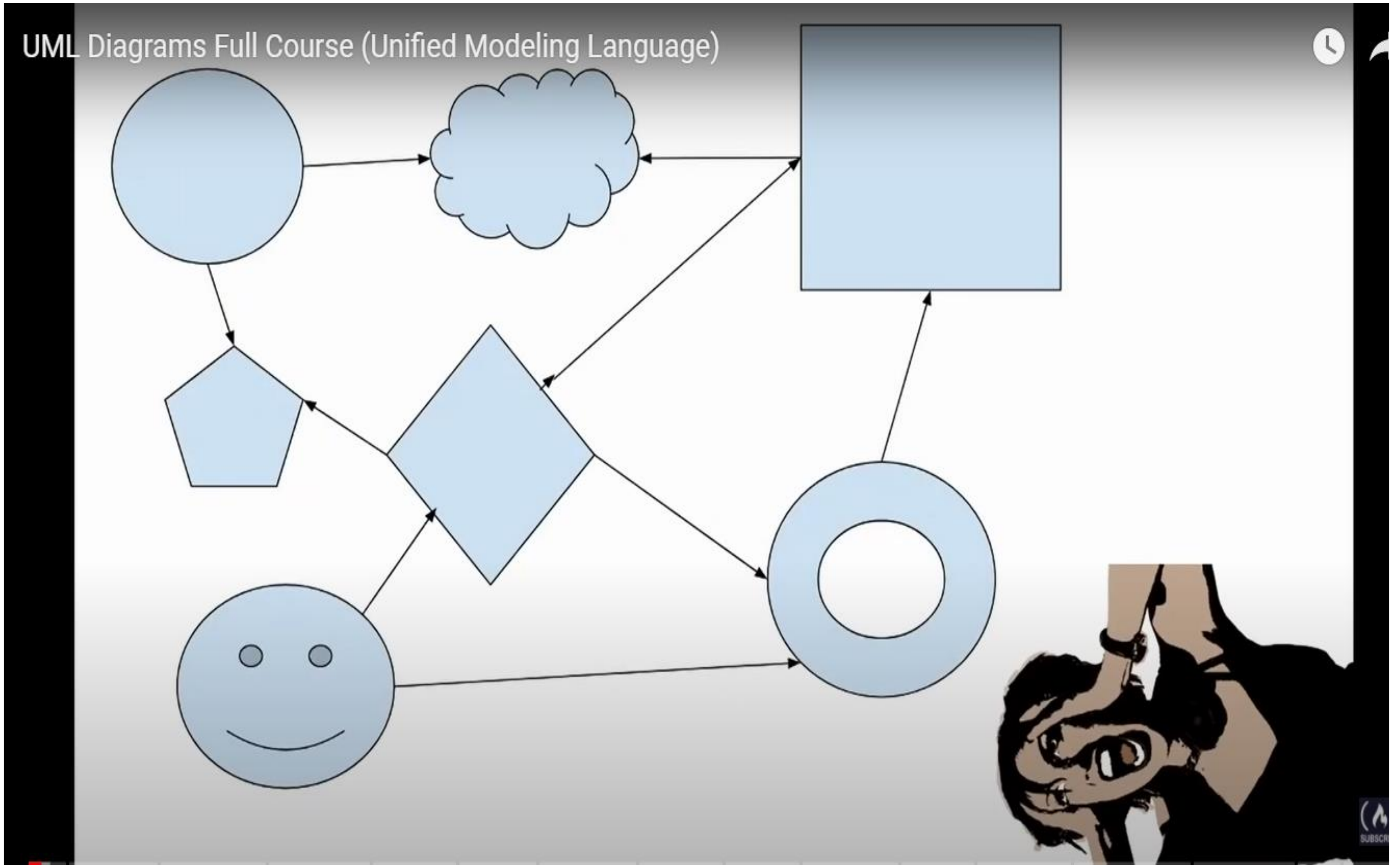
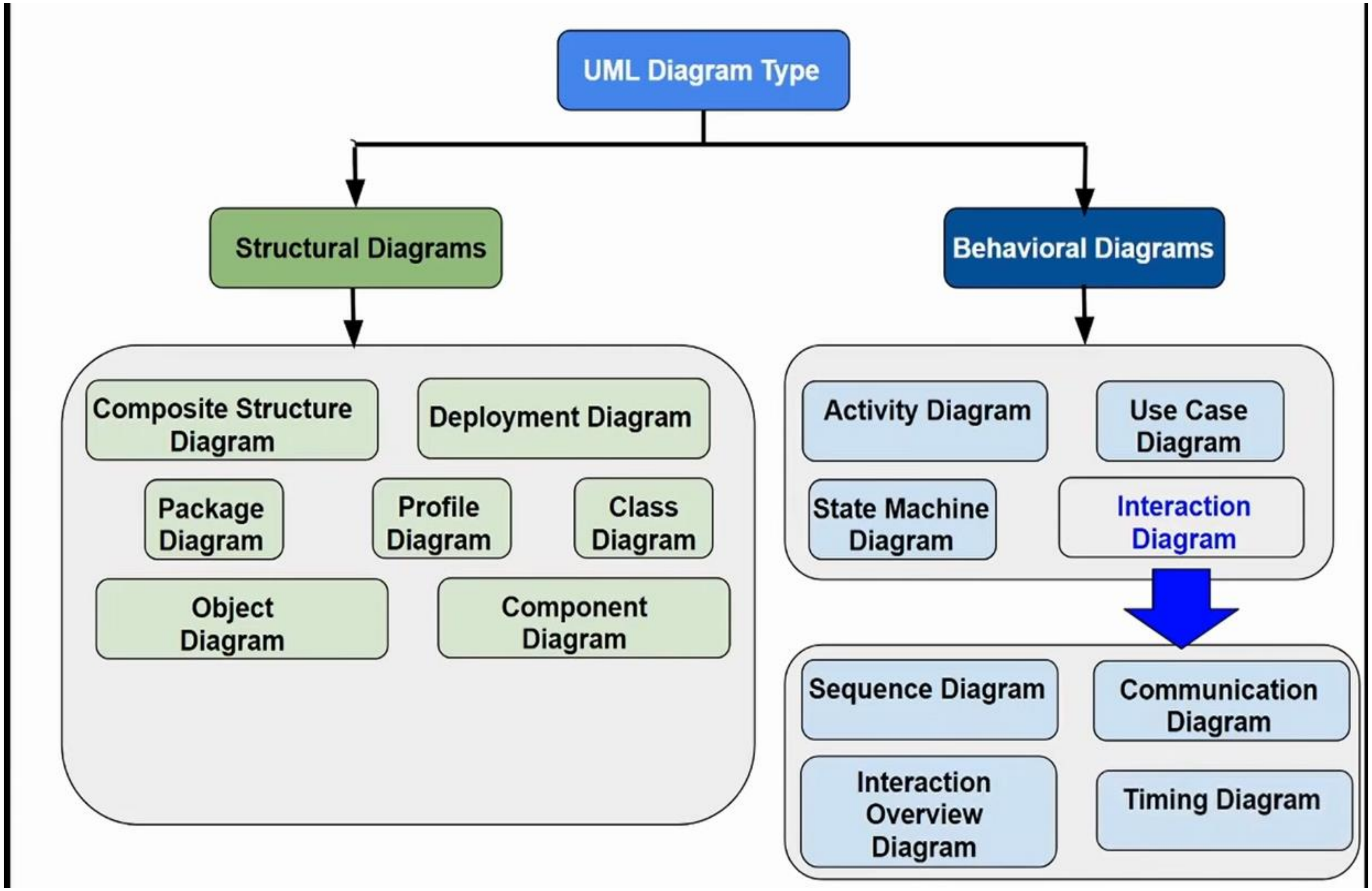
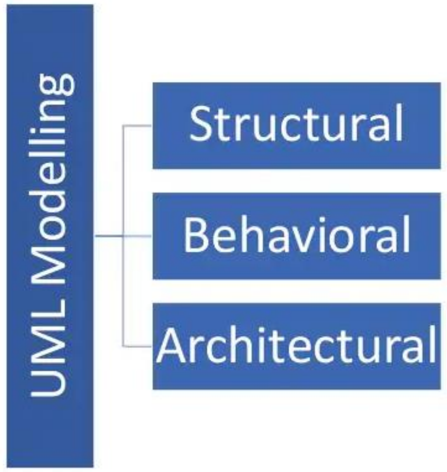
What is UML ?

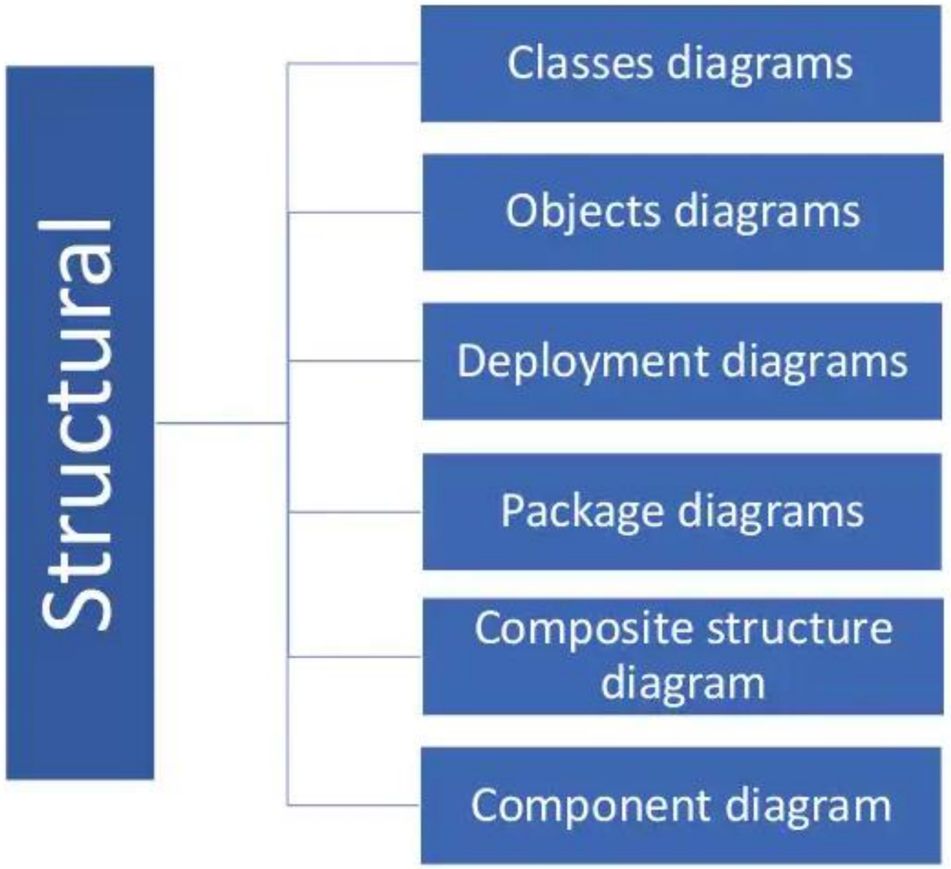
- UML stands for “**Unified Modeling Language**”
- It is a industry-standard **graphical language** for **specifying, visualizing, constructing, and documenting** the artifacts of software systems
- The UML uses mostly **graphical notations to express the OO analysis** and design of software projects.
- **Simplifies the complex process** of software design

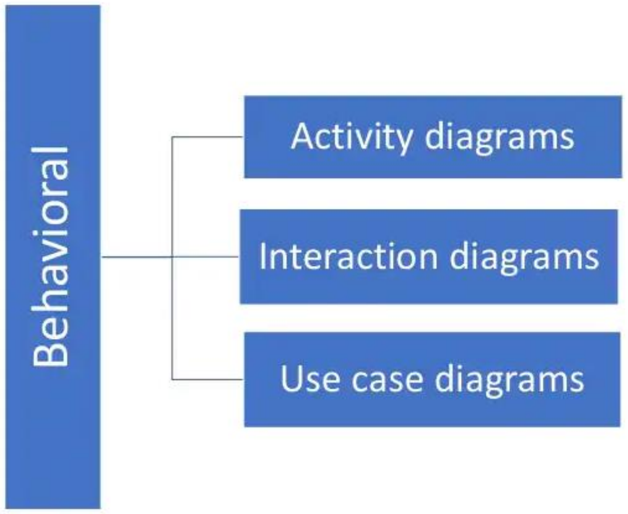

UML Diagrams Full Course (Unified Modeling Language)

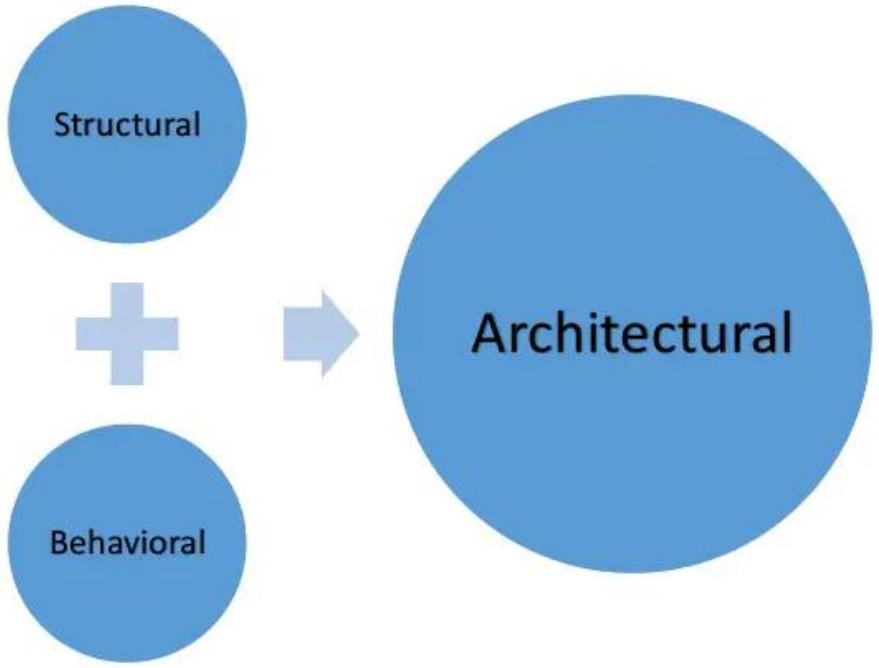

UML Diagram Type
Structural Diagrams
Behavioral Diagrams
Composite Structure Diagram
Deployment Diagram
Package Diagram
Profile Diagram
Class Diagram
Object Diagram
Component Diagram
Activity Diagram
Use Case Diagram
State Machine Diagram
Interaction Diagram
Sequence Diagram
Communication Diagram
Interaction Overview Diagram
Timing Diagram



- Static features of a system.
- Dynamic Feature of a system.
- Blue print of the entire system.


Structural
Classes diagrams
Objects diagrams
Deployment diagrams
Package diagrams
Composite structure diagram
Component diagram


Behavioral
Activity diagrams
Interaction diagrams
Use case diagrams


Structural
Behavioral
Architectural

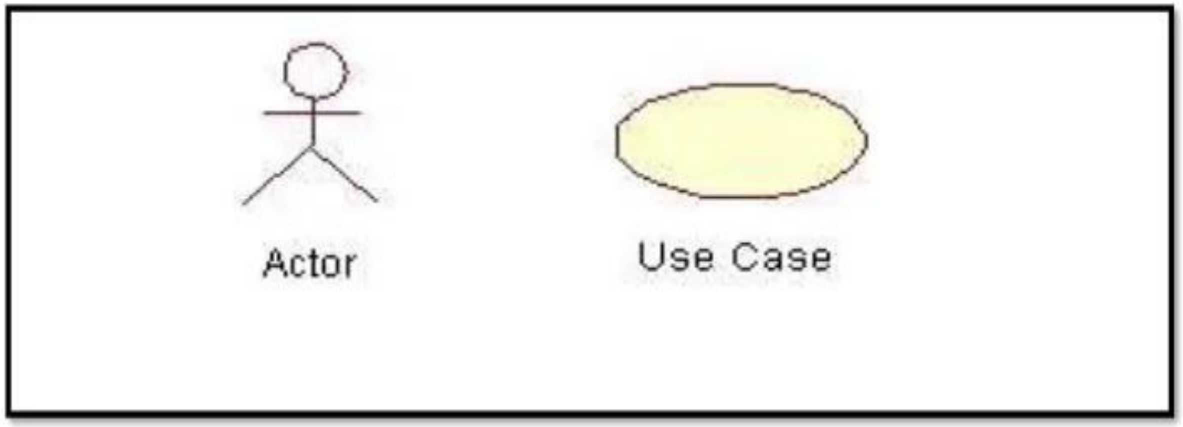
- Analysis (Requirement Engg, SRS)
- Design (Takes Input of Anaysis)
- Architecture of sys is done in detail

Types of UML Diagram

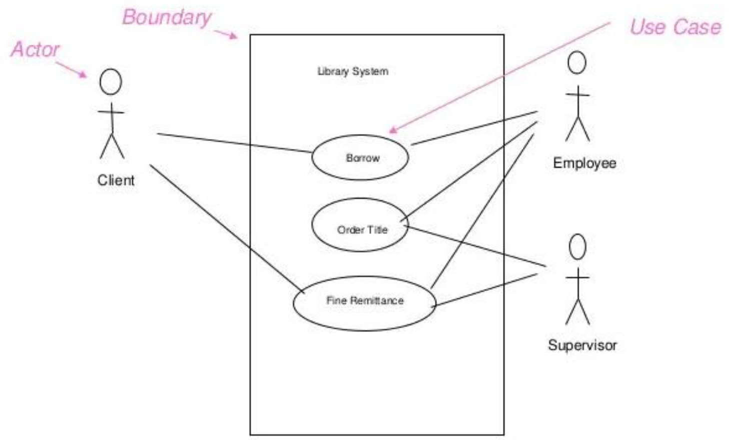
- Use Case Diagram
- Class Diagram
- Sequence Diagram
- Collaboration Diagram
- State Diagram

1. Use Case Diagram

- Used for describing a set of user **scenarios**
- Mainly used for capturing user requirements
- Work like a **contract** between end user and software developers



An Example of Use Case Diagram



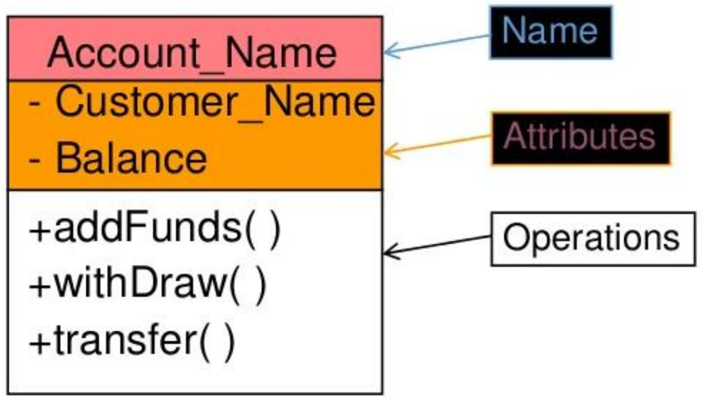
Class Diagram

- Provide a **conceptual model** of the system in terms of entities and their relationships
- Used for requirement capture, **end-user interaction.**
- Detailed class diagrams are **used for developers**.

Class Representation

- Each class is represented by a rectangle subdivided into three compartments
  - Name
  - Attributes
  - Operations
- Modifiers are used to indicate visibility of attributes and operations.
  - '+' is used to denote *Public* visibility (everyone)
  - '#' is used to denote *Protected* visibility (friends and derived)
  - '-' is used to denote *Private* visibility (no one)
- By default, attributes are hidden and operations are visible.

An example of Class



Interaction Diagram

1. The purposes of interaction diagrams are to

   - **visualize the interactive behavior** of the system.

2. Visualizing **interaction is a difficult task**.

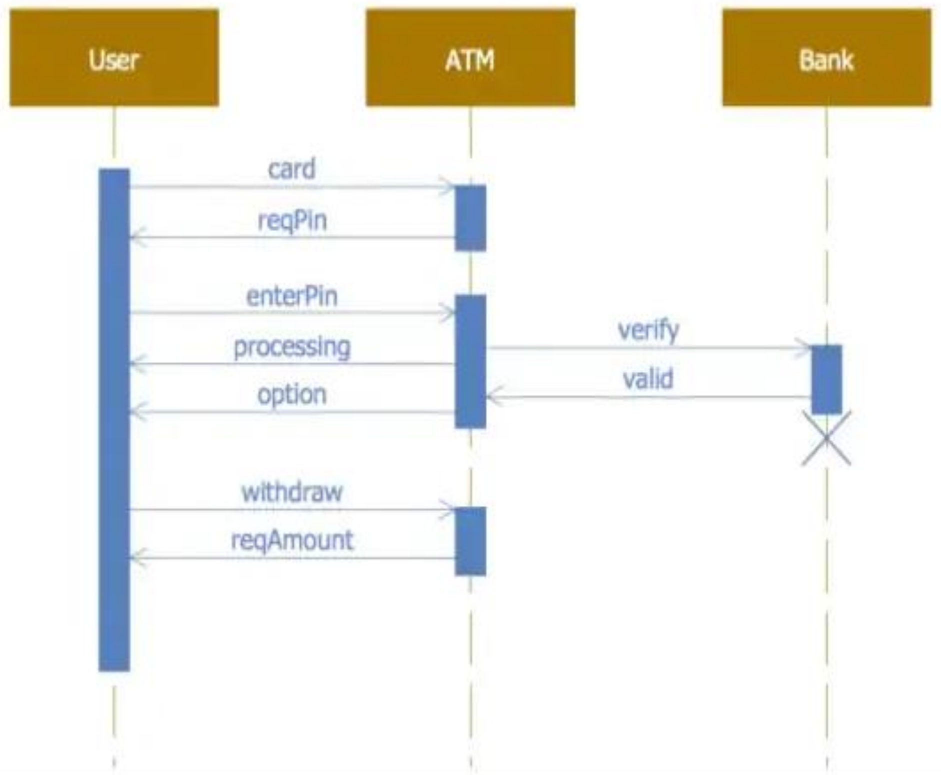
The solution is to use different types of models to capture the different aspects of the interaction.

a. **Sequence Diagram.**

b. **Collaboration Diagram**

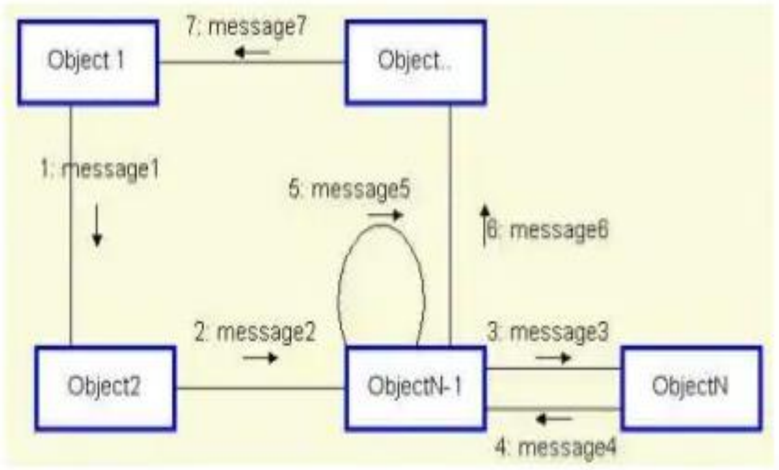
Interaction Diagram : Sequence Diagram

A Sequence diagram is an interaction diagram that shows **how processes operate with one another and in what order.**



Interaction Diagrams: Collaboration diagrams

1. Shows the **relationship** between **objects** and the **order of messages passed** between them.
2. The **objects** are listed as **rectangles** and **arrows indicate** the **messages being passed.**
3. The **numbers next to the messages** are called **sequence numbers**.
4. convey the same information as sequence diagrams, but **focus on object roles instead of the time sequence.**

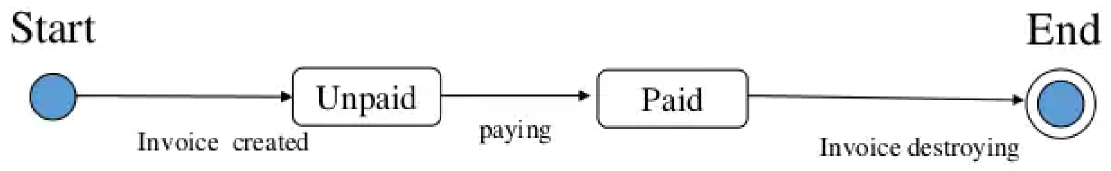


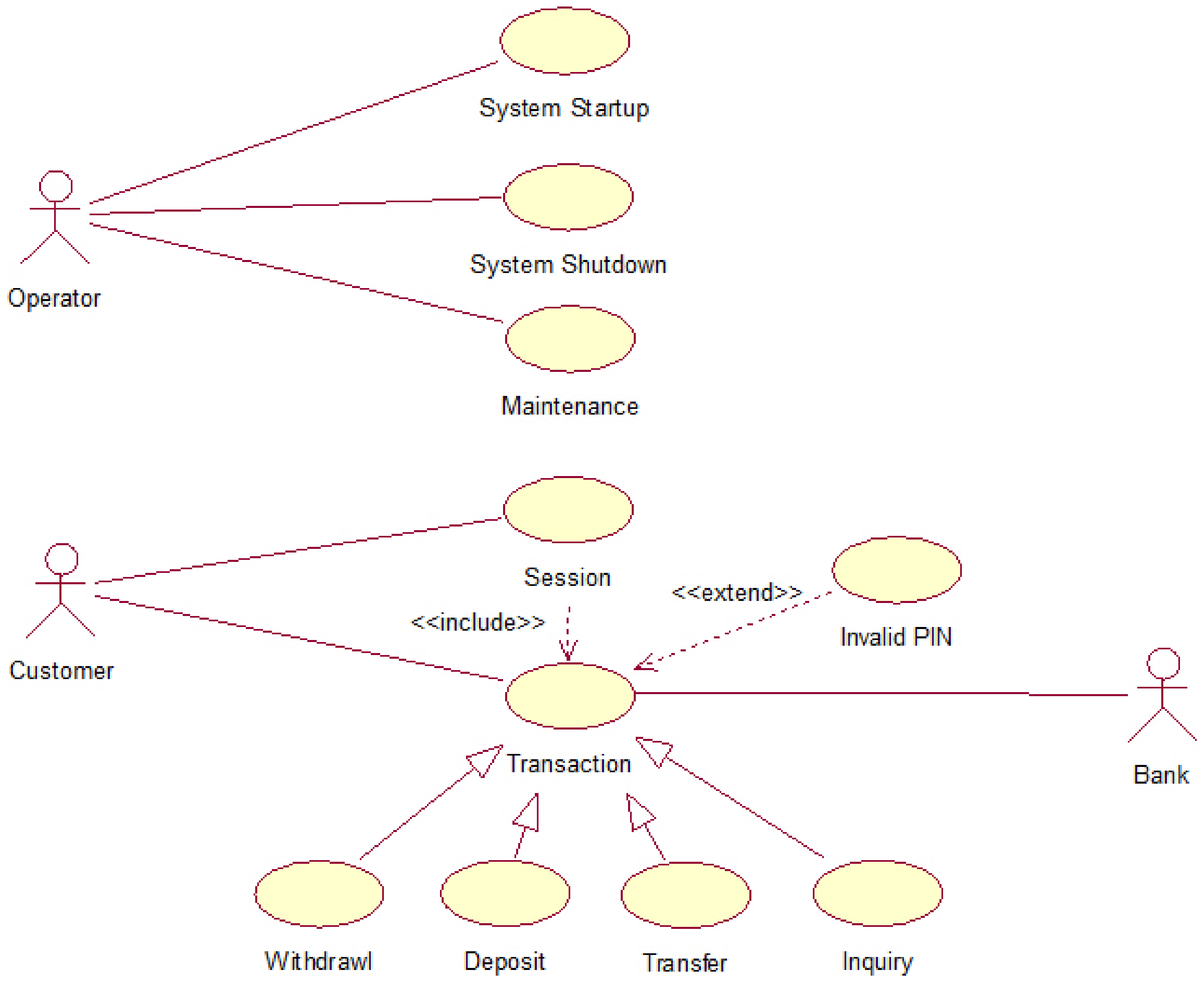
Tools for UML

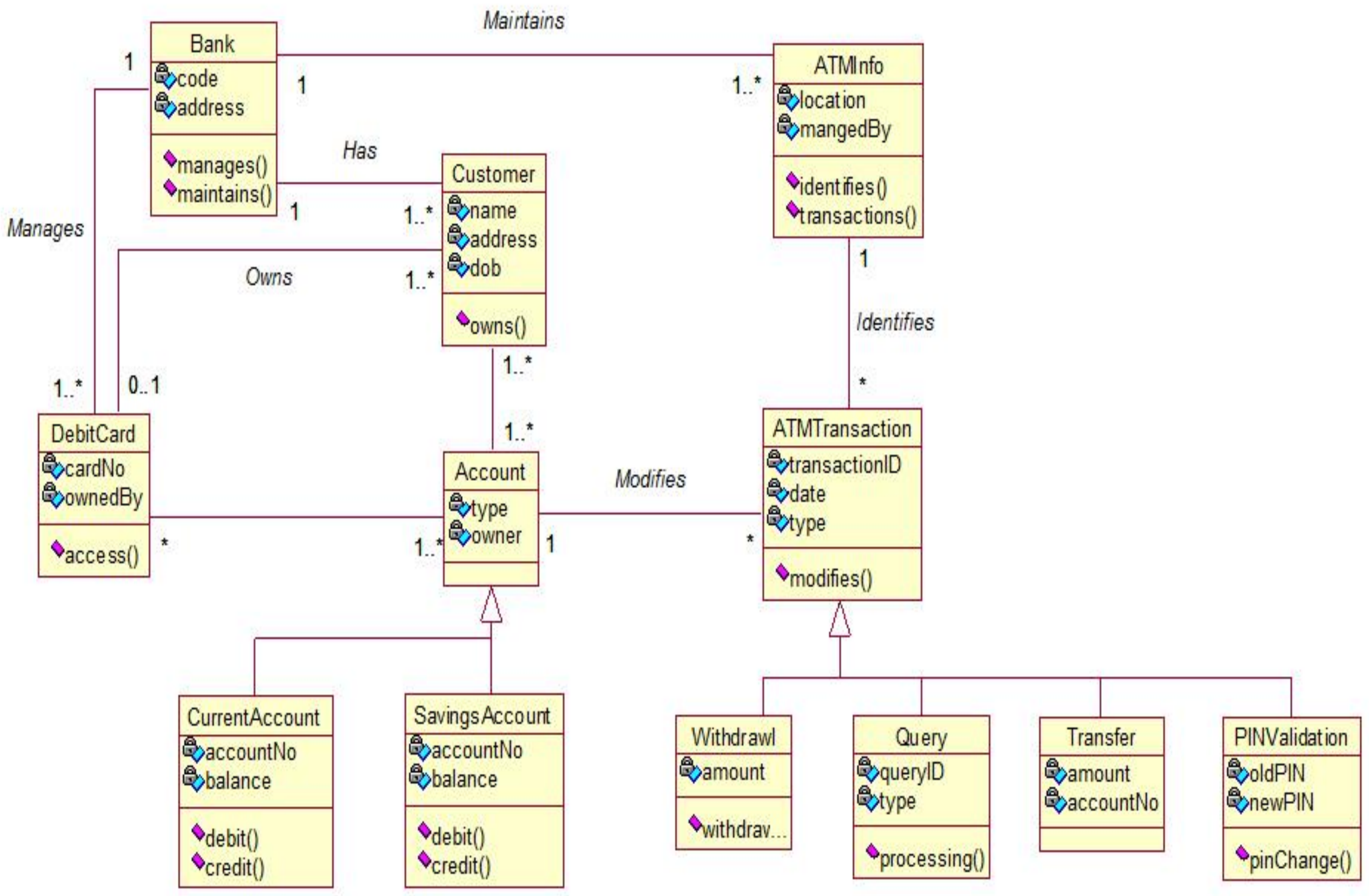
- Star UML
- Rational Rose
- Lucidchart

State Diagrams (Billing Example)

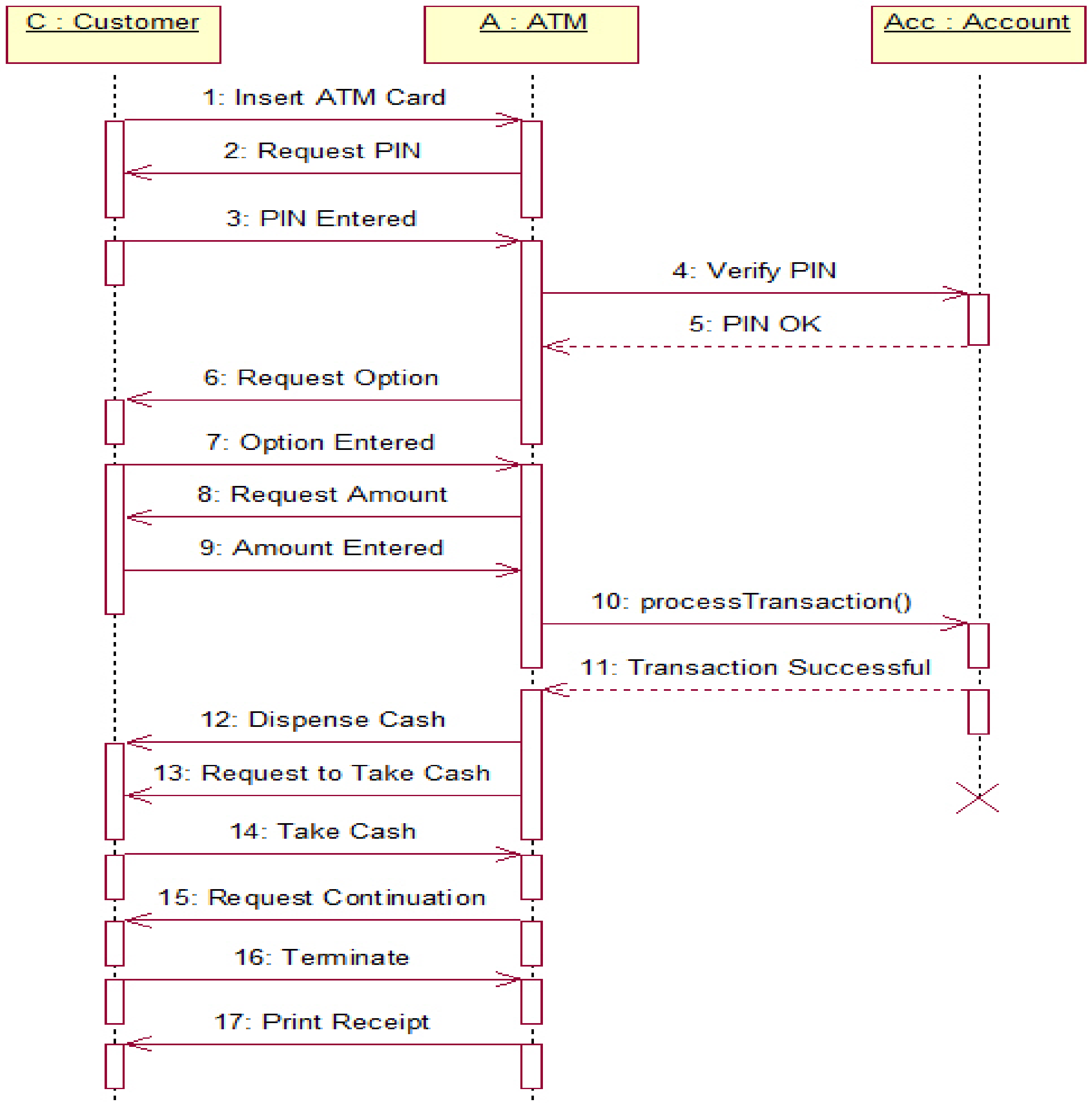
State Diagrams show the **sequences of states an object goes through during its life cycle in response to stimuli**, together with its responses and actions; an abstraction of all possible behaviors.

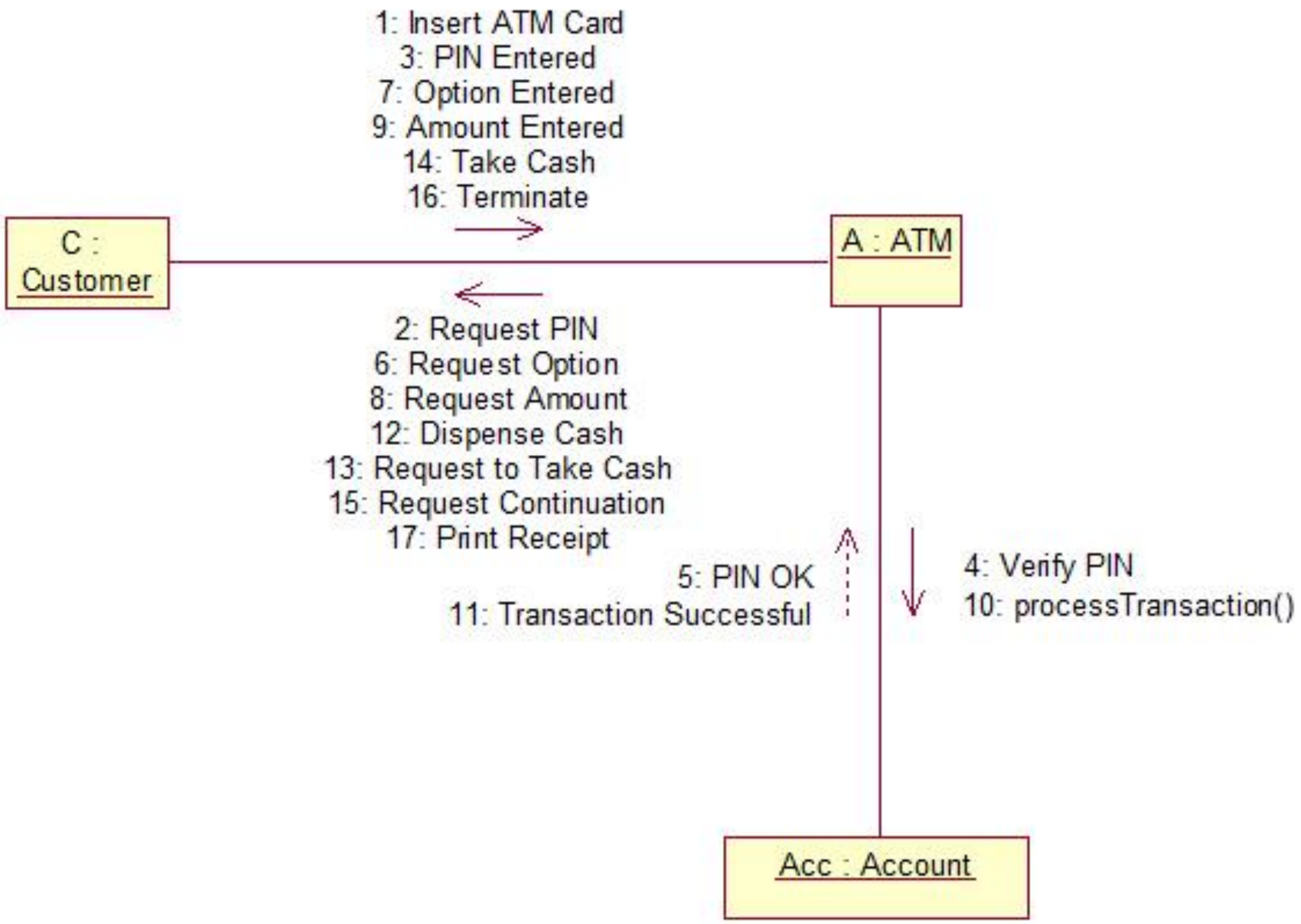



System Startup
Operator
System Shutdown
Maintenance
Session
Customer
<<extend>>
Invalid PIN
<<include>>
Transaction
Bank
Withdrawl
Deposit
Transfer
Inquiry

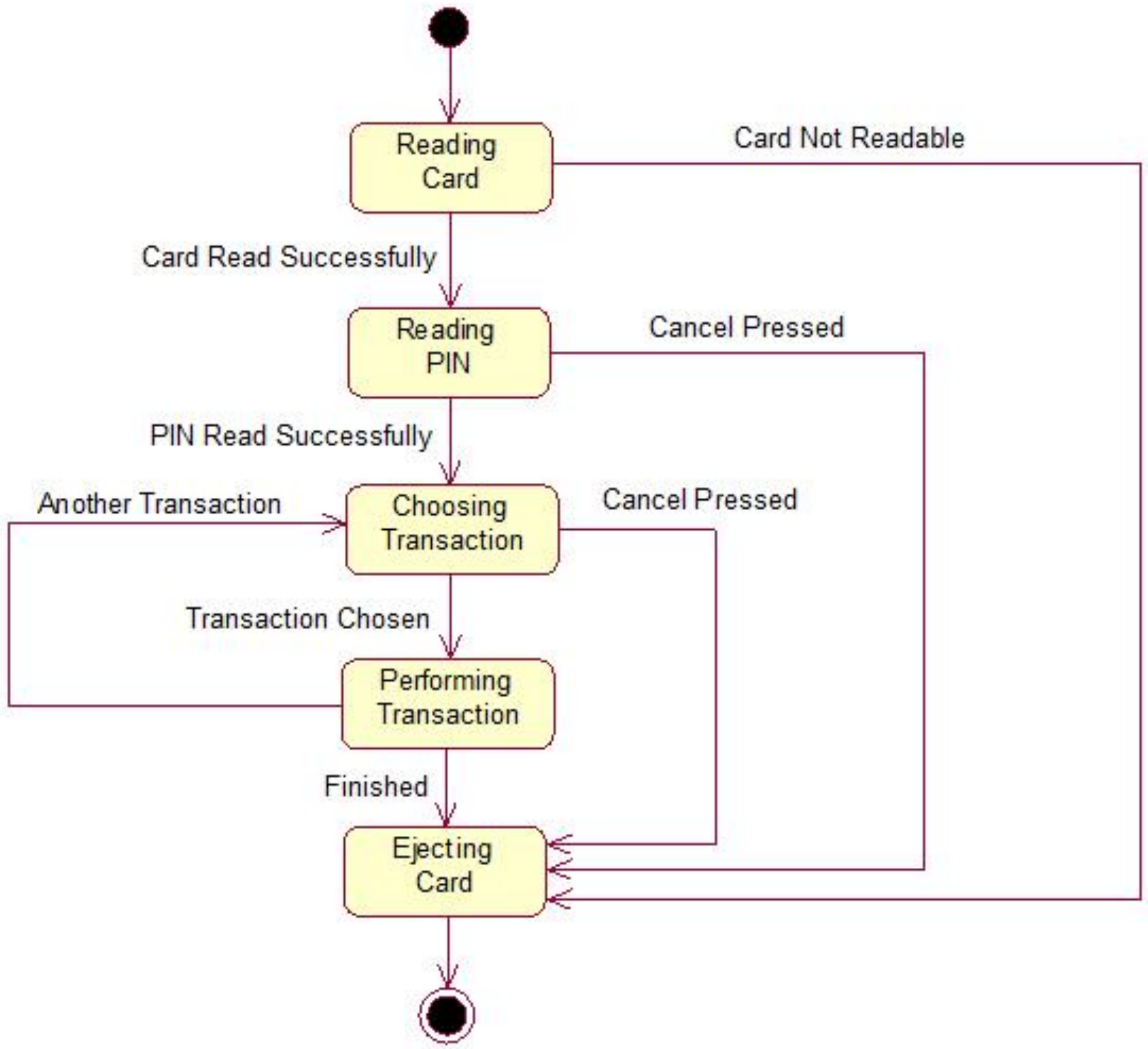


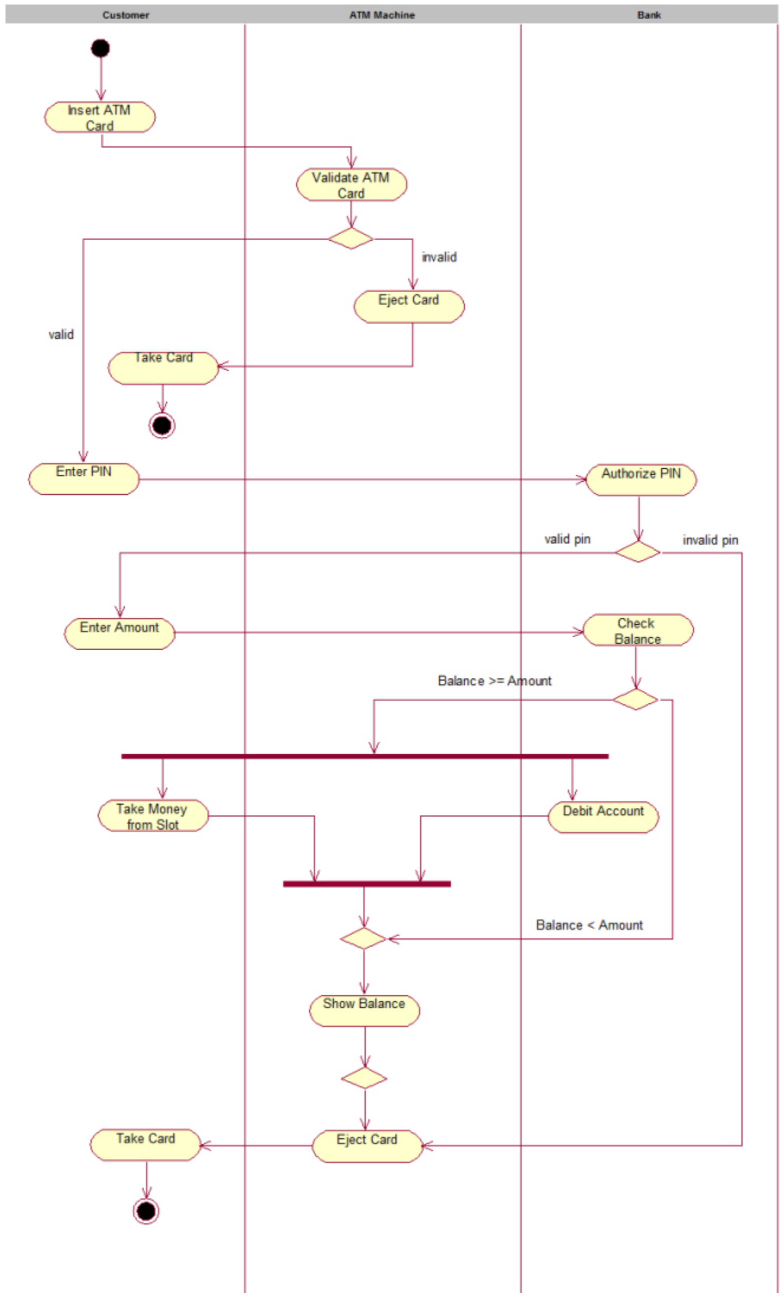
Maintains
Bank
code
address
manages()
maintains()
1
1
Has
1
1..*
ATMInfo
location
mangedBy
identifies()
transactions()
1..*
Customer
name
address
dob
owns()
Manages
Owns
1..*
1..*
1
Identifies
*
1..*
0..1
DebitCard
cardNo
ownedBy
access()
*
1..*
Account
type
owner
1..*
1
Modifies
*
ATMTransaction
transactionID
date
type
modifies()
CurrentAccount
accountNo
balance
debit()
credit()
SavingsAccount
accountNo
balance
debit()
credit()
Withdrawl
amount
withdrav...
Query
queryID
type
processing()
Transfer
amount
accountNo
PINValidation
oldPIN
newPIN
pinChange()


C : Customer
A : ATM
Acc : Account
1: Insert ATM Card
2: Request PIN
3: PIN Entered
4: Verify PIN
5: PIN OK
6: Request Option
7: Option Entered
8: Request Amount
9: Amount Entered
10: processTransaction()
11: Transaction Successful
12: Dispense Cash
13: Request to Take Cash
14: Take Cash
15: Request Continuation
16: Terminate
17: Print Receipt

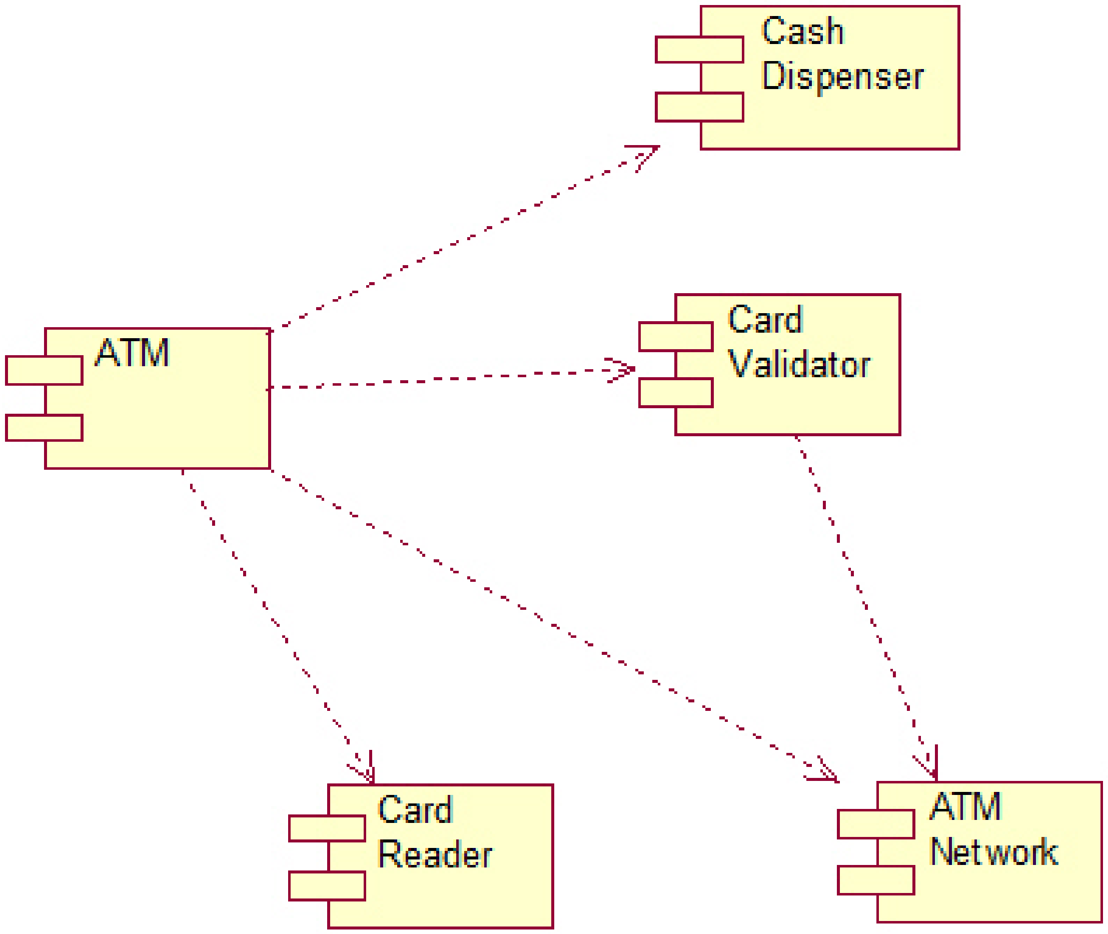

1: Insert ATM Card
3: PIN Entered
7: Option Entered
9: Amount Entered
14: Take Cash
16: Terminate
C : Customer
A : ATM
2: Request PIN
6: Request Option
8: Request Amount
12: Dispense Cash
13: Request to Take Cash
15: Request Continuation
17: Print Receipt
5: PIN OK
11: Transaction Successful
4: Verify PIN
10: processTransaction()
Acc : Account

State Chart Diagram

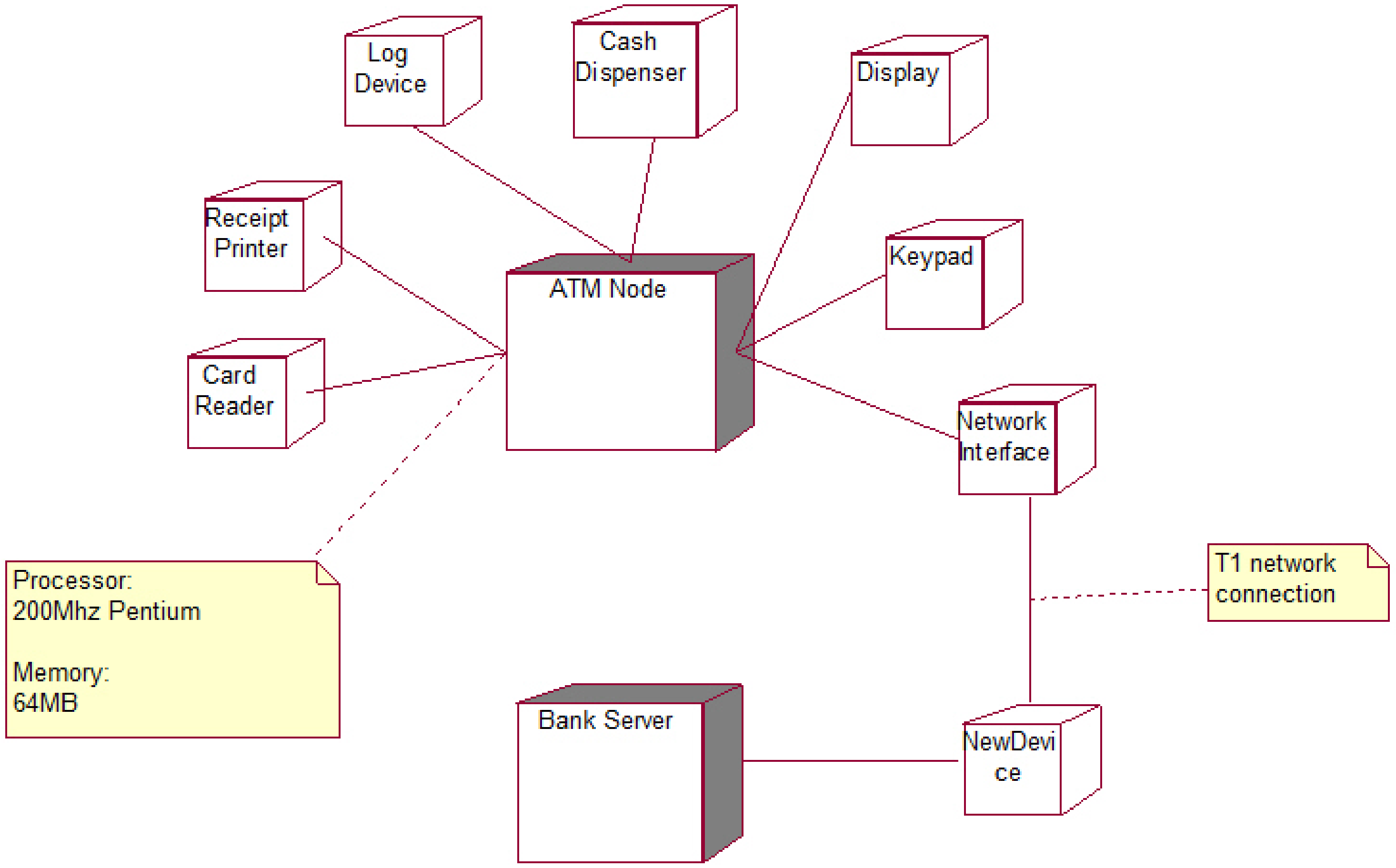



Customer
ATM Machine
Bank
Insert ATM Card
Validate ATM Card
invalid
Eject Card
valid
Take Card
Enter PIN
Authorize PIN
valid pin
invalid pin
Enter Amount
Check Balance
Balance >= Amount
Take Money from Slot
Debit Account
Balance < Amount
Show Balance
Take Card
Eject Card

component



Deployment


Log Device
Cash Dispenser
Display
Receipt Printer
Keypad
ATM Node
Card Reader
Network Interface
T1 network connection
Processor:
200Mhz Pentium
Memory:
64MB
Bank Server
NewDevi ce