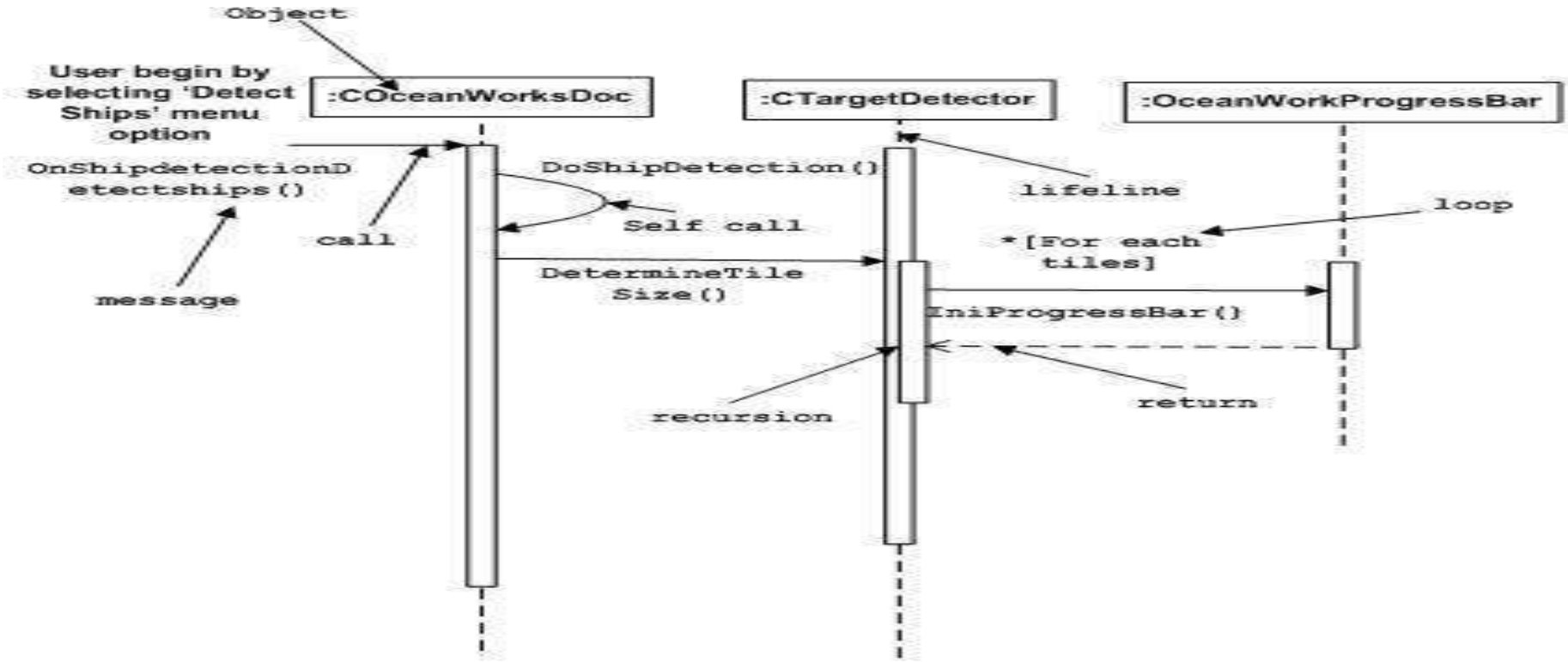
State machine Notation:

- State machine describes the different states of a component in its life cycle. The notations are described in the following diagram.
- State machine is used to describe different states of a system component. The state can be active, idle or any other depending upon the situation.



Interaction notation

- Interaction is basically message exchange between two UML components. The following diagram represents different notations used in an interaction.
- Interaction is used to represent communication among the components of a system.



Case

ATM Machine

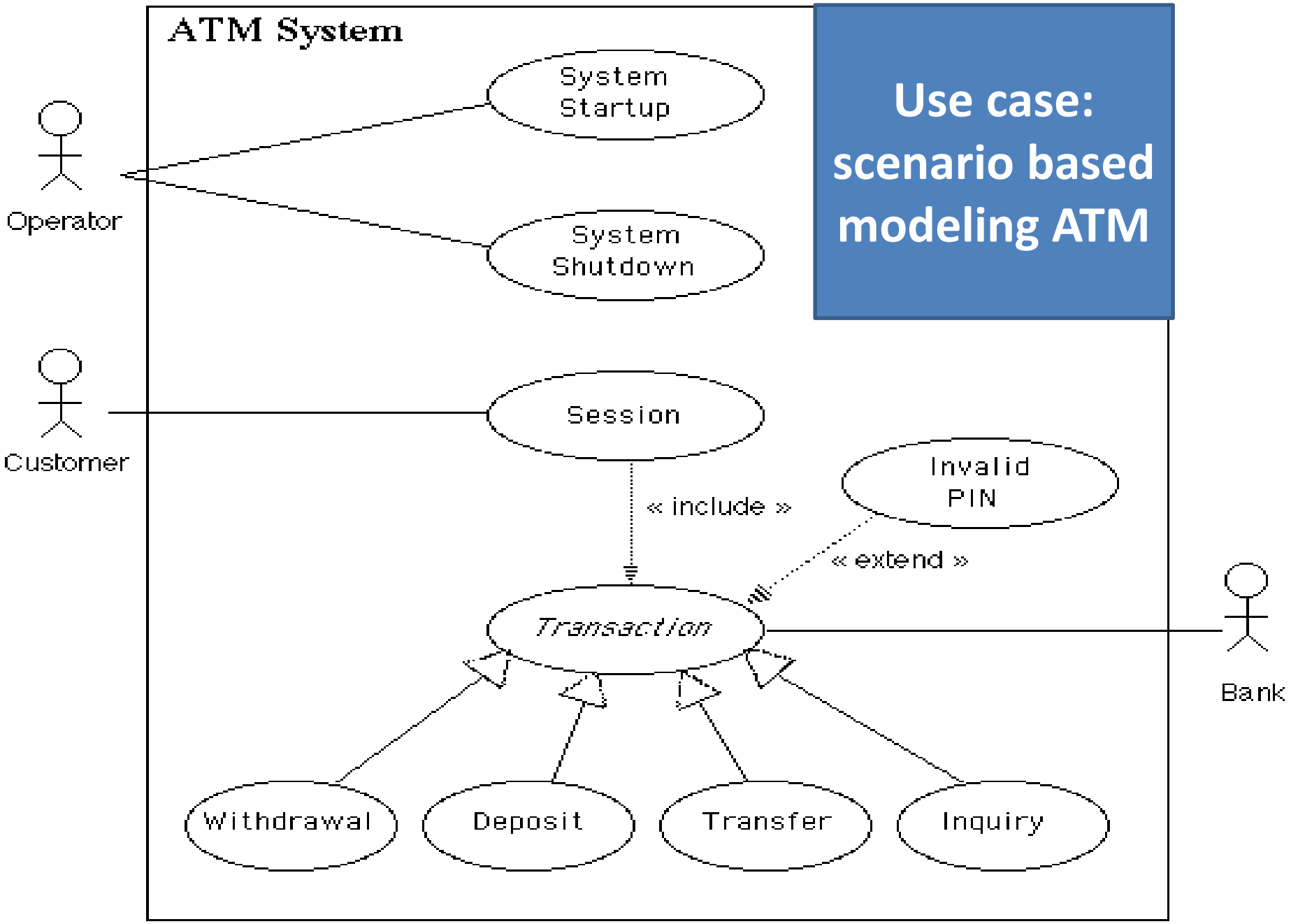
Requirement statement for an ATM machine

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

- A customer must be able to make a **cash withdrawal** from any suitable account linked to the card. Approval must be obtained from the bank before cash is dispensed.
- A customer must be able to make a deposit to any account linked to the card.
- A customer must be able to make a **transfer of money** between any two accounts linked to the card.
- 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.

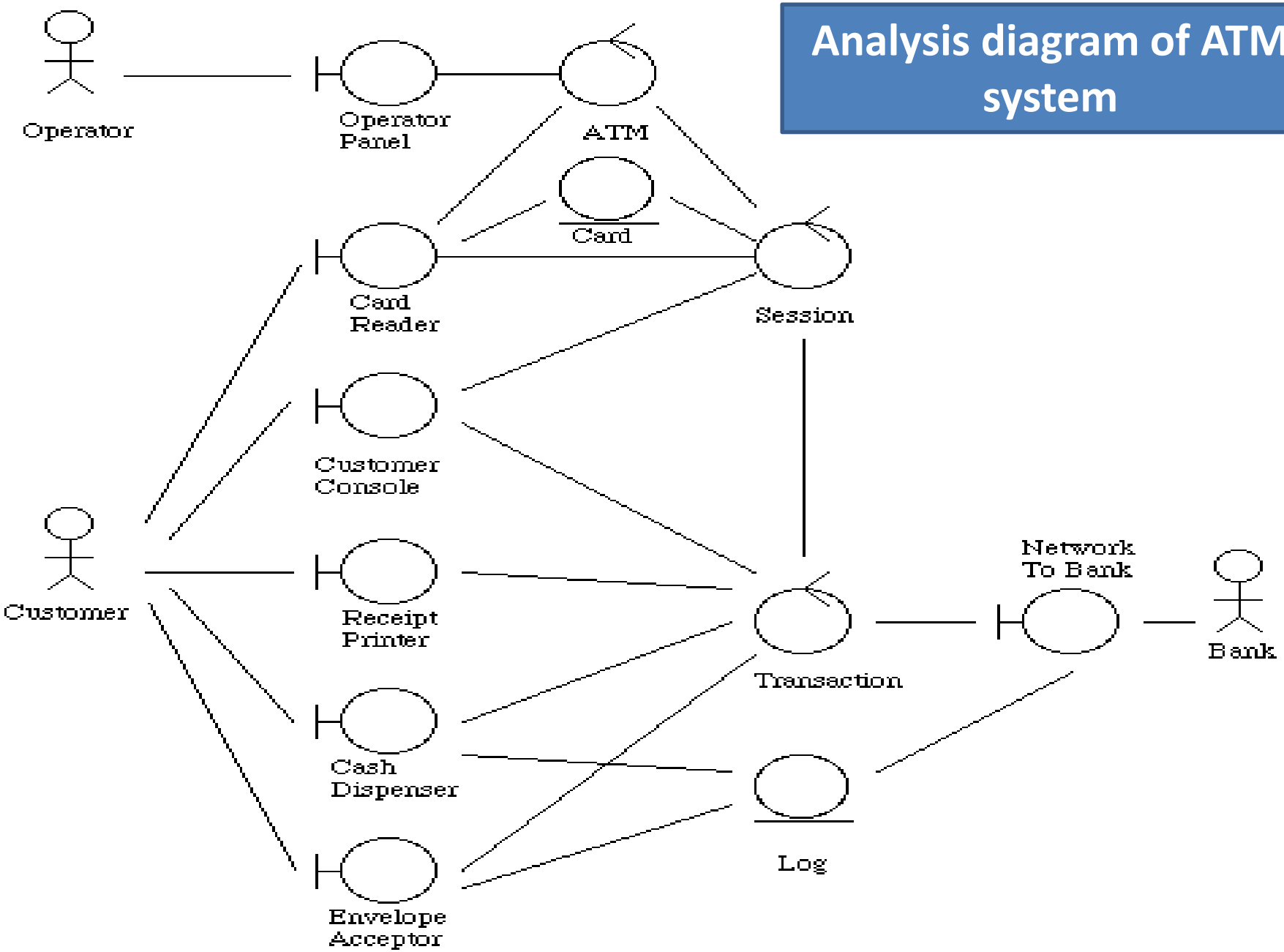
- The ATM will communicate each transaction to the bank and **obtain verification** that it was allowed by the bank.
- A transaction will be considered **complete** by the bank once it has been approved.
- In the **case of a deposit, a second message** will be sent to the bank indicating that the customer 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 **three 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, amount, and ending and available balance.
- The ATM will have a **key-operated switch that will allow an operator to start and stop the servicing of customers**. 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.**



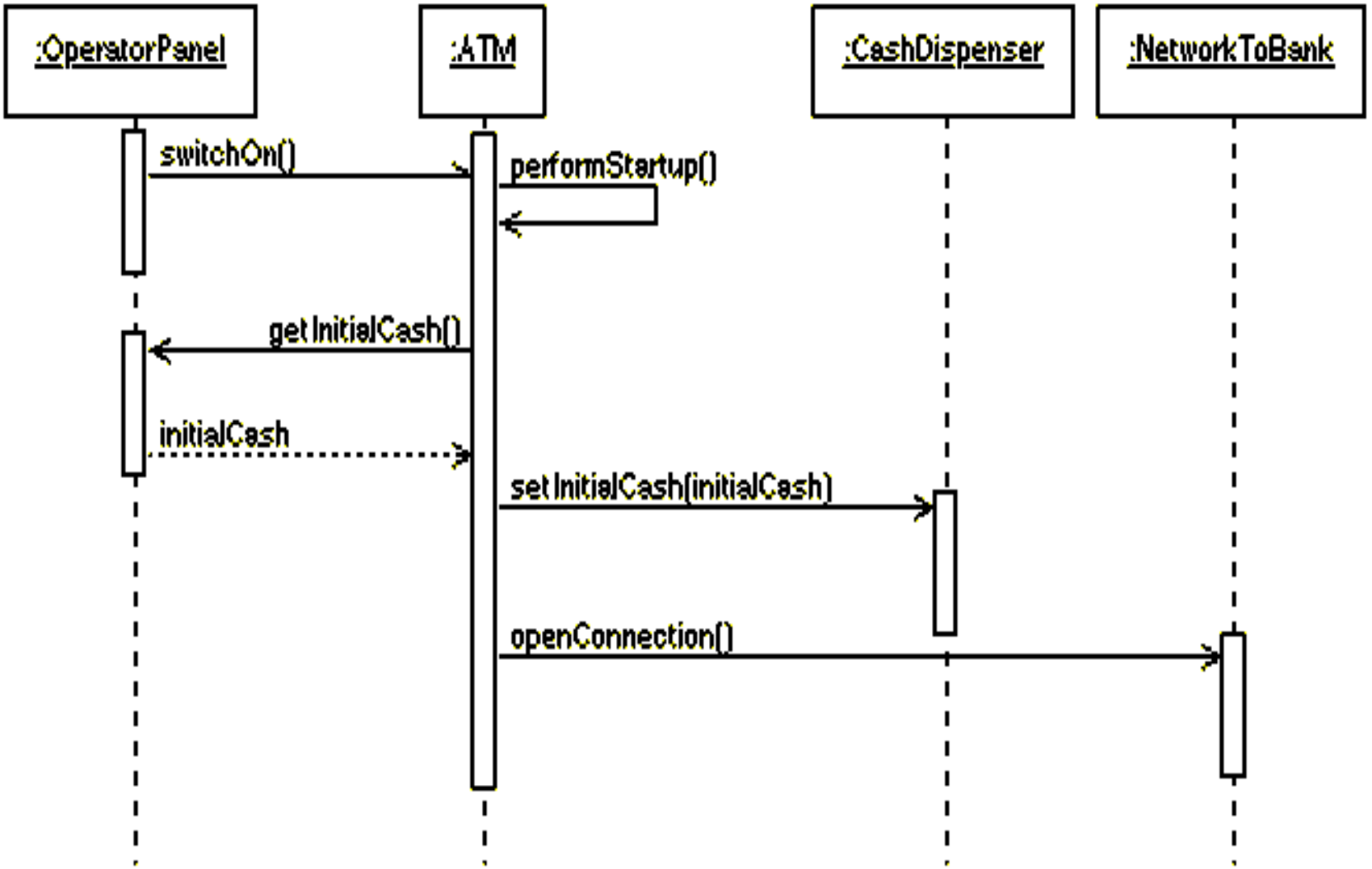
Use case: scenario based modeling ATM

Analysis diagram of ATM system

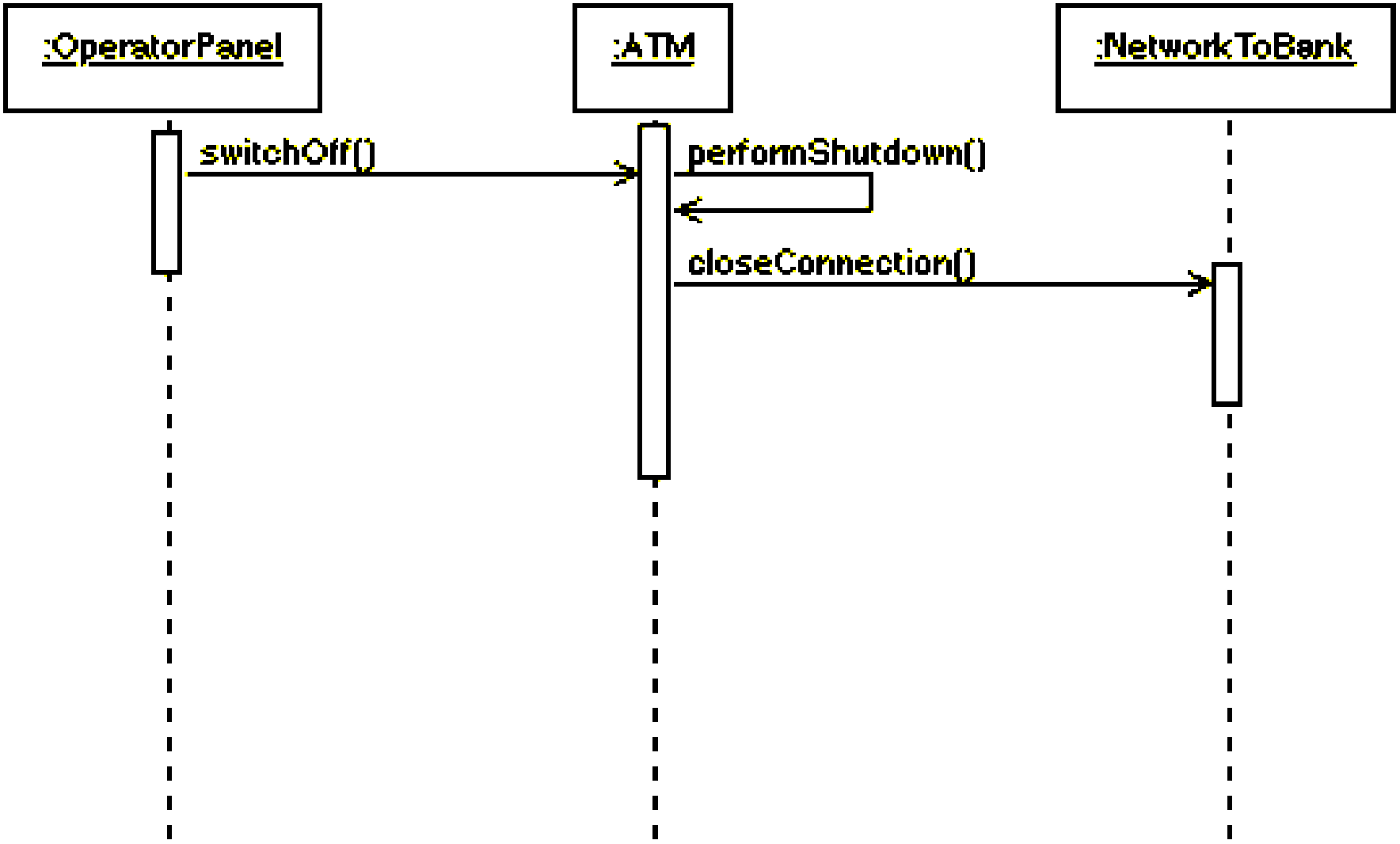


Flow of Events individual Use cases

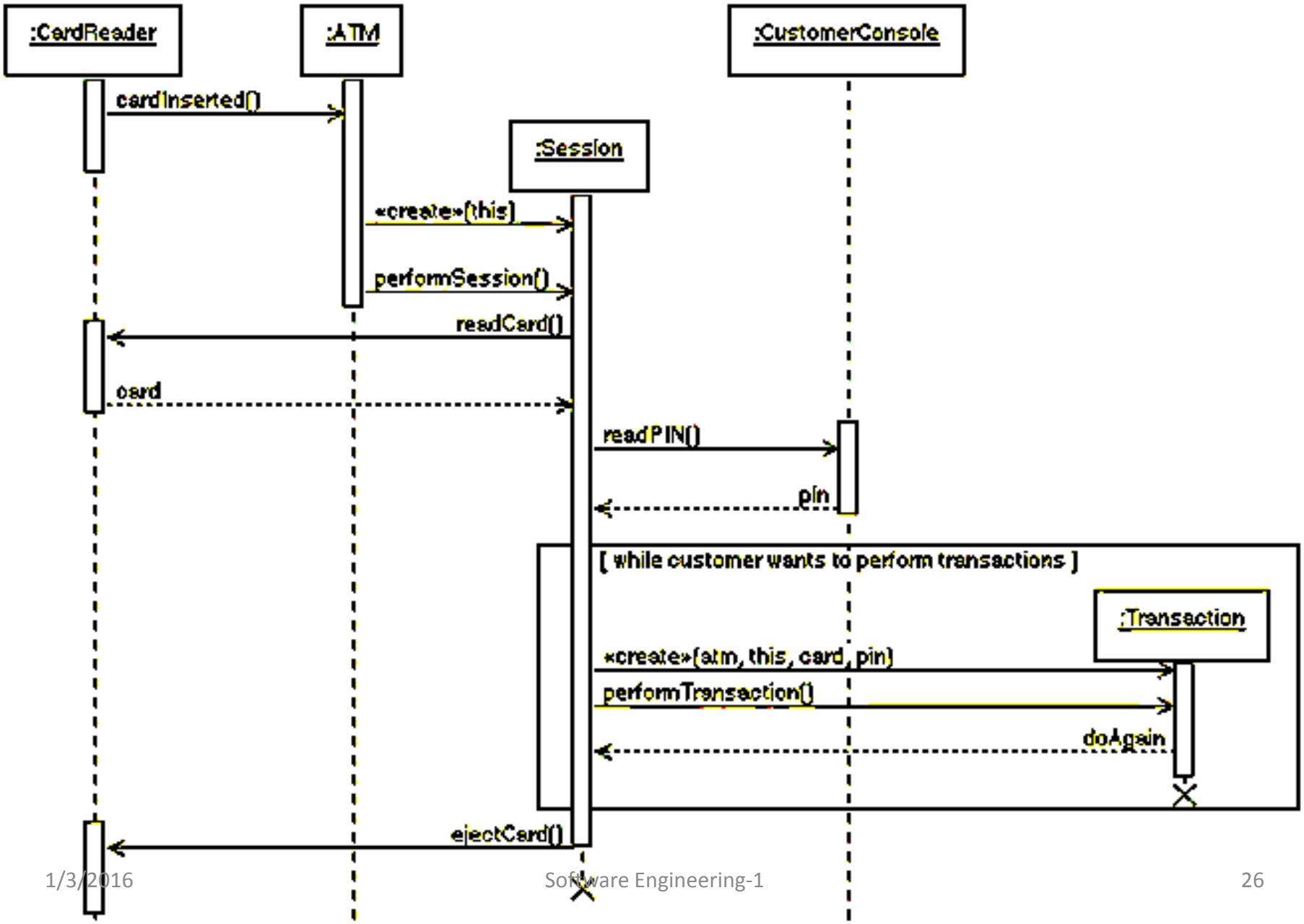
System Startup Sequence Diagram



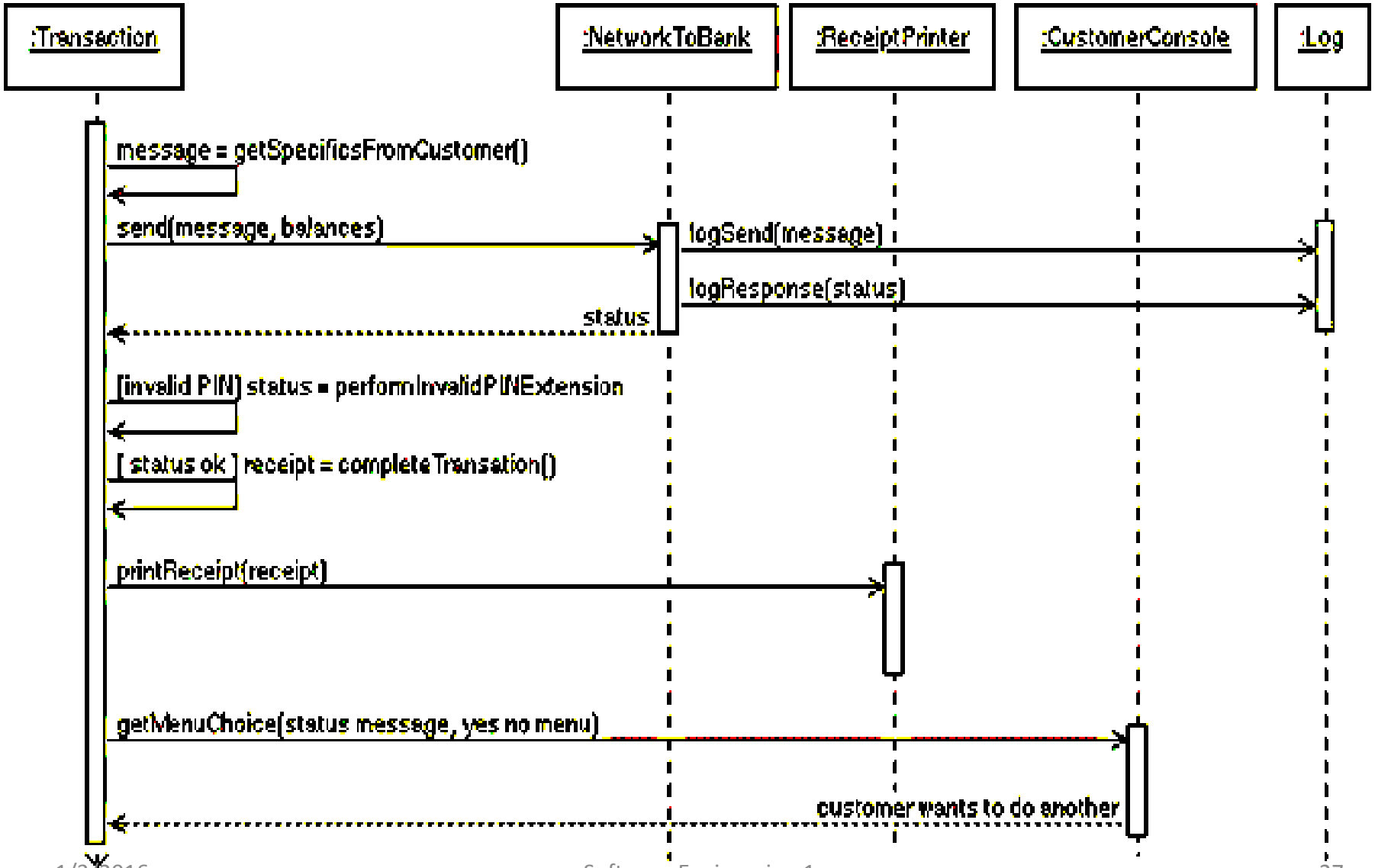
System Shutdown Sequence Diagram



Session Sequence Diagram



Transaction Sequence Diagram



Analysis Classes

- An initial reading of the use cases suggests that the following will be part of the system.
- A controller object representing the ATM itself (managing the boundary objects listed below.)
- **Boundary objects** representing the individual component parts of the ATM:
  - Operator panel.
  - Card reader.
  - Customer console, consisting of a display and keyboard.
  - Network connection to the bank.
  - Cash dispenser.
  - Envelope acceptor.
  - Receipt printer.
- **Controller objects** corresponding to use cases. (Note: class ATM can handle the Startup and Shutdown use cases itself, so these do not give rise to separate objects here.)
  - Session
  - Transaction (abstract generalization, responsible for common features, with concrete specializations responsible for type-specific portions)
- **An entity object** representing the information encoded on the ATM card inserted by customer.
- An entity object representing the log of transactions maintained by the machine.
- This leads to the following diagram of analysis classes:

Class ATM

- **Boundary/entity objects - component parts of the ATM.**
  - Class CardReader
  - Class CashDispenser
  - Class CustomerConsole
  - Class EnvelopeAcceptor
  - Class Log
  - Class NetworkToBank
  - Class OperatorPanel
  - Class ReceiptPrinter
- **Controller objects corresponding to the various use cases.**
  - Class Session
  - Class Transaction
  - Class Withdrawal
  - Class Deposit
  - Class Transfer
  - Class Inquiry
- **Entity objects found necessary when assigning responsibilities to other objects.**
  - Class Balances
  - Class Card
  - Class Message
  - Class Receipt
  - Class Status

Class ATM

Responsibilities

Start up when switch is turned on

Shut down when switch is turned off

Start a new session when card is inserted by customer

Provide access to component parts for sessions and transactions

Collaborators

Operator Panel
Cash Dispenser
Network To Bank
Network To Bank
Customer Console Session

Class Card Reader

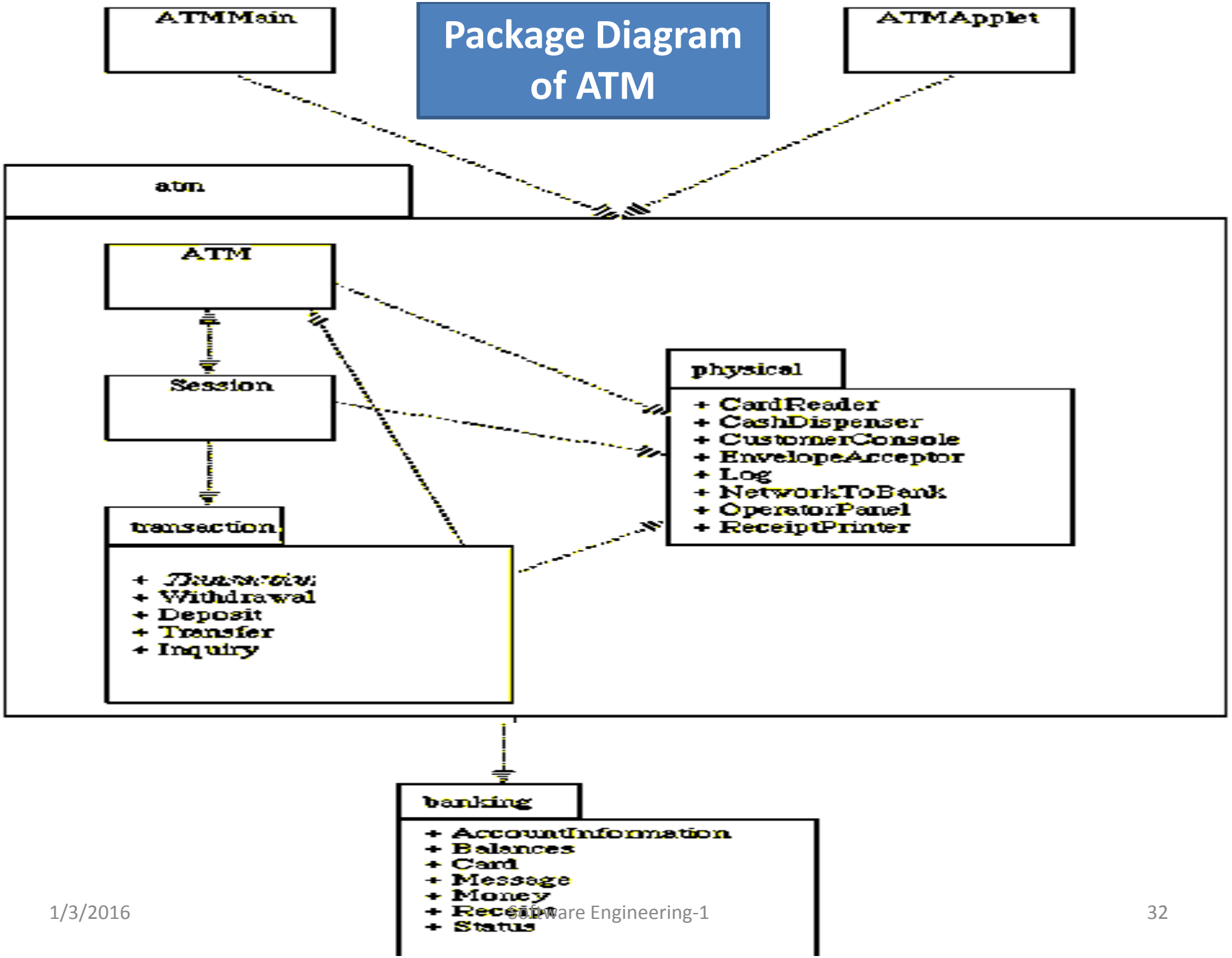
Responsibilities

Inform ATM as card is inserted
Read information of the card
Eject card
Retain card

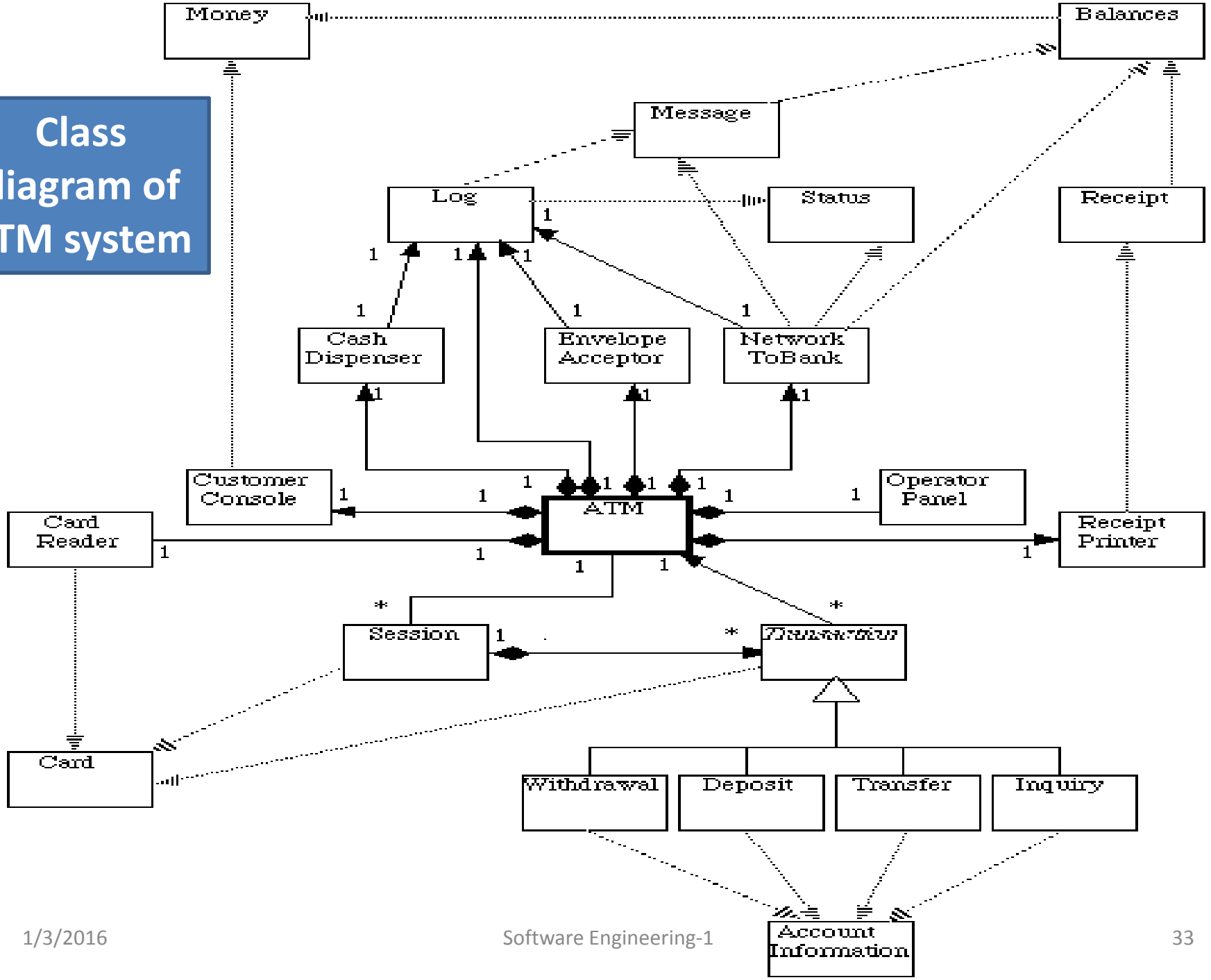
Collaborators

ATM
Card

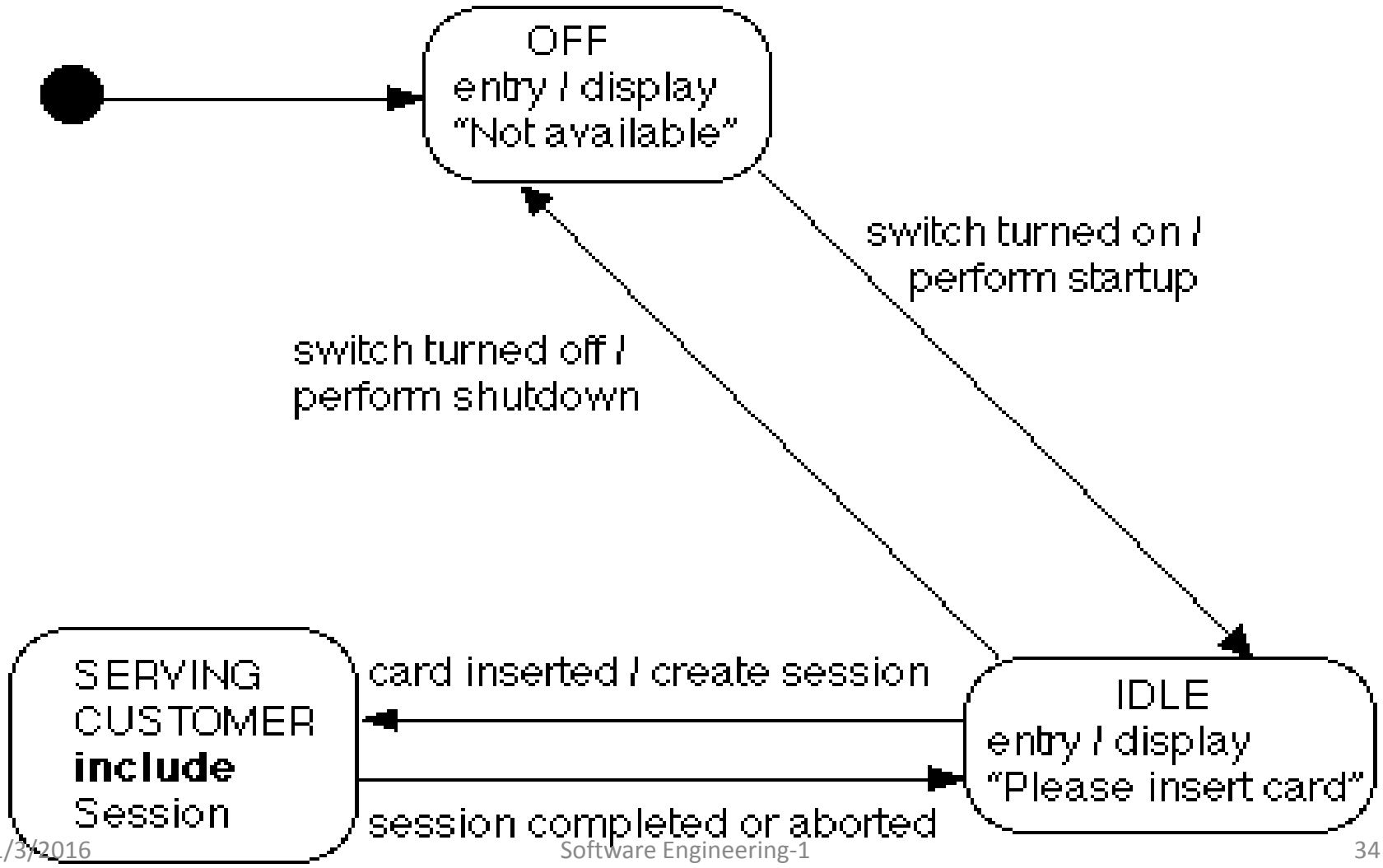
Package Diagram of ATM



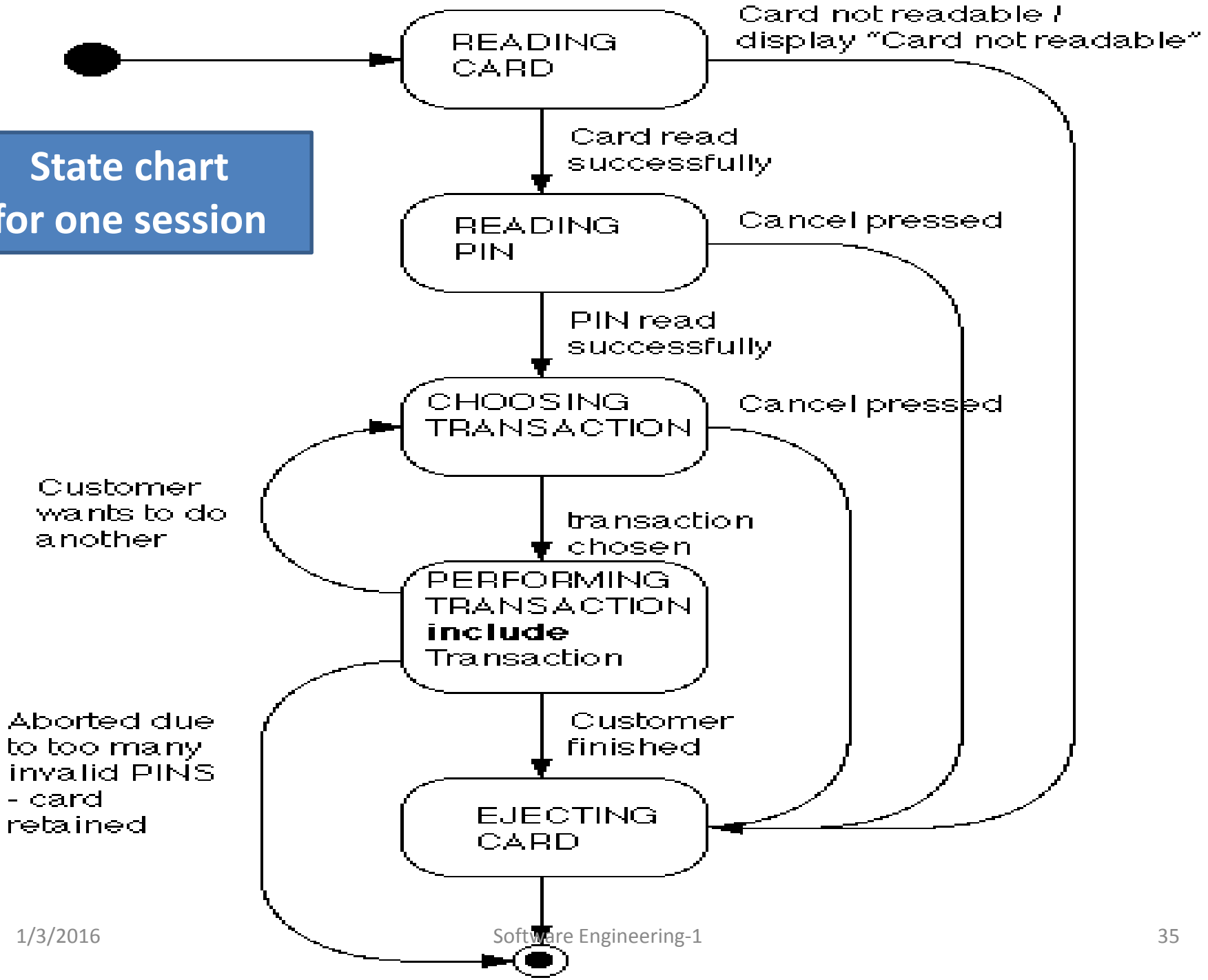
Class diagram of ATM system



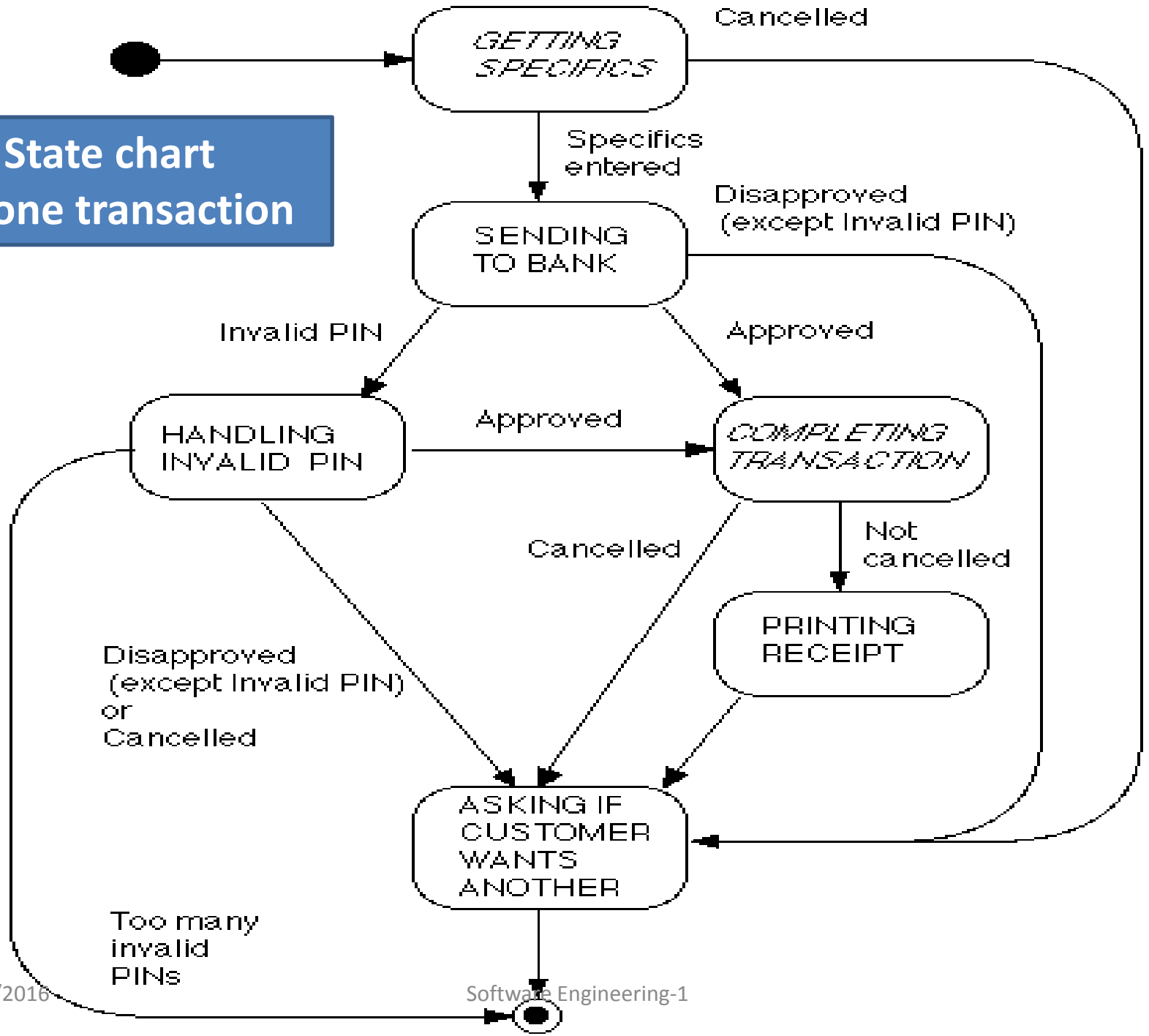
State chart for over all ATM for system startup and shut down case



State chart for one session



State chart for one transaction



Testing

Use Case	Function Being Tested	Initial System State	Input	Expected Output
System Startup	System is started when the switch is turned "on"	System is off	Activate the "on" switch	System requests initial cash amount
System Startup	System accepts initial cash amount	System is requesting cash amount	Enter a legitimate amount	System is on
System Startup	Connection to the bank is established	System has just been turned on	Perform a legitimate inquiry transaction	System output should demonstrate that a connection has been established to the Bank
System Shutdown	System is shut down when the switch is turned "off"	System is on and not servicing a customer	Activate the "off" switch	System is off
System Shutdown	Connection to the Bank is terminated when the system is shut down	System has just been shut down		Verify from the bank side that a connection to the ATM no longer exists

Use Case	Function Being Tested	Initial System State	Input	Expected Output
Session	System reads a customer's ATM card	System is on and not servicing a customer	Insert a readable card	Card is accepted; System asks for entry of PIN
Session	System rejects an unreadable card	System is on and not servicing a customer	Insert an unreadable card	Card is ejected; System displays an error screen; System is ready to start a new session
Session	System accepts customer's PIN	System is asking for entry of PIN	Enter a PIN	System displays a menu of transaction types
Session	System allows customer to perform a transaction	System is displaying menu of transaction types	Perform a transaction	System asks whether customer wants another transaction
Session	System allows multiple transactions in one session	System is asking whether customer wants another transaction	Answer yes	System displays a menu of transaction types
Session	Session ends when customer chooses not to do another transaction	System is asking whether customer wants another transaction	Answer no	System ejects card and is ready to start a new session

Use Case	Function Being Tested	Initial System State	Input	Expected Output
Transaction	System handles an invalid PIN properly	A readable card has been entered	Enter an incorrect PIN and then attempt a transaction	The Invalid PIN Extension is performed
Withdrawal	System asks customer to choose an account to withdraw from	Menu of transaction types is being displayed	Choose Withdrawal transaction	System displays a menu of account types
Withdrawal	System asks customer to choose a dollar amount to withdraw	Menu of account types is being displayed	Choose checking account	System displays a menu of possible withdrawal amounts
Withdrawal	System performs a legitimate withdrawal transaction properly	System is displaying the menu of withdrawal amounts	Choose an amount that the system currently has and which is not greater than the account balance	System dispenses this amount of cash; System prints a correct receipt showing amount and correct updated balance; System records transaction correctly in the log (showing both message to the bank and approval back)

Success is just the state of mind

And

Thank you for your time