Unit II

Software Requirements: Functional and non-functional requirements, User requirements, System requirements, Interface specification, the software requirements document.

Requirements engineering process: Feasibility studies, Requirements elicitation and analysis, Requirements validation, Requirements management**.**

System models: Context Models, Behavioral models, Data models, Object models, structured methods. UML Diagrams.

Software Requirements

Gathering software requirements is the foundation of entire software development project. Hence they must be clear , correct and well defined.

We should try to understand what sort of requirements may arise in the requirements elicitation phase and what kinds of requirements are expected from the software system .

Categorization :

- Functional Requirements
- Non functional Requirements

Functional Requirements

- Requirements which are related to functional aspect of software fall into this category .
- They define functions and functionality within and from the software system.

Examples:

- Search option given to user to search from various invoices .
- User should be able to mail any report to management.
- User can be divided into groups and groups can be given separate rights.
- Software should be developed keeping downward compatibility intact.

Non functional Requirements

Requirements which are not related to functional aspect of software, fall into this category. They are implicit or expected characteristics of software which users make assumption of.

Non functional requirements includes,

- Security
- Storage
- Configuration
- Performance
- cost

Requirements are categorized logically as:

- **Must have**: software cannot have be said operational without them.
- **Should have**: Enhancing the functionality of software
- **Could have**: software can still properly function with these requirements.
- **Wish list**: These requirements do not map to any objectives of software.

  While developing software , “must have “ must be implemented , “should have” is a matter of debates with stakeholders and cloud have and wish list can be kept for software updates.

Examples

Library Management System

Functional requirements:

- Issue a book
- Return a book
- Searching a book
- Membership cards
- Recording a data

Non functional requirements:

Security

Usability

Availability

Banking System

Think what are the

- Functional requirements &
- Non functional requirements for Banking system.

Difference between non functional and functional requirements

Parameters	Functional Requirement	Non-Functional Requirement
What it is	Verb	Attributes
Requirement	It is mandatory	It is non-mandatory
Capturing type	It is captured in use case.	It is captured as a quality attribute.
End-result	Product feature	Product properties
Capturing	Easy to capture	Hard to capture
Objective	Helps you verify the functionality of the software.	Helps you to verify the performance of the software.
Area of focus	Focus on user requirement	Concentrates on the user's expectation.
Documentation	Describe what the product does	Describes how the product works
Type of Testing	Functional Testing like System, Integration, End to End, API testing, etc.	Non-Functional Testing like Performance, Stress, Usability, Security testing, etc.
Test Execution	Test Execution is done before non-functional testing.	After the functional testing
Product Info	Product Features	Product Properties

Software Requirements Specification (SRS)

It is the official statement of what is required of the system developers. Should include both a definition of user requirements and a specification of the system requirements. It is NOT a design document. As far as possible ,it should set of WHAT the system should do rather than HOW it should do it

Purpose of SRS

- communication between the Customer, Analyst, system developers, maintainers
- firm foundation for the design phase
- support system testing activities
- Support project management and control
- controlling the evolution of the system

Goal of SRS:

- Feedback to the customer.
- designing decompositions.
- input to design specification.
- production validation check.

Quality of SRS

- Correct
- Complete
- Modifiable
- verifiable

IEEE Standard - SRS

1.Introduction
- 1.1 Purpose
- 1.2 Scope
- 1.3 Definitions, Acronyms and Abbreviations
- 1.4 References
- 1.5 Overview

2. General description
- 2.1 Product perspective
- 2.2 Product function summary
- 2.3 User characteristics
- 2.4 General constraints
- 2.5 Assumptions and dependencies

IEEE Standard - SRS

3. Specific Requirements
   Functional requirements
   External interface requirements
      - Performance requirements
      - Design constraints
      - Attributes

eg. security, availability, maintainability,
   Transferability/Conversion
   Other requirements

- Appendices
- Index

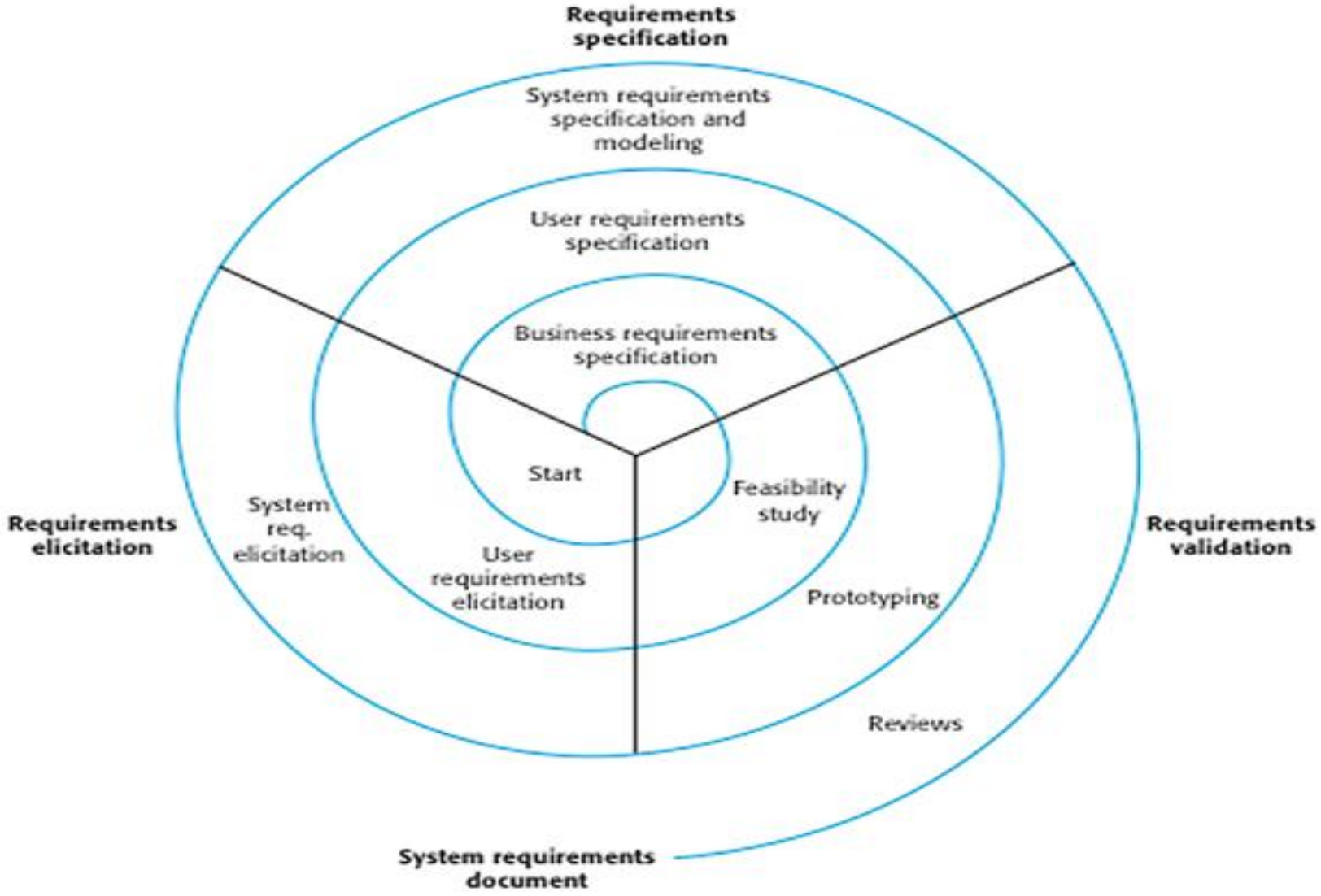
REQUIREMENTS ENGINEERING PROCESS

REQUIREMENTS ENGINEERING PROCESS

To create and maintain a system requirement document

1. **Feasibility study** --Concerned with assessing whether the system is useful to the business
2. **Elicitation and analysis** --Discovering requirements
3. **Specifications** --Converting the requirements into a standard form
4. **Validation --** Checking that the requirements actually define the syste m that the customer wants

SPIRAL REPRESENTATION



FEASIBILITY STUDIES

Starting point of the requirements engineering process

Input: Set of preliminary business requirements, an outline description of the system and how the system is intended to support business processes

Output: Feasibility report that recommends whether or not it is worth carrying out further .

Feasibility report answers a number of questions:

1. Does the system contribute to the overall objective
2. Can the system be implemented using the current technology and within given cost and schedule
3. Can the system be integrated with other system which are already in place.

REQUIREMENTS ELICITATION ANALYSIS

Involves a number of people in an organization.

This process is very difficult.

Reasons :

1. Stakeholder often don't know what they want from the computer system.
2. Stakeholder expression of requirements in natural language is sometimes difficult to understand.
3. Different stakeholders express requirements differently
4. Influences of political factors Change in requirements due to dynamic environments.

REQUIREMENTS ELICITATION ANALYSIS

Process activities

1. **Requirement Discovery** -- Interaction with stakeholder
2. **Requirements classification and organization** – dealt with unstructured collection of requirements
3. **Requirements prioritization and negotiation**
4. **Requirements documentation** -- Requirements be documented and placed in the next round of spiral

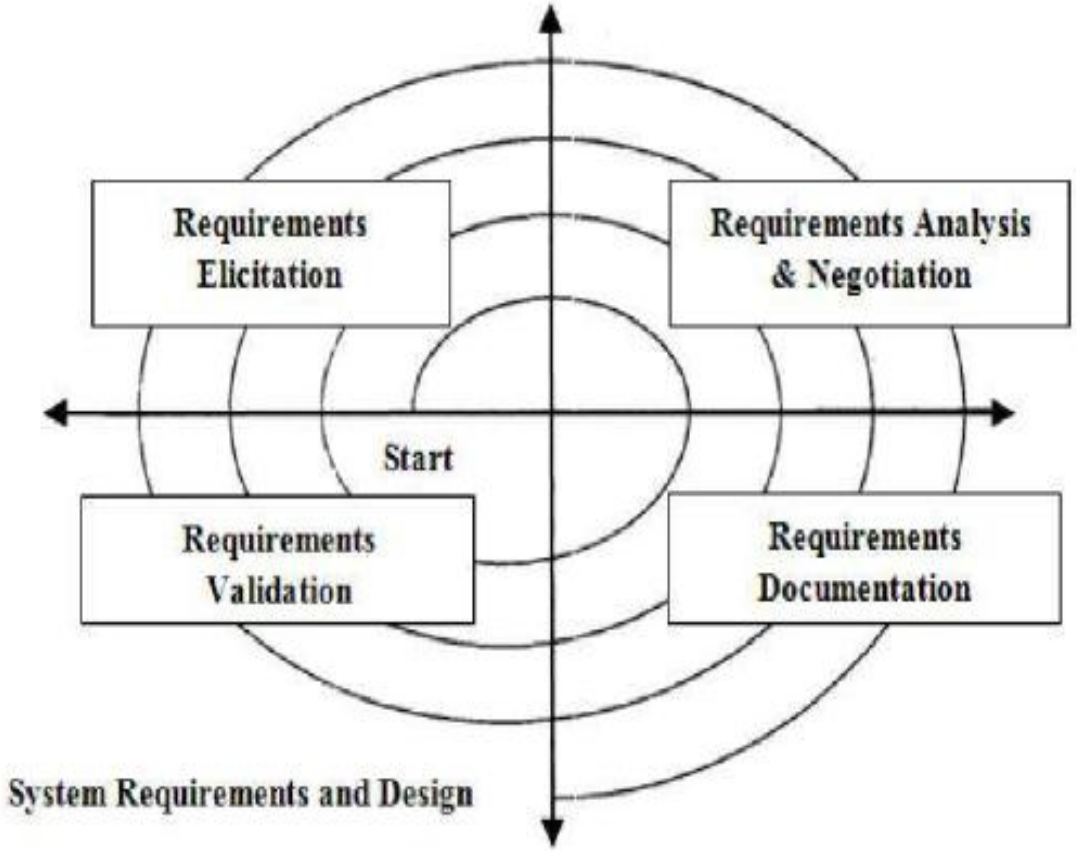


Figure 4: From the work of Kotonya and Sommerville.

REQUIREMENTS VALIDATION

Validation checks

1.Validity checks --Verification that the system performs the intended function by the user

2.Consistency check --Requirements should not conflict

3. Completeness checks --Includes requirements which define all functions and constraints intended by the system user

4. Realism checks --Ensures that the requirements can be actually implemented

5. Verifiability -- Testable to avoid disputes between customer and developer.

Requirements change management

Consists of three principal stages:

1. Problem analysis and change specification--Process starts with a specific change proposal and analyzed to verify that it is valid
2. Change analysis and costing--Impact analysis in terms of cost, time and risks
3. Change implementation--Carrying out the changes in requirements document, system design and its implementation

System Models

System Models

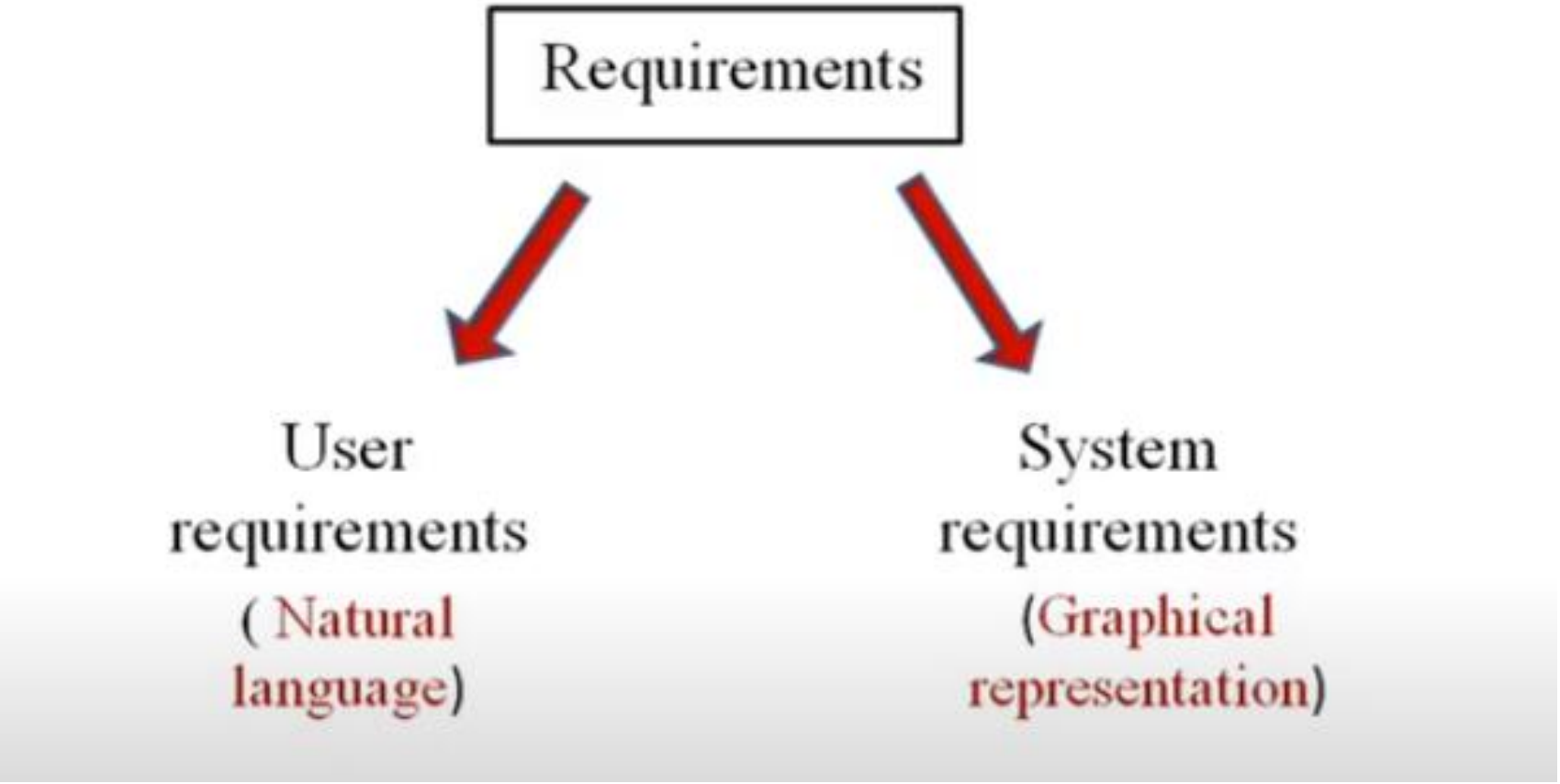
Definition :

It is a abstract view of the system that ignores system details.

A model is graphical representation that describes the system which is to be developed.

Natural language is used to write both user and system requirements.

Requirements



Types of system models

1.Context models

2. Behavioral models

3.Data models

4.Object models

5.Structured models

1.Context models

A type of Architectural Model.

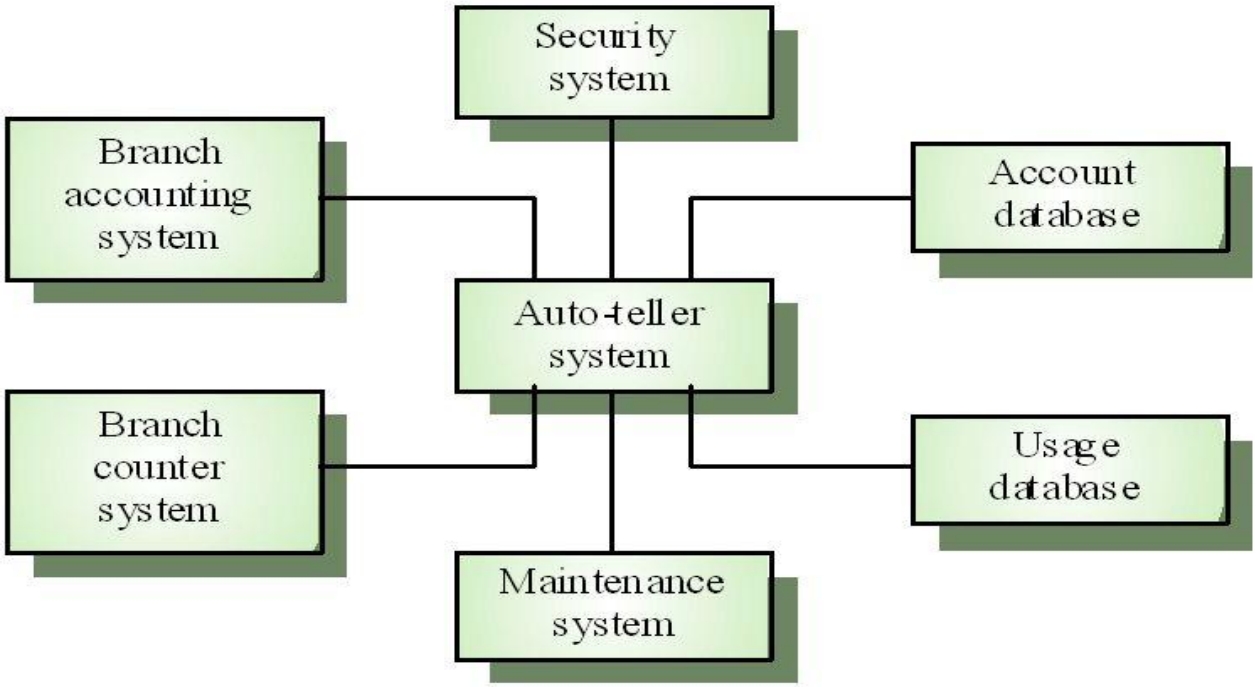
Consists of sub-systems that make up an entire system .

Steps:

1. To identify the subsystem.
2. Represent the high level architectural model as simple block diagram.
3. Depict each sub system a named rectangle.
4. Lines between rectangles indicate associations between subsystems.

Context Model – ATM Machine

Context model



2. Behavioural models

Describes the overall behaviour of a system.

Two types of behavioural model

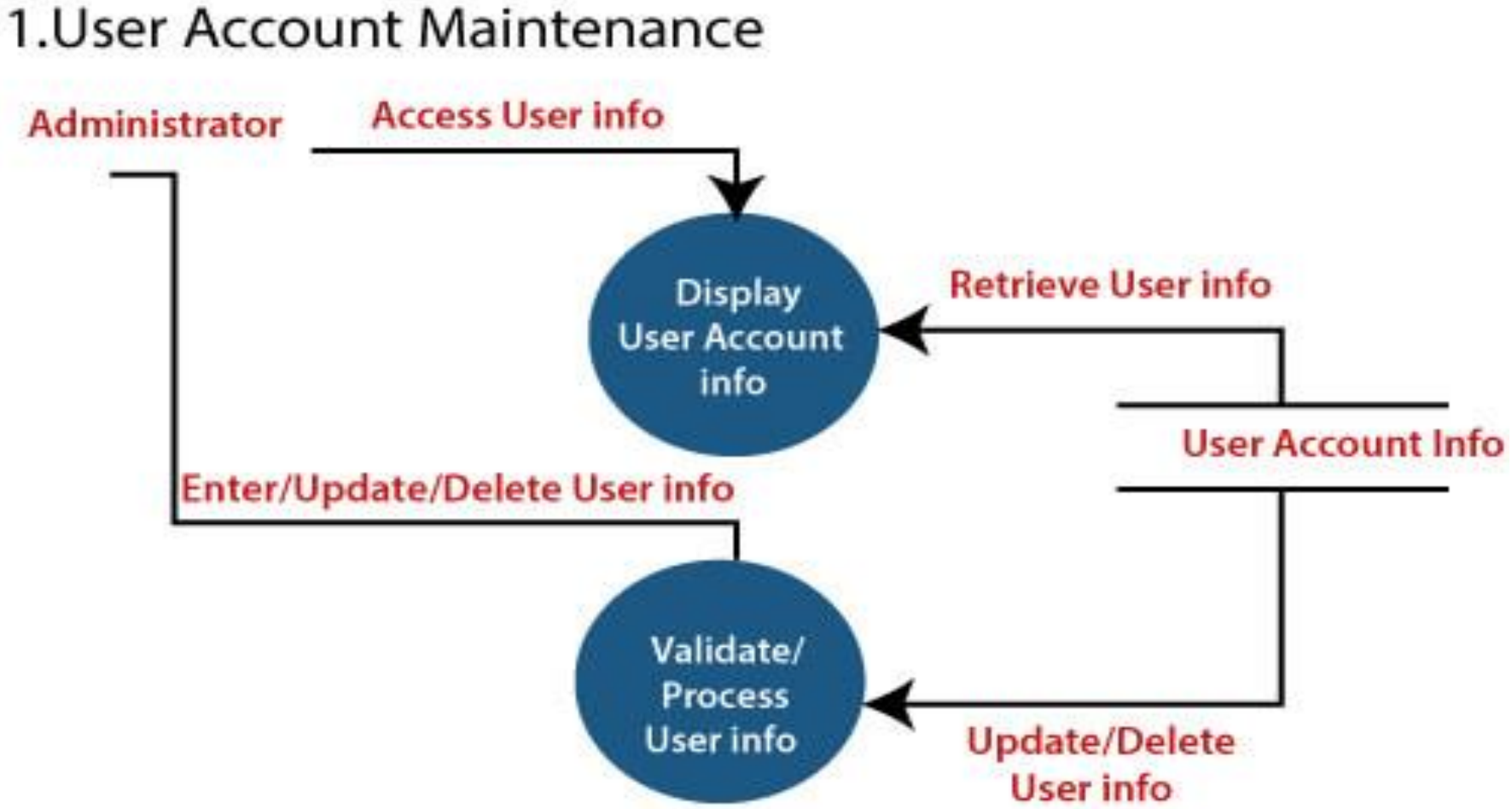
1.Data Flow models

2.State machine models

Data flow models

- Concentrate on the flow of data and functional transformation on that data.
- Show the processing of data and its flow through a sequence of processing steps.

Data Flow Models

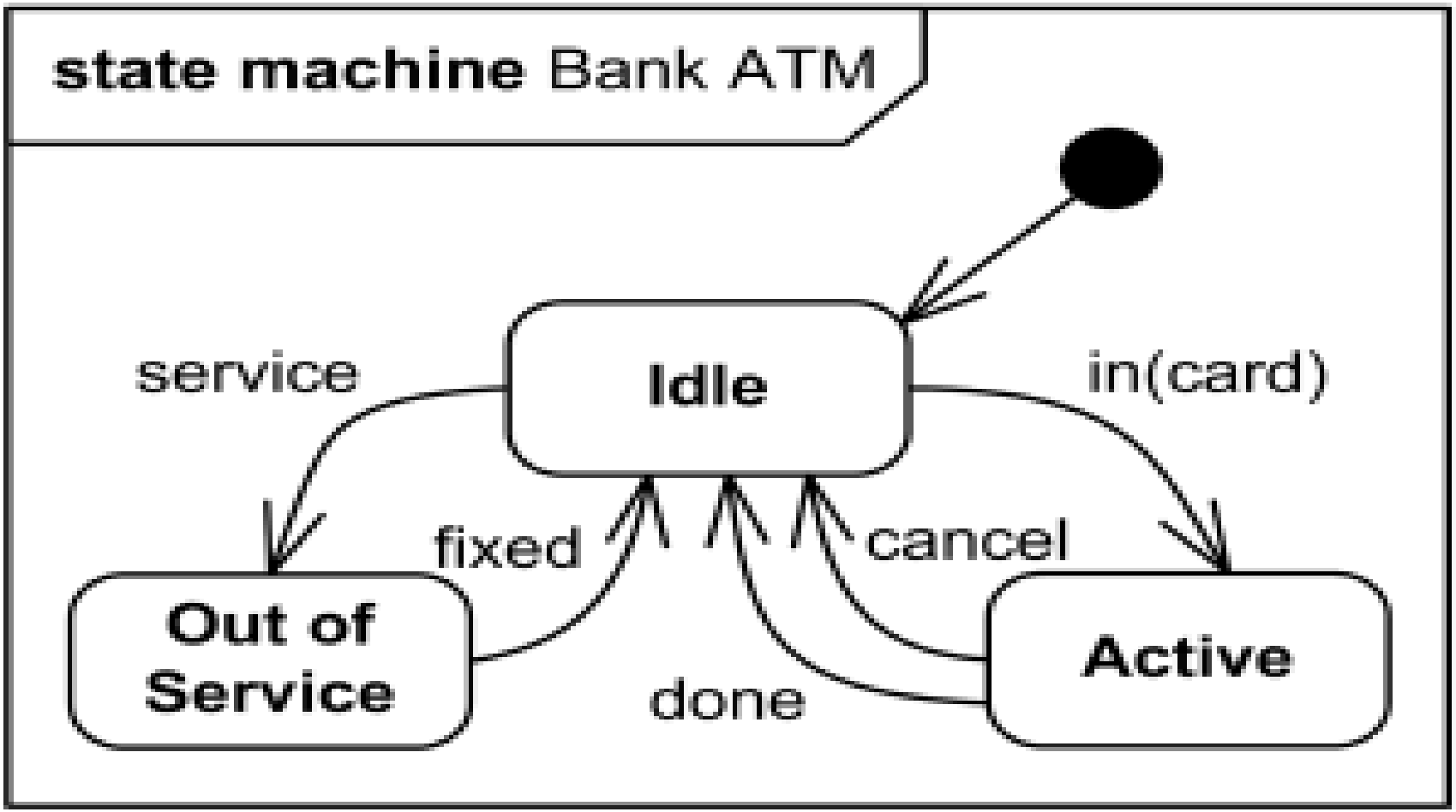


State Machine models

- Describe how a system responds to internal or external events .
- Shows system states and events that cause transition from one state to another.
- Does not show the flow of data within the system.
- Used for modeling of real time systems .

[e.g : Microwave Oven, washing machine]

State Machine models



3.Data models

- Describe the logical structure of data.
- An entity-relation attribute model.
- Widely used in database design
- No specific notation provided in the UML but objects and associations can be used.
- Defines all data object, the relationships between the data object.

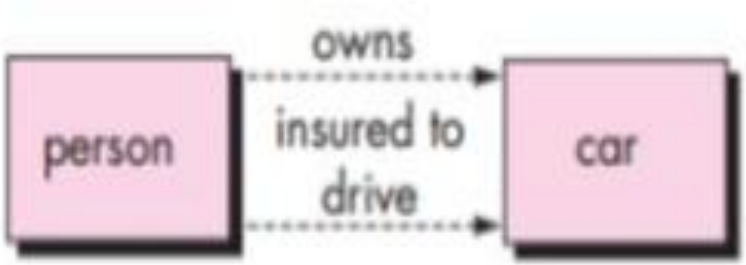
Data objects

- Composite information which is processed by software.
- External Entity. (a thing, a report, an occurrence, a role, an organizational unit , a place .)
- Car : data object

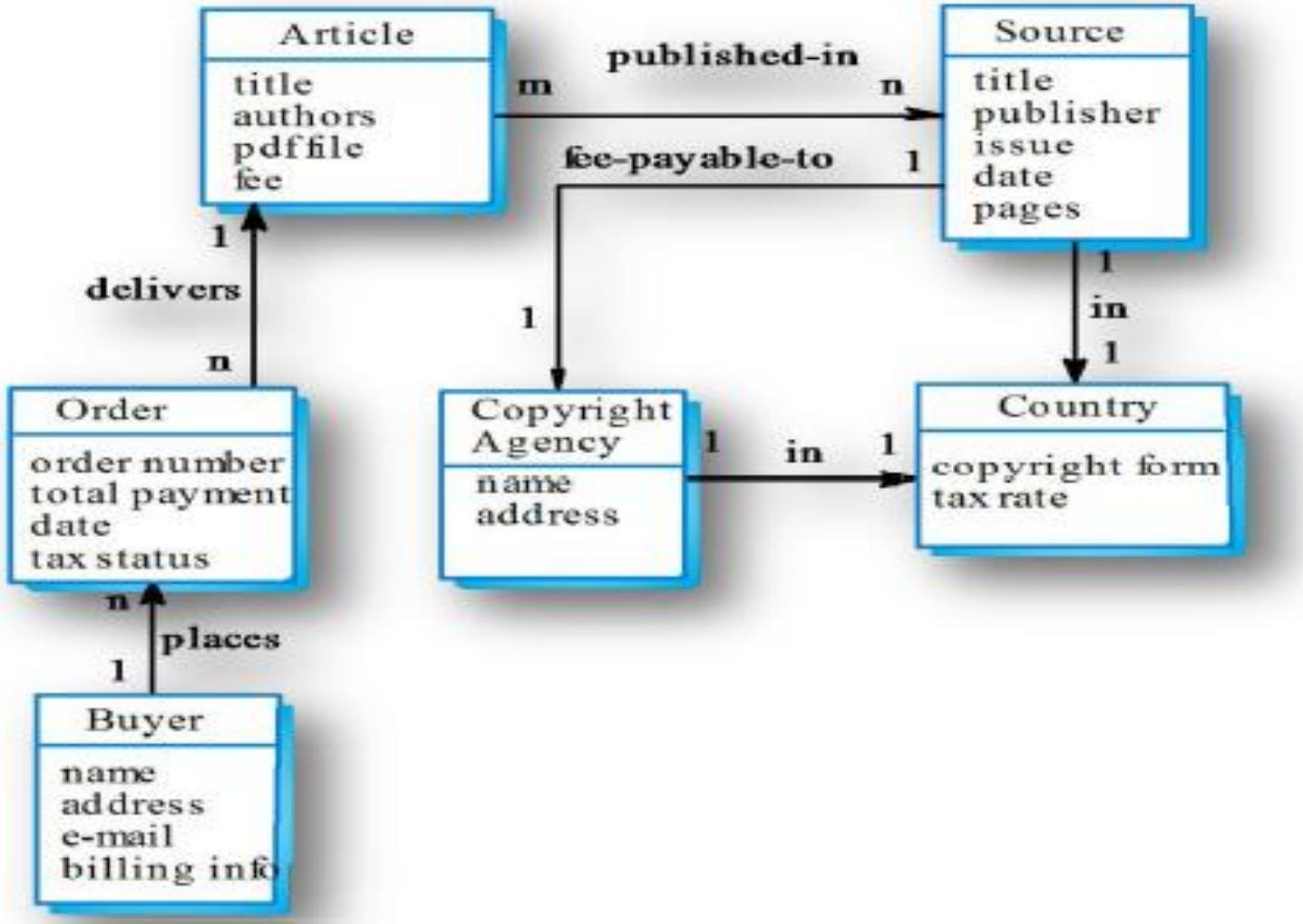
  e.g : BMW , 750iL XZ765 white ABC (Attributes)

Relationships

- A person owns a car.
- A person is insured to drive a car.



Library semantic model



4.Object models

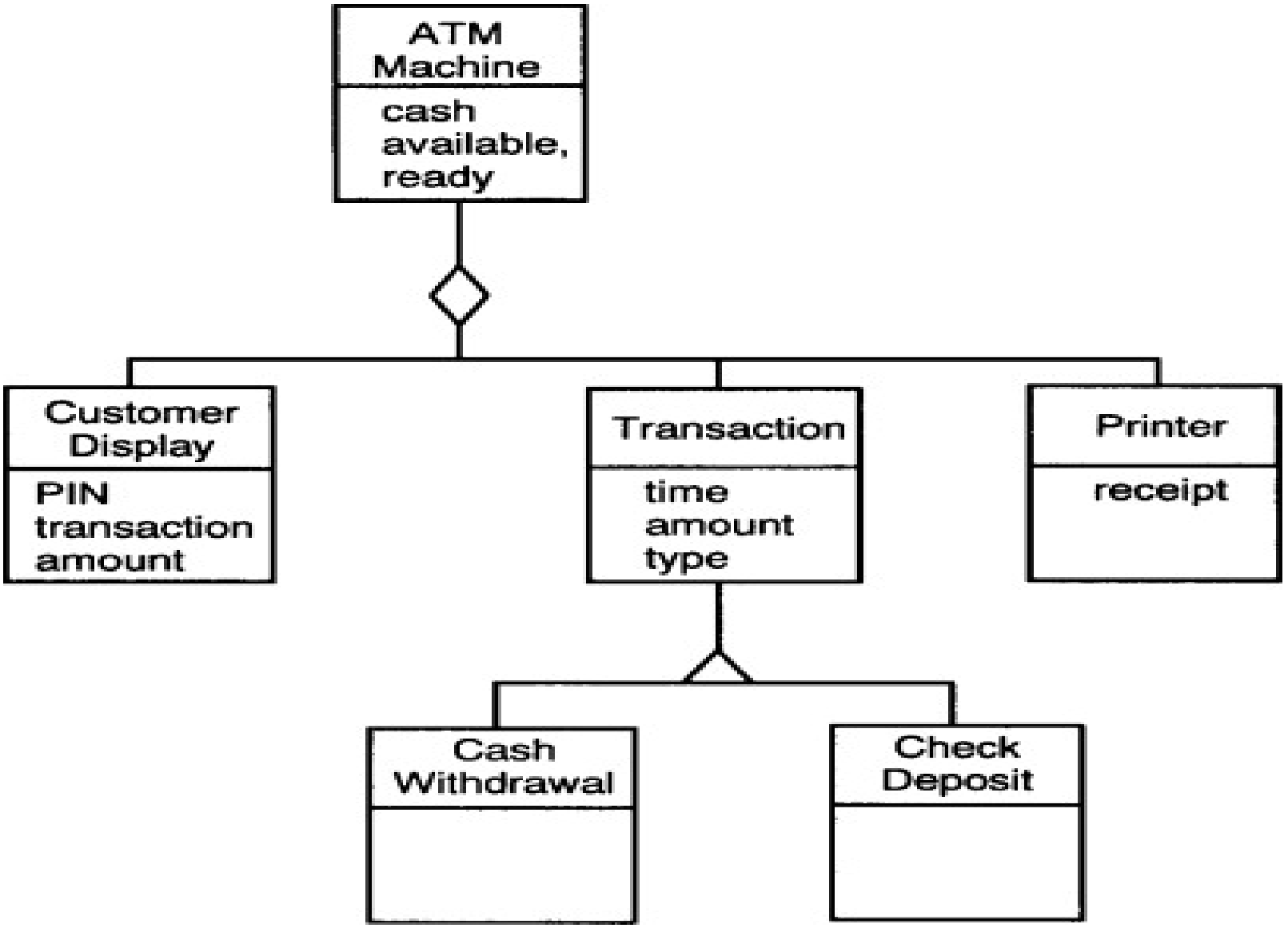
- A type object oriented model.
- Object classes and attributes.
- Interactive Systems Development.
- Arranged in an hierarchy of inheritance.

UML Notation

Inheritance is shown upward rather than downward

Single Inheritance: Every object class inherits its attributes and operations from a single parent class

Multiple Inheritance: A class of several of several parents.


ATM Machine
cash available, ready
Customer Display
PIN
transaction amount
Transaction
time
amount
type
Printer
receipt
Cash Withdrawal
Check Deposit

UML

- Unified modeling language.
- standardized general-purpose visual modeling language in the field of Software Engineering.
- used for specifying, visualizing, constructing, and documenting the primary artifacts of the software system.
- Data modeling, business modeling, object modeling and component modeling.

Types of UML

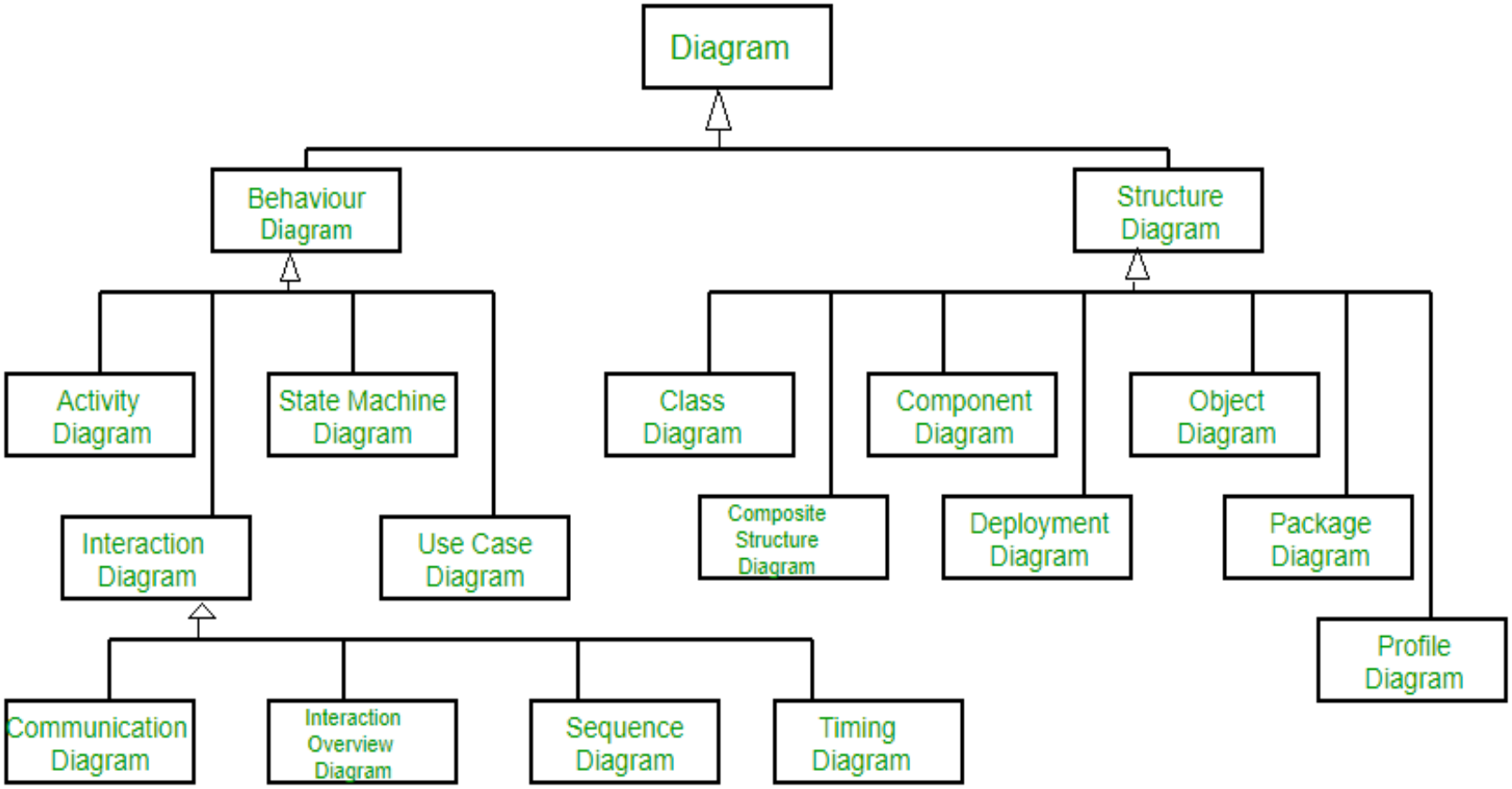


Diagram
Behaviour Diagram
Structure Diagram
Activity Diagram
State Machine Diagram
Interaction Diagram
Use Case Diagram
Communication Diagram
Interaction Overview Diagram
Sequence Diagram
Timing Diagram
Class Diagram
Component Diagram
Object Diagram
Composite Structure Diagram
Deployment Diagram
Package Diagram
Profile Diagram

UML Diagrams

1. **Use case diagram**

   it represents user's interaction with the system.

Relationship between user and use cases.

2. Activity diagram

illustrate the flow of control in a system and refer to the steps involved in the execution of a use case.

activity -------flow---→ activity

Flow chart

UML Diagrams

3. Sequence diagram :

it represents interaction between objects in a sequential order.

focus on life lines.

4. Class Diagram

structure of the system's classes, their attributes, operations and relationship among objects.

-- name, attributes and operation

UML Diagrams

5. Object diagrams :

encompasses objects and their relationship at a point in time.

6. State diagram :

represents the condition of system or part of the system of finite instance of time.

different states of components.

UML Diagrams

7. Component diagram

represents how components are wired together to form a larger components or software system.

complex system

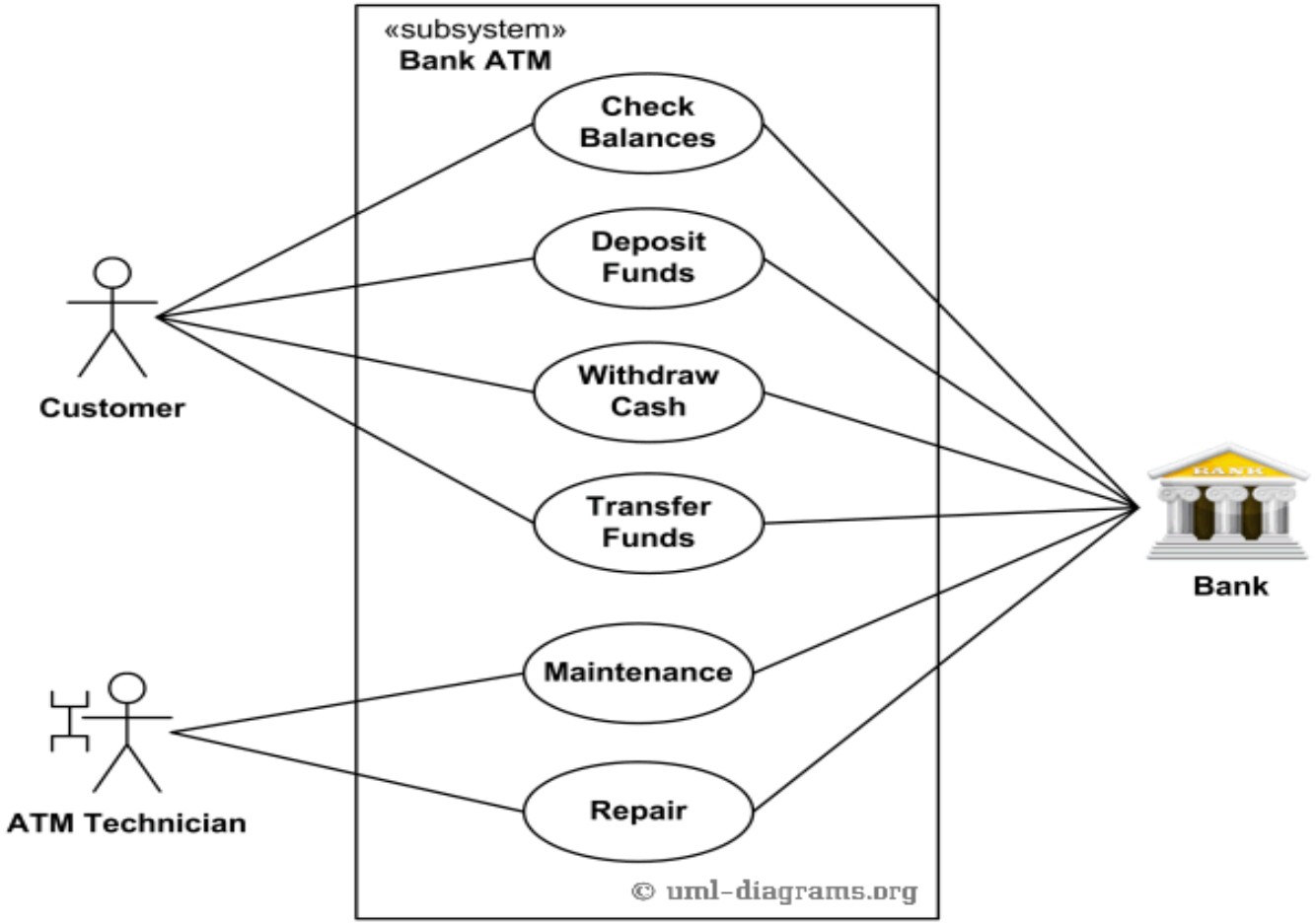
8. Deployment diagram

used to visualize topology of the physical component of a system.

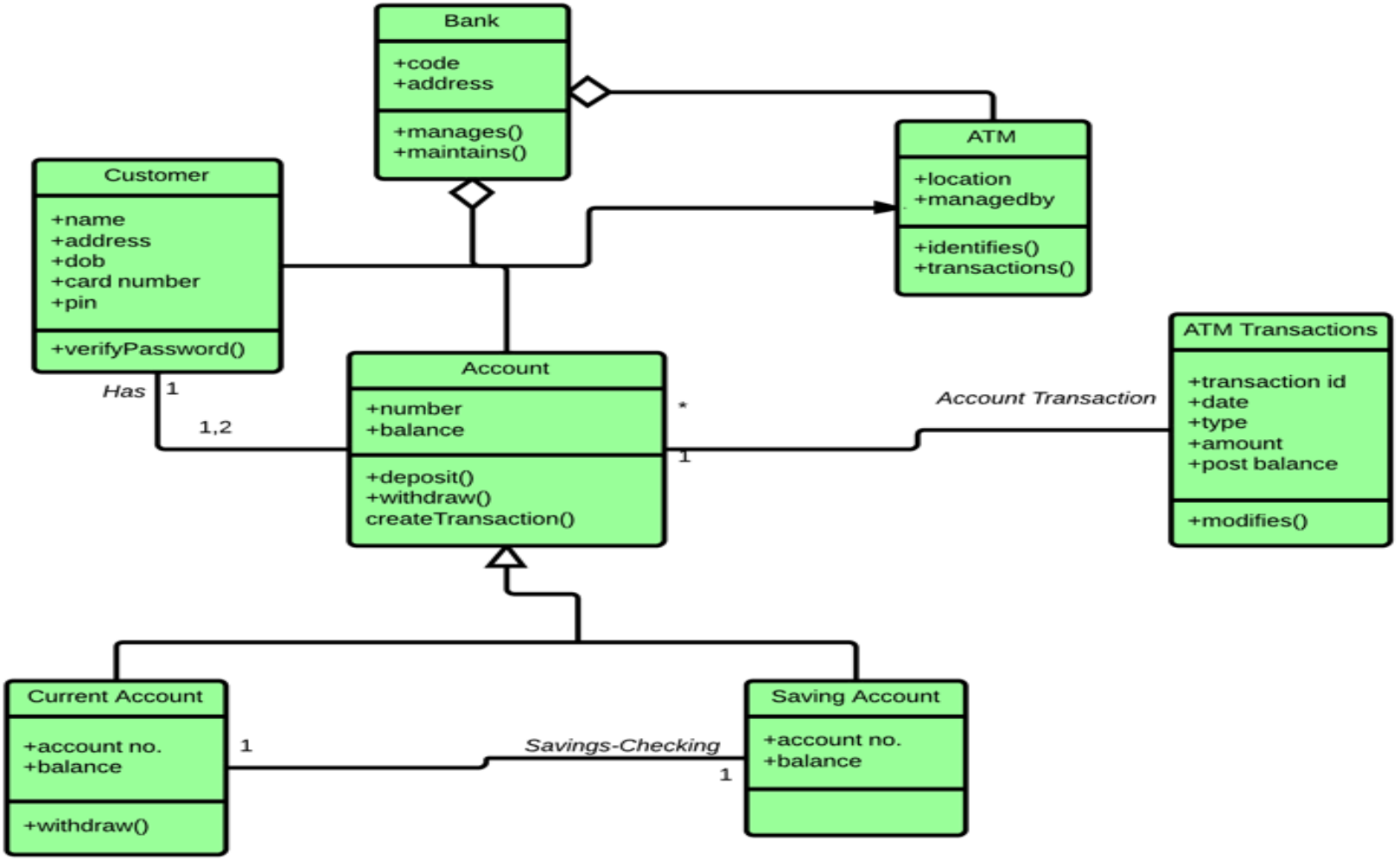
9. Collaboration diagram

shows objects and relationships involved in an interaction, and the sequence of messages exchanged among the objects during the interaction.

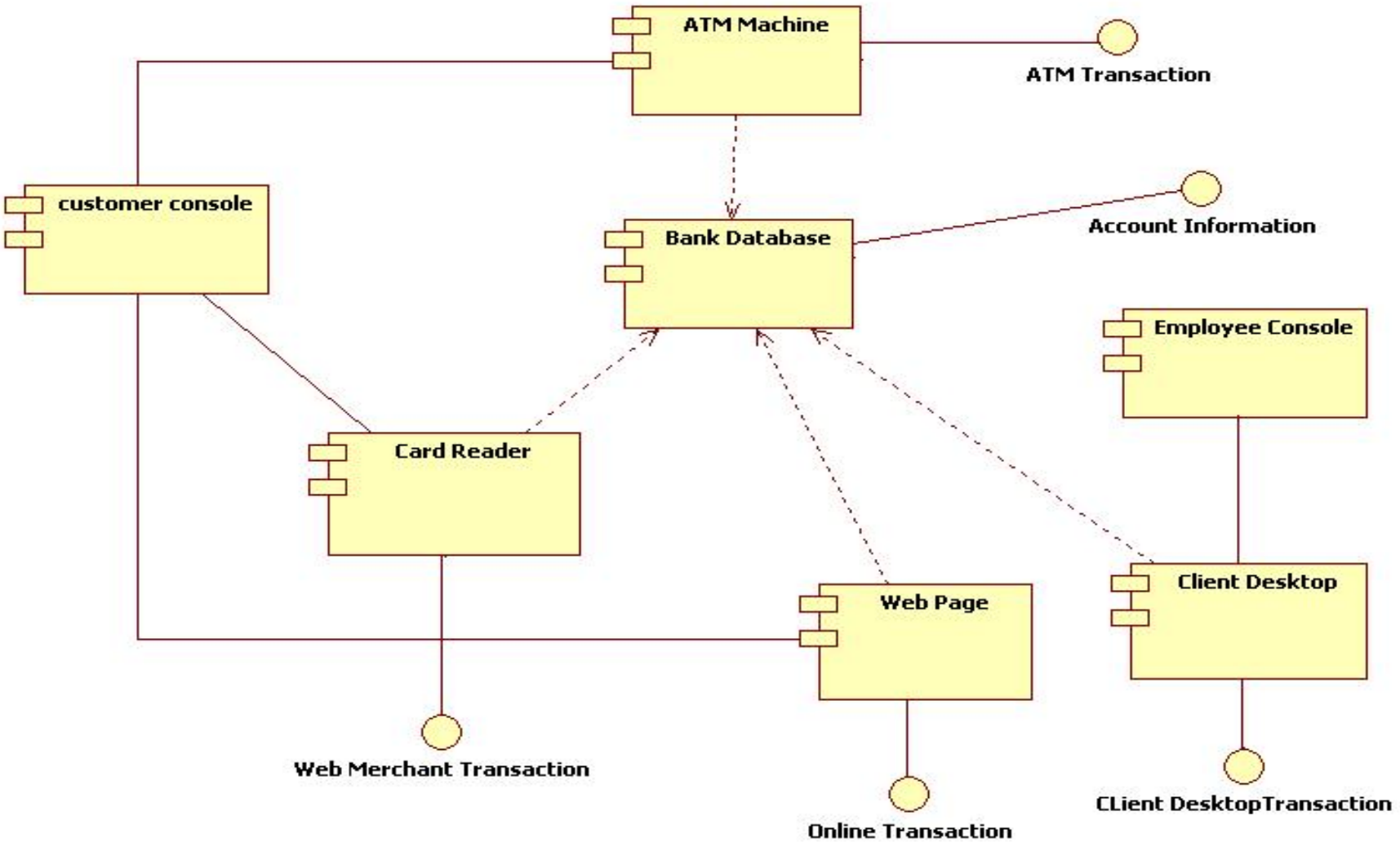
Use case Diagram



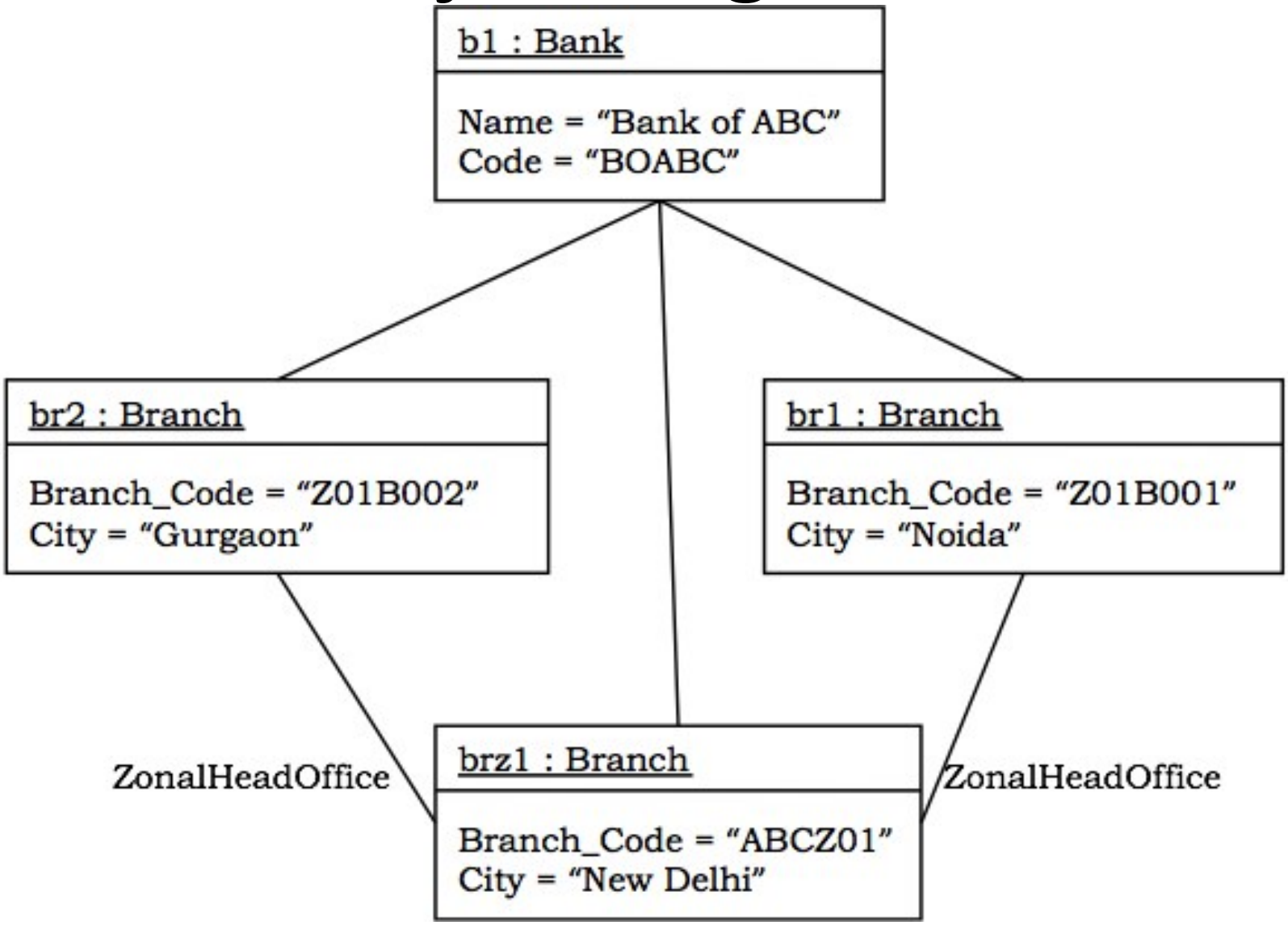
Class Diagram



Component Diagram



Object diagrams



Sequence diagram