Object Oriented Fundamentals and Analysis

What is an object?

- **Objects** are the physical and conceptual things we find in the universe around us. Hardware, software, documents, human beings, and even concepts are all examples of objects.
- An automotive engineer would see tires, doors, engines, top speed, and the current fuel level as objects.
- Atoms, molecules, volumes, and temperatures would all be objects a chemist might consider in creating an object-oriented simulation of a chemical reaction.
- A software engineer would consider stacks, queues, windows, and check boxes as objects.

What is a class?

- A **class** is a pattern, template, or blueprint which describes characteristics of similar objects.
- Group of objects that share common properties, behavior and relationships.
- Computer in college, home or office are similar objects. They are manufactured by different companies having different models.
- Although they have different models, capacities and appearances they have common characteristics to identify them.
- The general common name to describe objects of common or similar characteristics and behavior is class.

Structured vs Object Oriented System

Structured System:

-> System is divided/ decomposed into functions
-> Functions and data are modelled separately
-> System components are dependent on each other
-> Inheritance and polymorphism are impossible
-> Reuse is limited
-> Development process is linear
-> Testing is done in implementation phase

Object Oriented System:

-> System is divided/ decomposed into objects
-> Functions and data are modelled in one place
-> System components are independent of each other
-> Inheritance and Polymorphism are possible
-> No restriction for reuse
-> Development process is iterative and incremental
-> Testing is distributed evenly

What is Object Oriented?

- Paradigm that relies on the concept of objects and classes.
- bind data and functions together that operate on them so that no other part of the code can access this data except that function.

What is Analysis & Design?

- Analysis emphasizes on investigation of the problem & requirements, rather than a solution
- For e.g. if a new trading system is required
  - Analysis deals with
    - what are its functions?
    - how will it be used?
- Analysis is a broad term, best qualified as in requirement analysis(investigation of requirements) or object oriented analysis .

What is Analysis & Design?

- Design emphasizes a conceptual solution(in software & hardware) that fulfills the requirement, rather than its implementation.
- For e.g. description of database schema
- Design ideas often exclude low level details
- Ultimately designs are implemented , & the implementation expresses the true & complete realized design

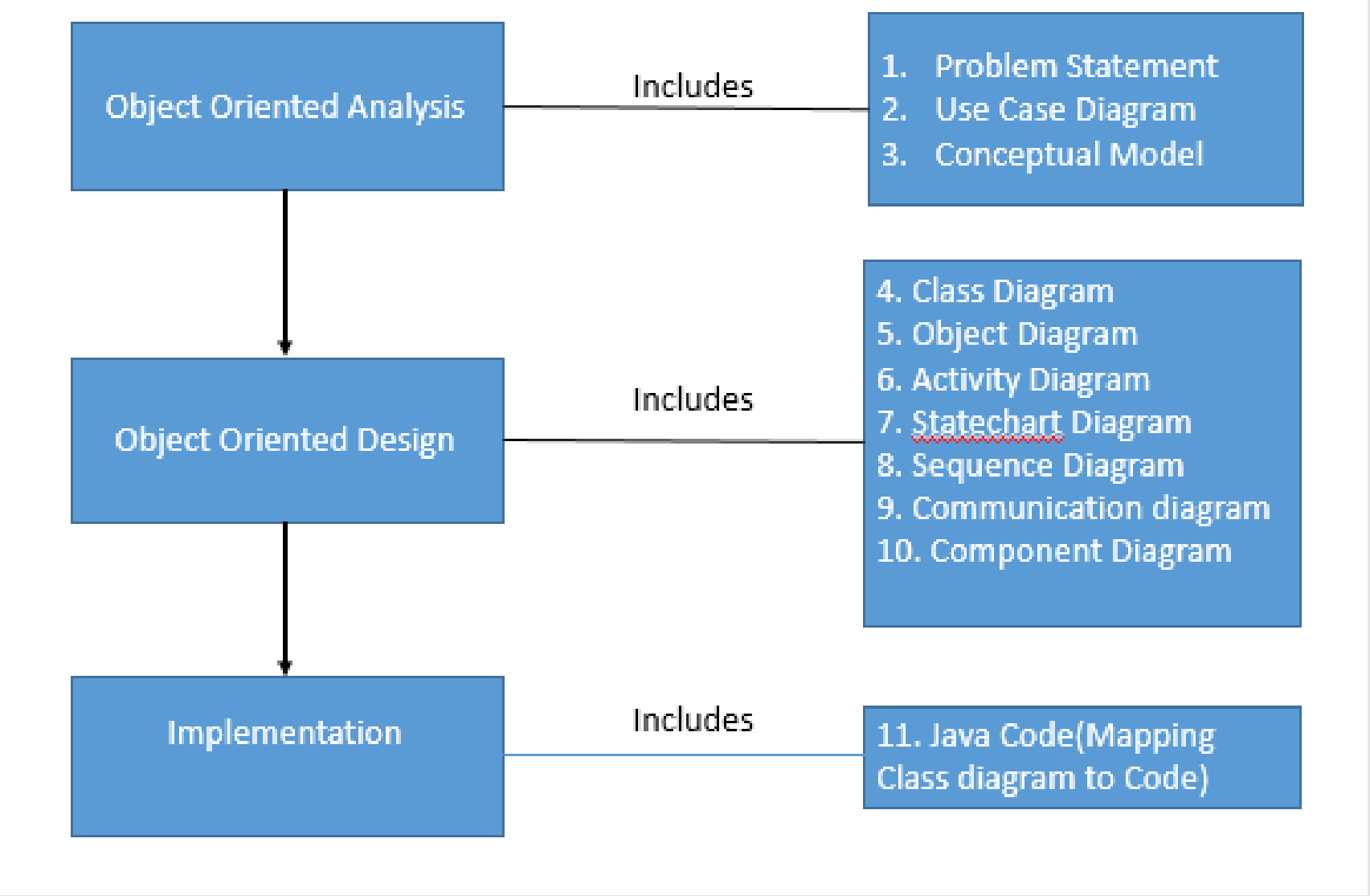
What is object oriented analysis & design?

- Object Oriented Analysis emphasizes on finding & describing the objects- or concepts in the problem domain.
- For e.g. flight automation system
  - Concepts include Plane, Flight, Pilot etc.
- During the object oriented design, there is an emphasis on defining software objects & how they collaborate to fulfill the requirements.

What is object oriented analysis & design?

- For example Flight Automation System has one object –Plane
- May have
  - Attribute – tailNumber
  - Method - getFlightHistory
- finally during implementation i.e. OOP, design objects are implemented, such as Plane class in Java

Flow of OOAD


Object Oriented Analysis
Includes
1. Problem Statement
2. Use Case Diagram
3. Conceptual Model
Object Oriented Design
Includes
4. Class Diagram
5. Object Diagram
6. Activity Diagram
7. Statechart Diagram
8. Sequence Diagram
9. Communication diagram
10. Component Diagram
Implementation
Includes
11. Java Code(Mapping Class diagram to Code)

Object Oriented Analysis

- *The OOA phase deals with the investigation of the problem and requirements, rather than finding a solution to the problem.*
  - the phase of OOA the typical question starts with What...? Like
    - "What will my program need to do?",
    - "What will the classes in my program be?" and
    - "What will each class be responsible for?"
- Priority is given to find and describe the objects or concepts in the problem domain.
- Object oriented analysis emphasizes the building of real-world model using the object oriented view of the world
- For example, in the case of the Library information system, some of the concepts include *Book, Library,* and Member

Object Oriented Design

- OOD phase, the question typically starts with How...? like
  - “How will this class handle it’s responsibilities?”,
  - “How to ensure that this class knows all the information it needs?” and
  - “How will classes in the design communicate?”
- Finds a conceptual solution of the problem analyzed in analysis phase
- System attributes, activities, classes, objects, states are defined in this phase and emphasis is given how they collaborates to fulfill the requirement
- Does not deals with the system implementation and coding
- Blueprint for Implementation

Implementation

- Transform the solution created in design phase into code
- Implement all the system attributes, methods, states, class, objects
- Programs are organized as cooperative collection of objects, each of which represents an instance of some class and whose classes are all members of a hierarchy of classes united in inheritance relationships.

Defining Models

- A model of an object is a physical representation that shows what it looks like or how it works.
- Model helps to visualize, specify, construct and document the artifacts of a software intensive system
- We build models so that we can better understand the system we are developing.
- A model provides the blueprints of a system

Reasons to Model

- to communicate the desired structure and behavior of the system
- to visualize and control the system's architecture
- to better understand the system and expose opportunities for simplification and reuse
- to manage risk

Importance of Model Building

- Each model within a design describes a specific aspect/ *perspective* of the system under consideration, when put together will provide an overall view of the system.
- Model building appeals to the principles of decomposition and hierarchy.
- *Models in software help us to*
  - *visualize,*
  - *specify,*
  - *construct and*
  - *document the artifacts of a software intensive system.*
- Models give us the opportunity to fail under controlled conditions.
- We evaluate each model under both expected and unusual situations and then after them when they fail to behave as we expect or desire. More than one kind of model is used on order to express all the subtleties of a complex system.

Physical vs Logical Model

Logical Model:

- During the Analysis phase, Logical Models are created.
- It is a conceptual representation of the system that focuses on the system's functionality, behavior, and relationships between objects.
- It does not take into account the physical implementation details, such as hardware or software platforms, and is independent of any specific programming language or technology.
- The classes and objects that form the system are identified in a logical model. For this logical model, again two different perspectives have to be considered
  - A static perspective identifies the structure of classes and objects, their properties and the relationships classes and objects participate in.
  - A dynamic model identifies the dynamic behavior of classes and objects, the different valid states they can be in and the transitions between these states.

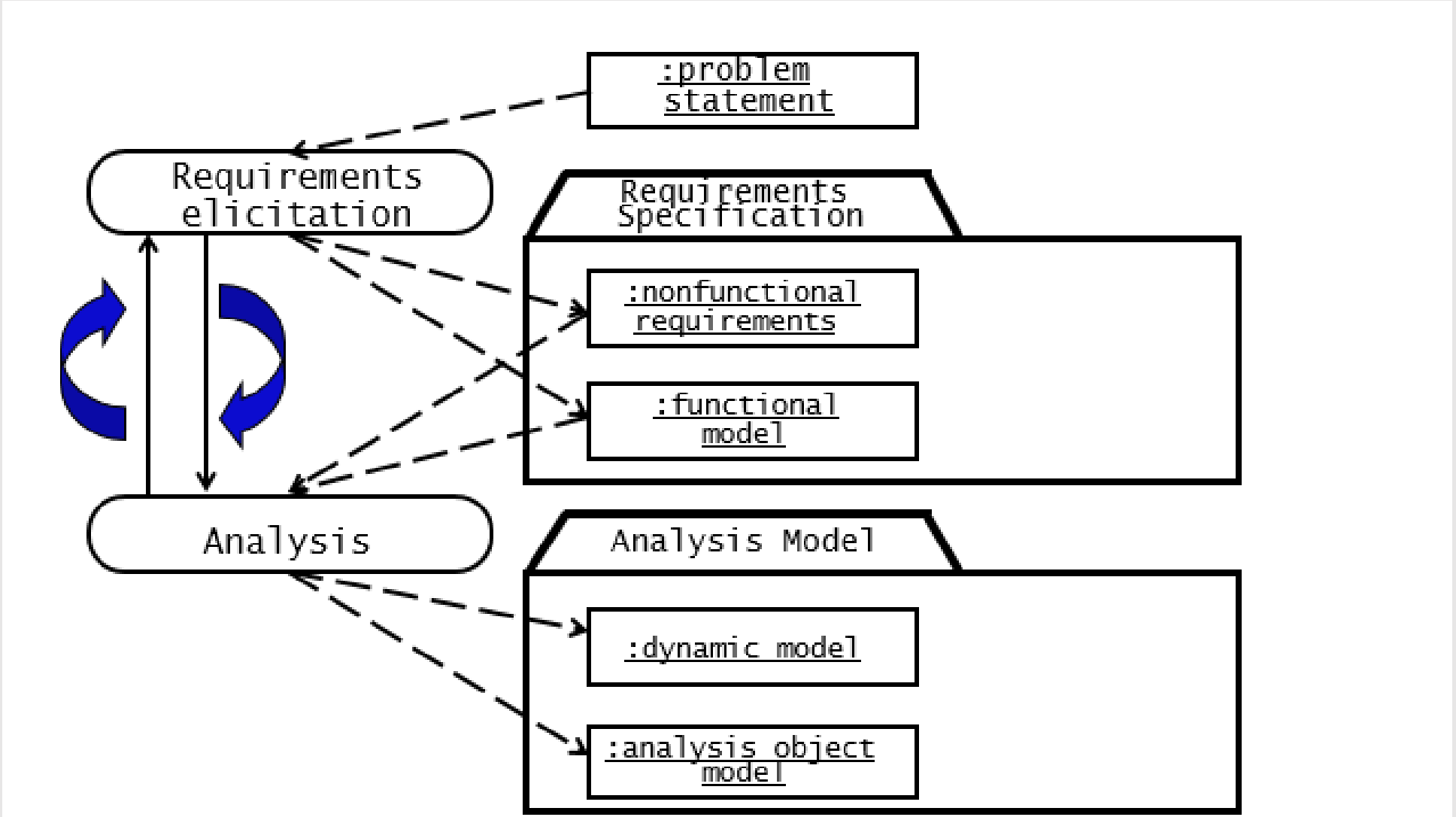
Physical Model

- During the Design phase, Physical Models are created.
- It is a representation of the system that focuses on the physical implementation details, such as the hardware and software components, system architecture, and network infrastructure.
- It is concerned with the technical aspects of the system and provides a blueprint for building and deploying the system.
- The module architecture identifies how classes are kept in separately compliable modules and the process architecture identifies how objects are distributed at run-time over different operating system processes and identifies the relationships between those.
  - Again for this physical model a static perspective is defined that considers the structure of module and process architecture and a dynamic perspective identifies process and object activation strategies and inter-process communication.

Requirement processes

- It is a systematic approach to find, document, organize and track the needs of the users and response on changing requirements of a system.
- Requirements are the aspect that the system must confirm in the initial phase
- It is the process of understanding and defining what services are required and identifying the constraints on these services.
- Requirements engineering processes ensures your software will meet the user expectations, and ending up with a high quality software.

Requirement Process



Requirement Elicitation

- It is an practice of collecting requirements of a system from the users, customers and other stakeholders.
- It is related to the various ways used to gain knowledge about the project domain and requirements.
- It deals with all the activities required in gathering the requirements of a system.
- The various sources of domain knowledge include customers, business manuals, the existing software of same type, standards and other stakeholders of the project.
- The developers and engineers work in close coordination with the customers and end-users to identify more about the problem to be solved and to bridge the gap between expectation and understanding.

Methods of Requirement Elicitation

- **Interviews**: Interviews are strong medium to collect requirements.
- **Questionnaire**: A document with pre-defined set of objective questions and respective options is handed over to all stakeholders to answer, which are collected and compiled.
- **Survey**: Organization may conduct surveys among various stakeholders by querying about their expectation and requirements from the upcoming system.
- **Domain Analysis**: Every software falls into some domain category. The expert people in the particular domain can be a great help to analyze general and specific requirements
- **Observation**: observe the actual working of the existing similar type of installed systems
- **Brain Storming** :a spontaneous group debate to produce ideas and ways of solving problems

Object Oriented SDLC

- The motive of the software development process that consists of analysis, design, implementation, testing, and refinement is to transform user's needs into a software solution that satisfies those needs.
- Some people view software development process as interesting but feel it has little importance in developing software.
- In general, dynamics of software development provides little room for such shortcuts, and bypasses have been less than successful.
- The object oriented approach requires a more rigid process to do things right.
- You need not see code until after about 25 percent of the development time, because you need to spend more time in gathering requirements, developing a requirement model and an analysis model, then turning them into the design model.

Object Oriented SDLC

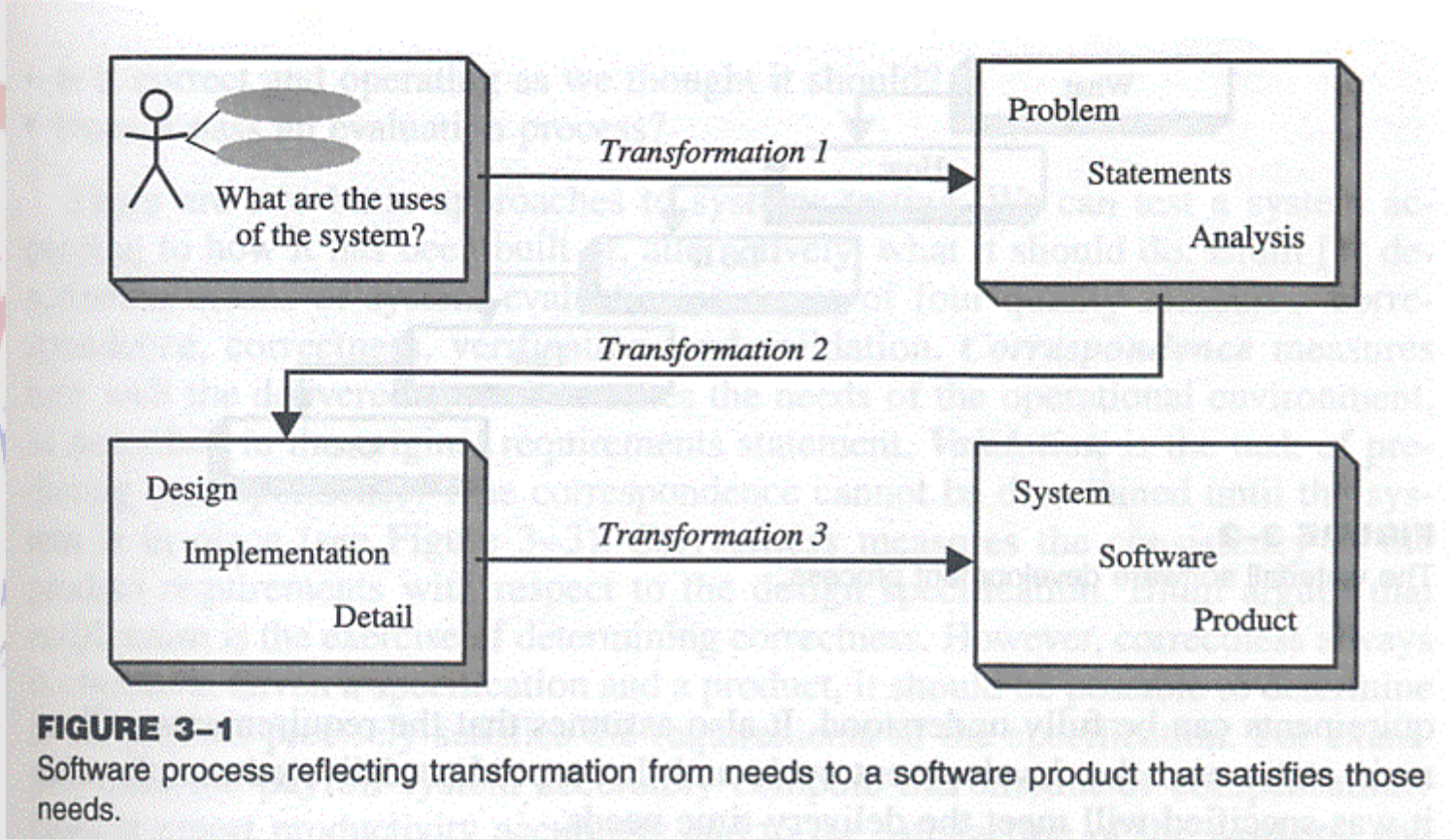


FIGURE 3–1
Software process reflecting transformation from needs to a software product that satisfies those needs.

Object Oriented Development Cycle

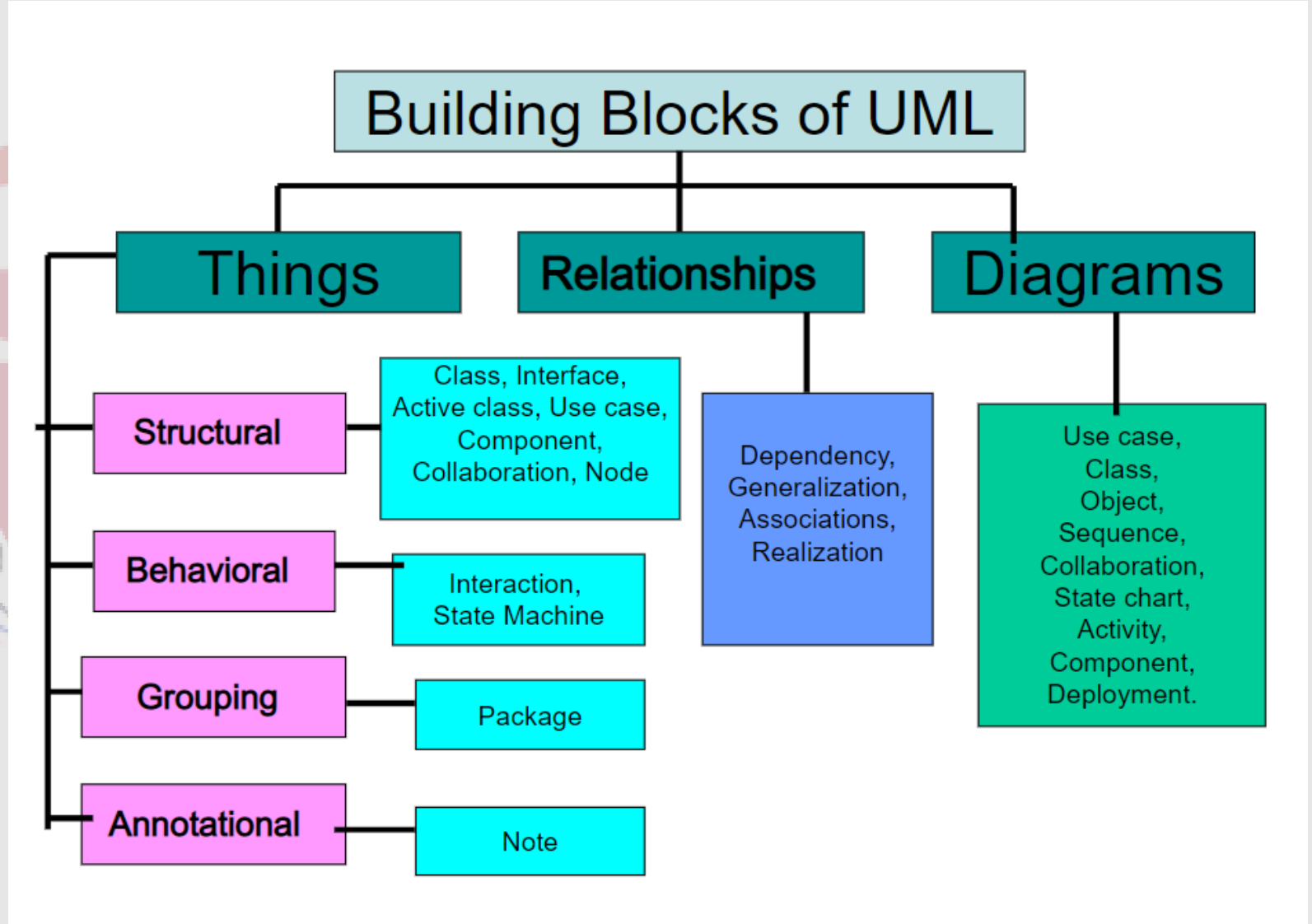
- transformation 1(analysis) - translates user's need into system's requirements & responsibilities
  - how they use system can give insight into requirements, e.g.: analyzing incentive payroll - capacity must be included in requirements
- transformation 2 (design) - begins with problem statement, ends with detailed design that can be transformed into operational system
  - Includes bulk of development activity, include definition on how to build software, its development, its testing, design description + program+ testing material
- transformation 3 (implementation) - refines detailed design into system deployment that will satisfy user's needs

Conceptual Model of UML

The conceptual model of UML contains the fundamentals of UML. It consists of three parts. They are:

- **Building blocks** of UML (syntax / vocabulary)
- **Rules** (semantics) that dictate how these building blocks may be put together
- **Common Mechanisms** that apply throughout the UML.

Building Blocks of UML



Use Cases

- A UML use case diagram is the primary form of system/software requirements for a new software program underdeveloped.
- Use cases specify the expected behavior (what), and not the exact method of making it happen (how).
- A key concept of use case modeling is that it helps us design a system from the end user's perspective. It is an effective technique for communicating system behavior in the user's terms by specifying all externally visible system behavior.
- It only summarizes some of the relationships between use cases, actors, and systems.
- It does not show the order in which steps are performed to achieve the goals of each use case.

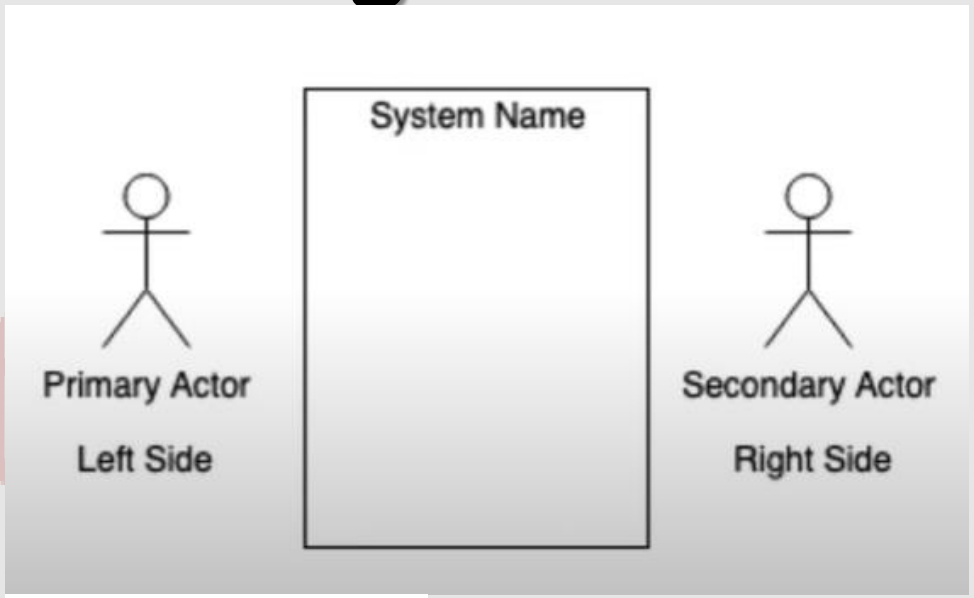
Purpose of Use Case Diagram

Use case diagrams are typically developed in the early stage of development and people often apply use case modeling for the following purposes:

- Specify the context of a system
- Capture the requirements of a system
- Validate a systems architecture
- Drive implementation and generate test cases
- Developed by analysts together with domain experts

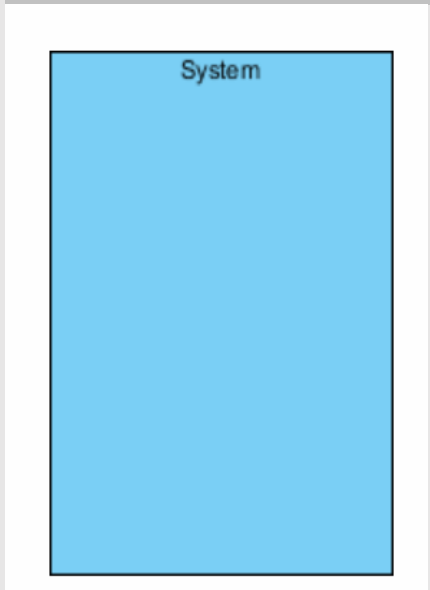
Essential Elements of Use Case Diagram

- Actors: Someone interacts with use case



- Use Case: system function

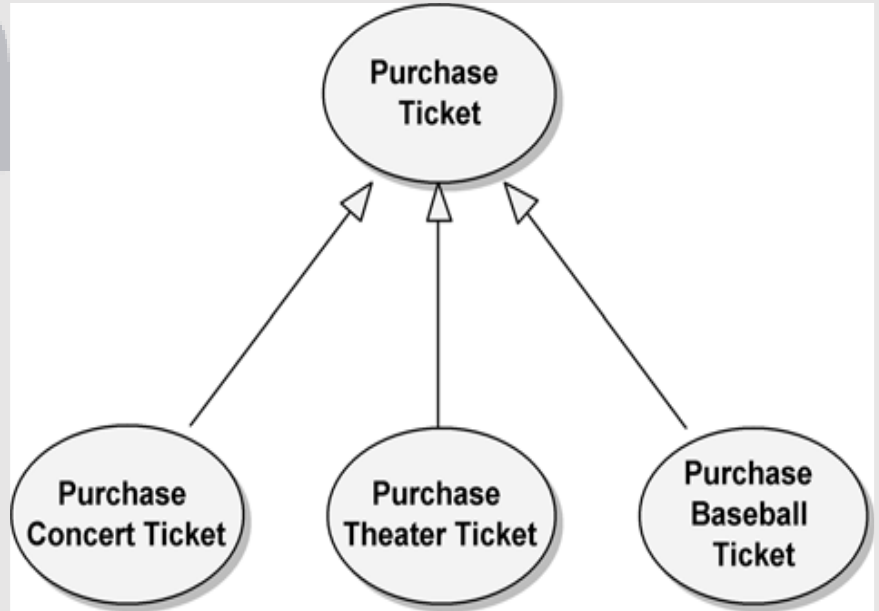
- System Boundary:



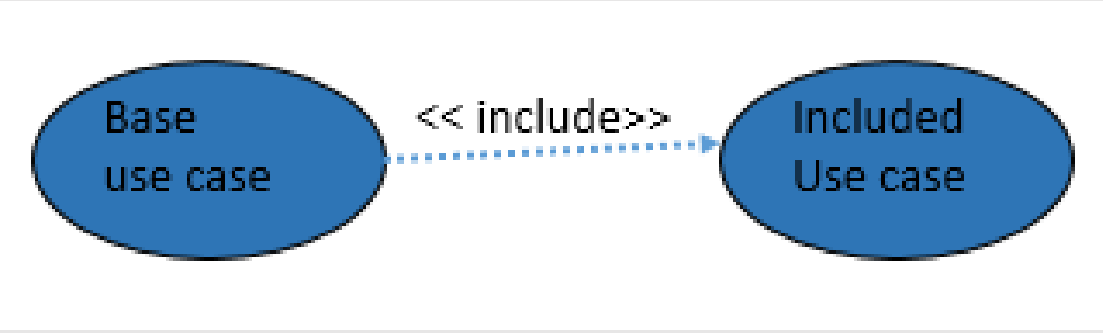
Relationship in Use Case Diagram

- **Association:** communication between an actor and a use case is represented by a solid line.

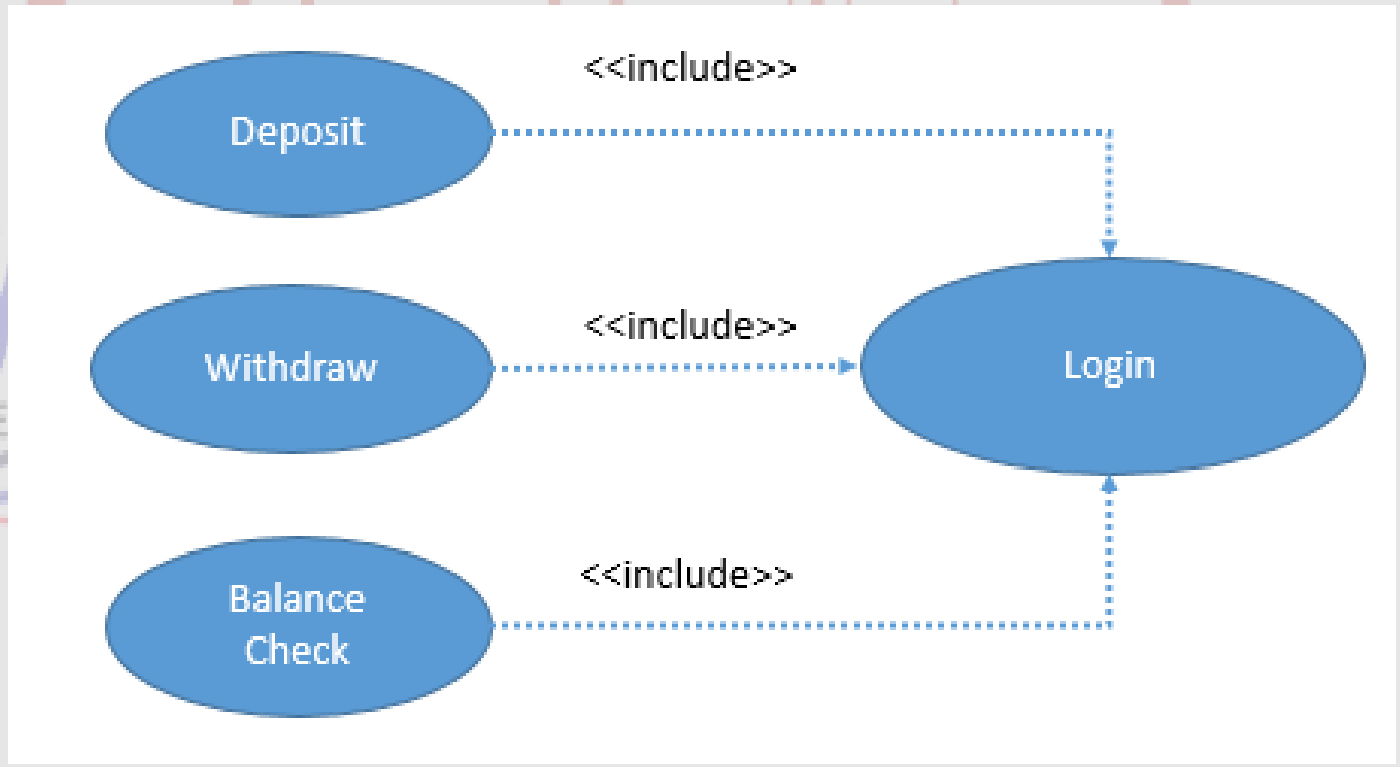
- **Generalization:** relationship between one general use case and a special use case



«include» Relationships

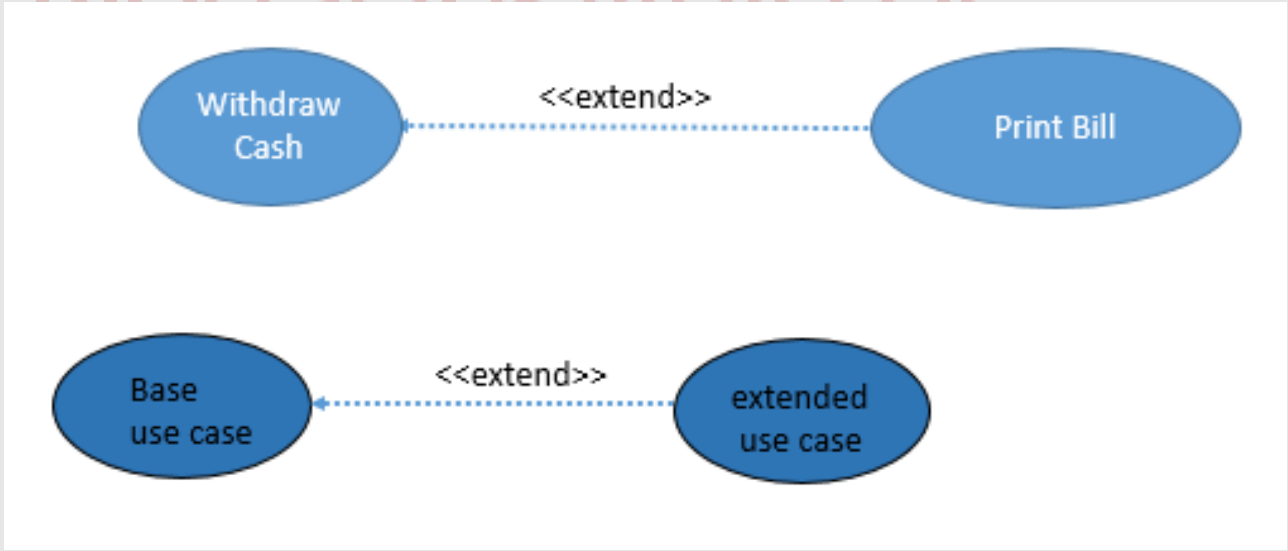


- It is a relationship in which one use case (the base use case) includes the functionality of another use case (the inclusion use case)

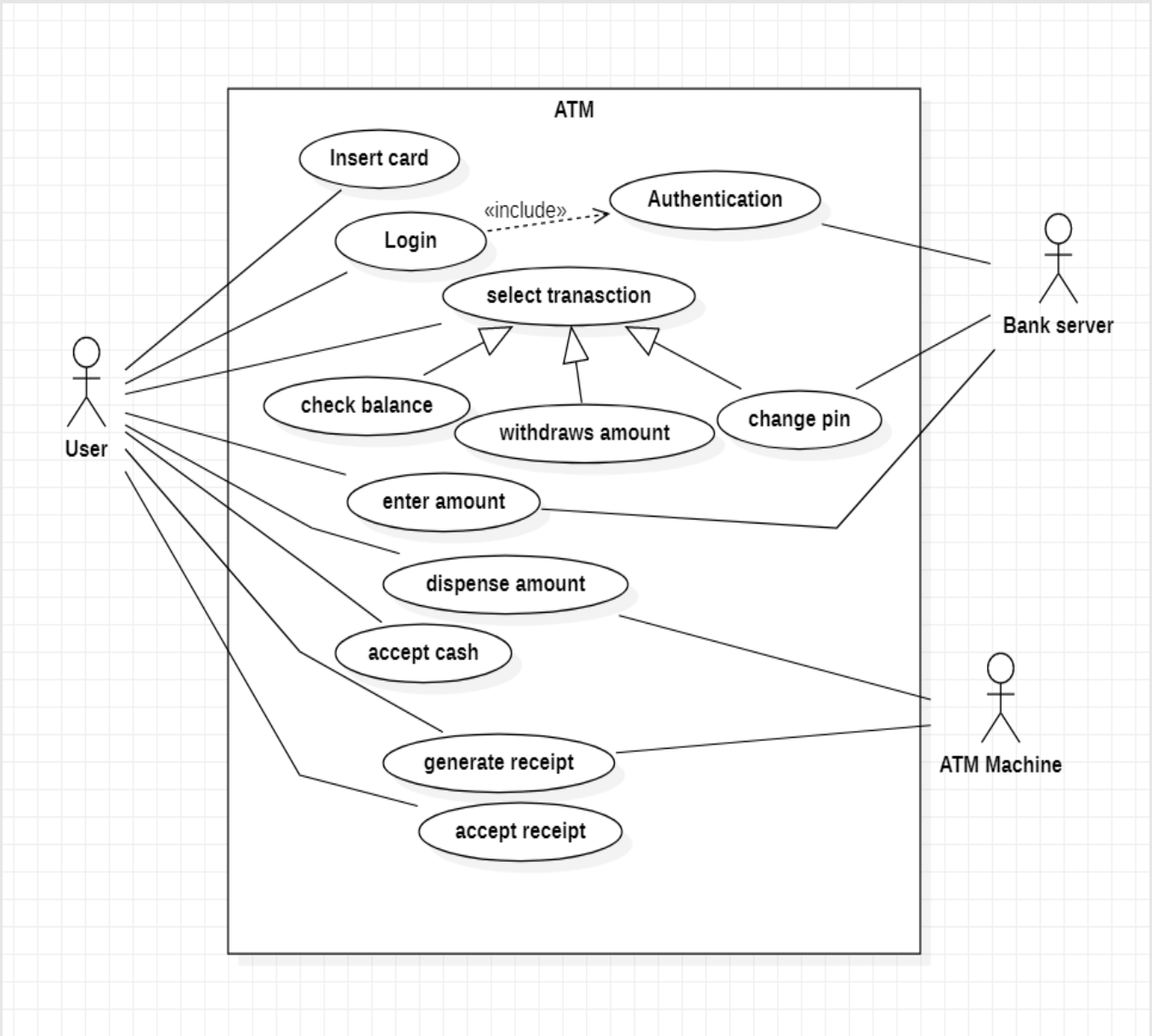


<<Extend>> relationship:

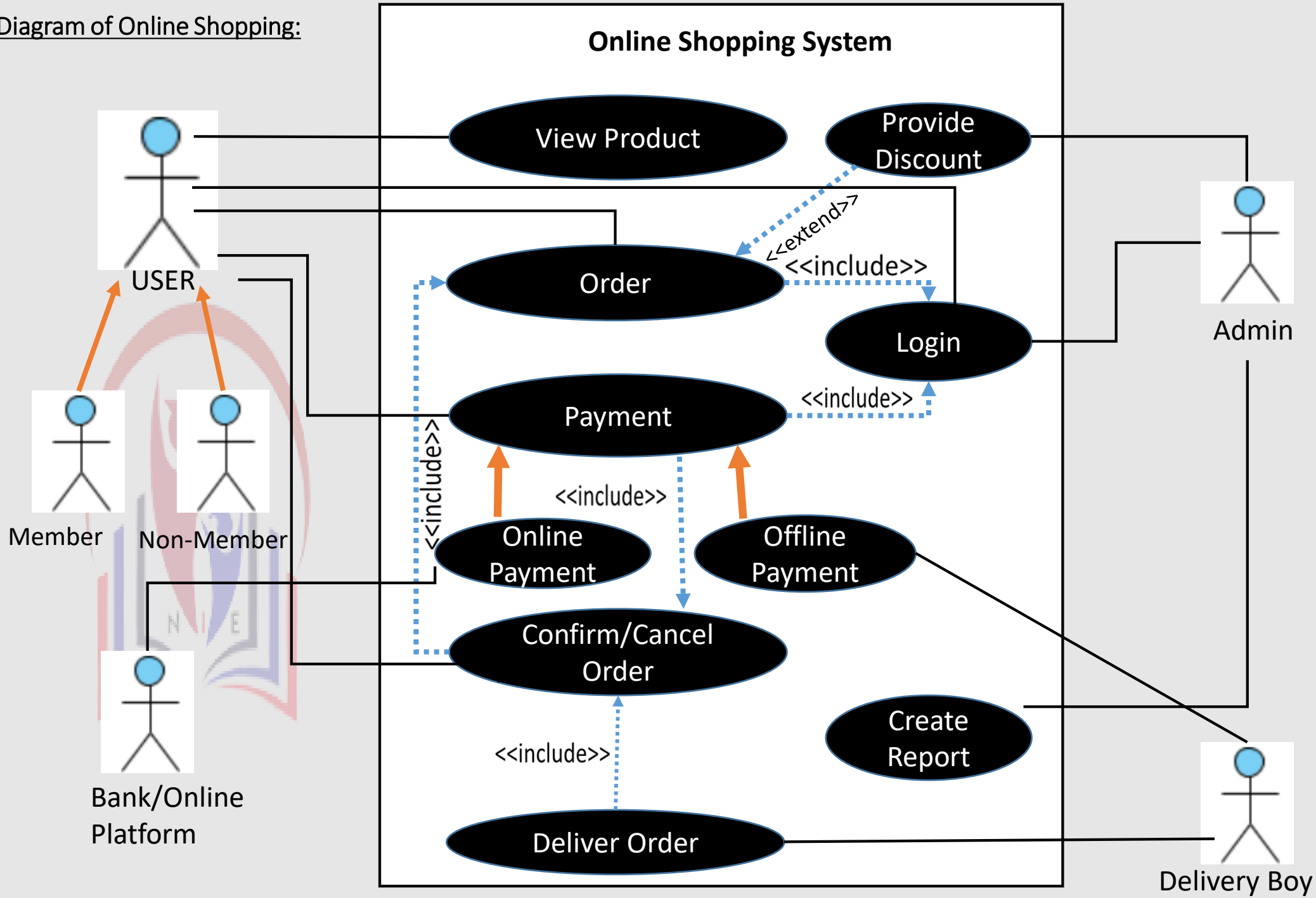
- While developing your use cases, you may find that certain activities might be performed as part of the use case but are not mandatory for that use case to run successfully.



ATM


ATM
Insert card
Authentication
«include»
Login
select tranasction
Bank server
User
check balance
withdraws amount
change pin
enter amount
dispense amount
accept cash
ATM Machine
generate receipt
accept receipt

Use Case Diagram of Online Shopping:



Domain Model

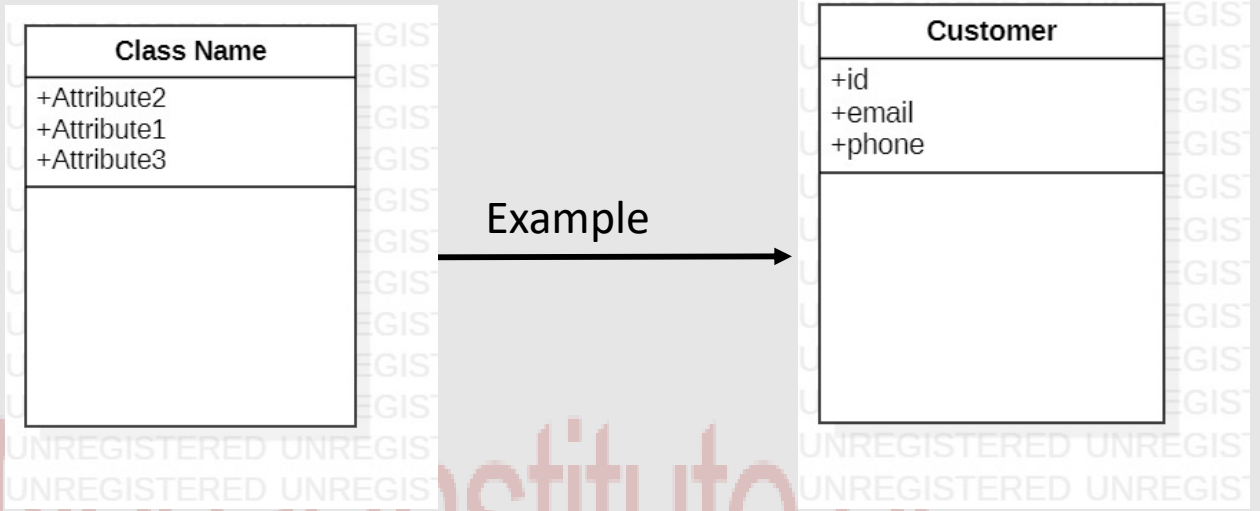
- Domain modeling is a technique used to understand the project problem description and to translate the requirements of that project into software components of a solution. The software components are commonly implemented in an object oriented programming language.
- It is also called **conceptual models, domain object models, and analysis object models.**
- A domain model contains conceptual classes, associations between conceptual classes, and attributes of a conceptual class.

Conceptual Model

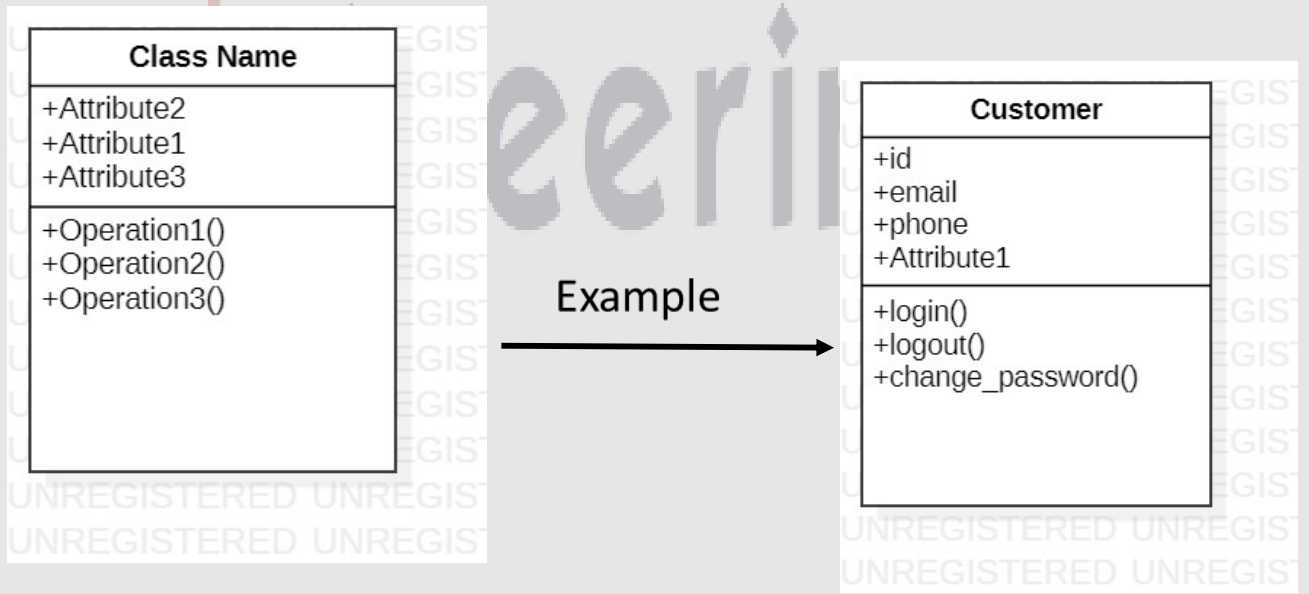
- Steps to build a domain/conceptual model:
  - Find Conceptual Classes/ Domain Entities
  - Add associations
  - Add attributes

Conceptual Class vs Complete Class

- Conceptual Class:



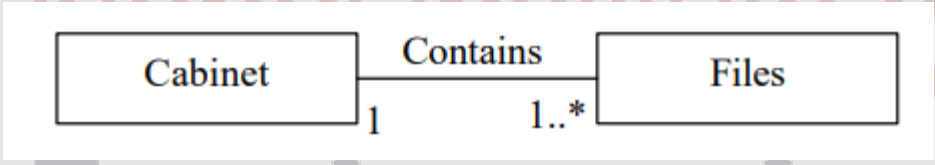
- Complete Class:



Relationships in Domain Model

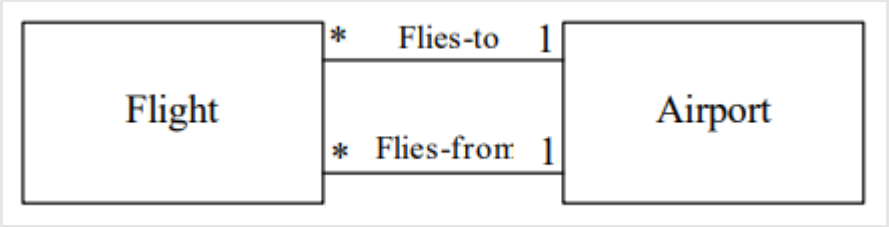
Associations:

- It is a semantic relationship between things, concepts or ideas. It represents meaningful connection.



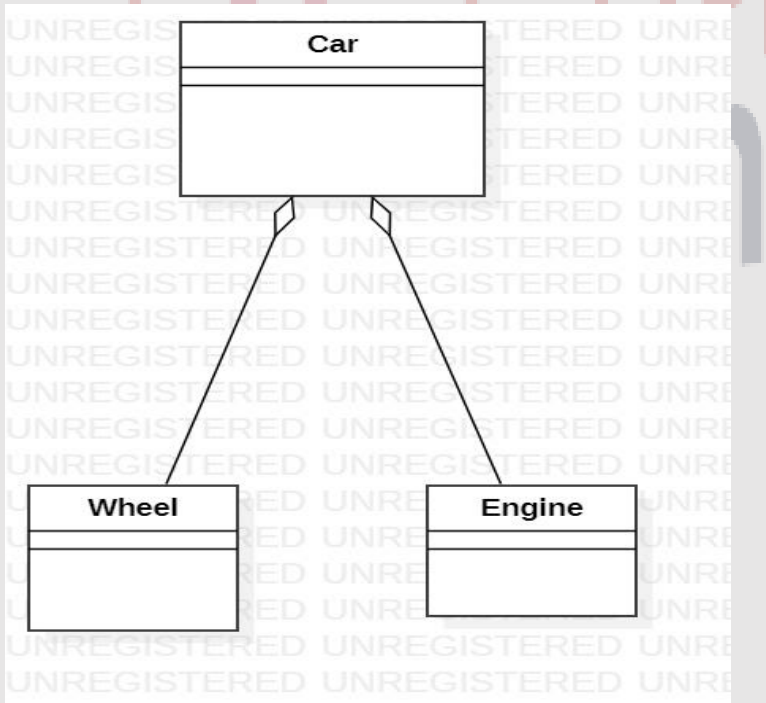
Multiple Association:

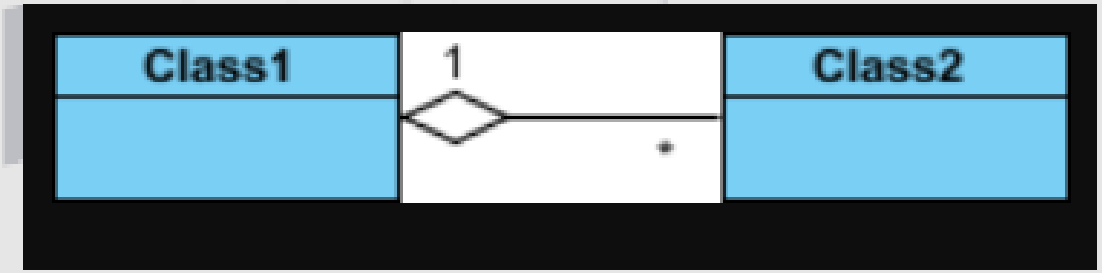
- More than an association can exist between two or more classes.



Aggregation

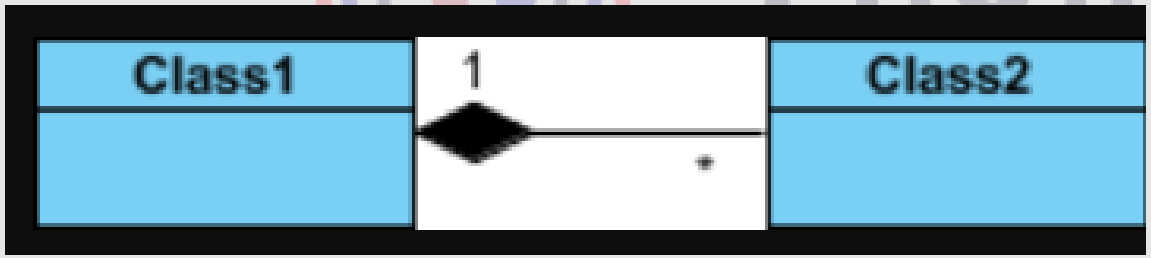
- It indicates a relationship where the child can exist separately from their parent class.
- Example: Automobile (Parent) and Car (Child). So, If you delete the Automobile, the child Car still exist.

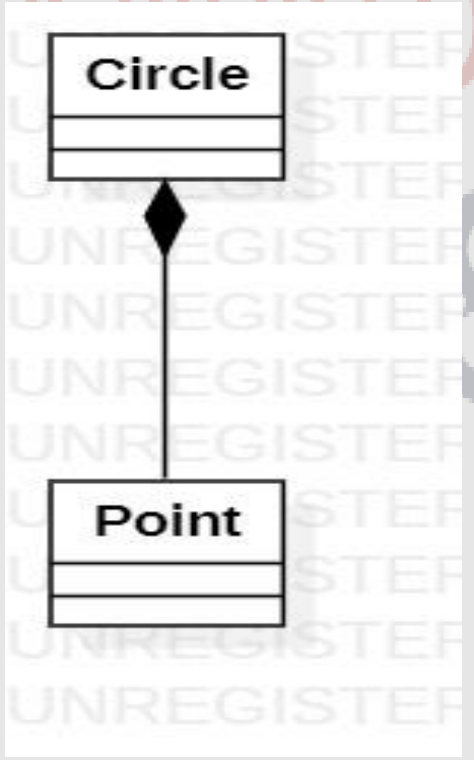


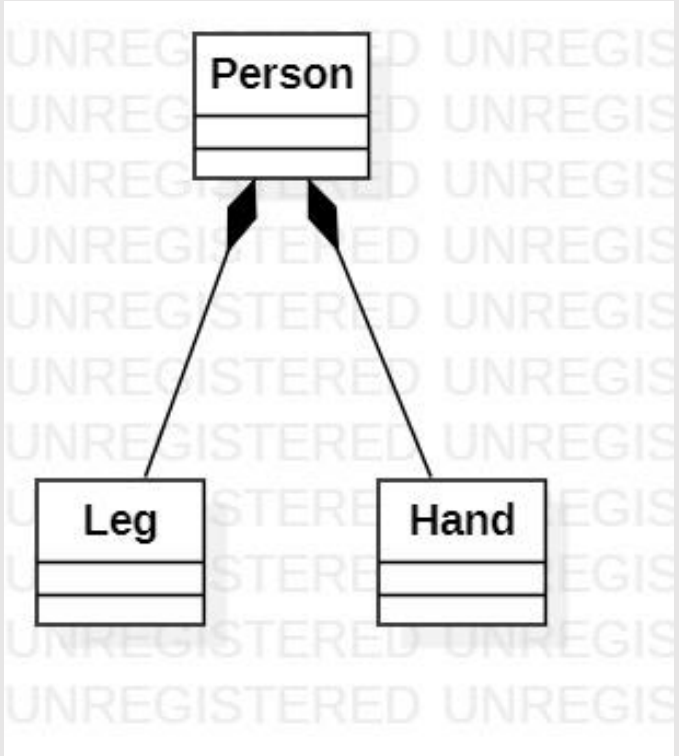


Composition

- **It** display relationship where the child will never exist independent of the parent.
- Example: House (parent) and Room (child). Rooms will never separate into a House.







Multiplicity

- How many objects of each class take part in the relationships and multiplicity can be expressed as:
- Exactly one - 1
- Zero or one - 0..1
- Many - 0..* or *
- One or more - 1..*
- Exact Number - e.g. 3..4 or 6
- Or a complex relationship - e.g. 0..1, 3..4, 6.* would mean any number of objects other than 2 or 5

Car Rental System

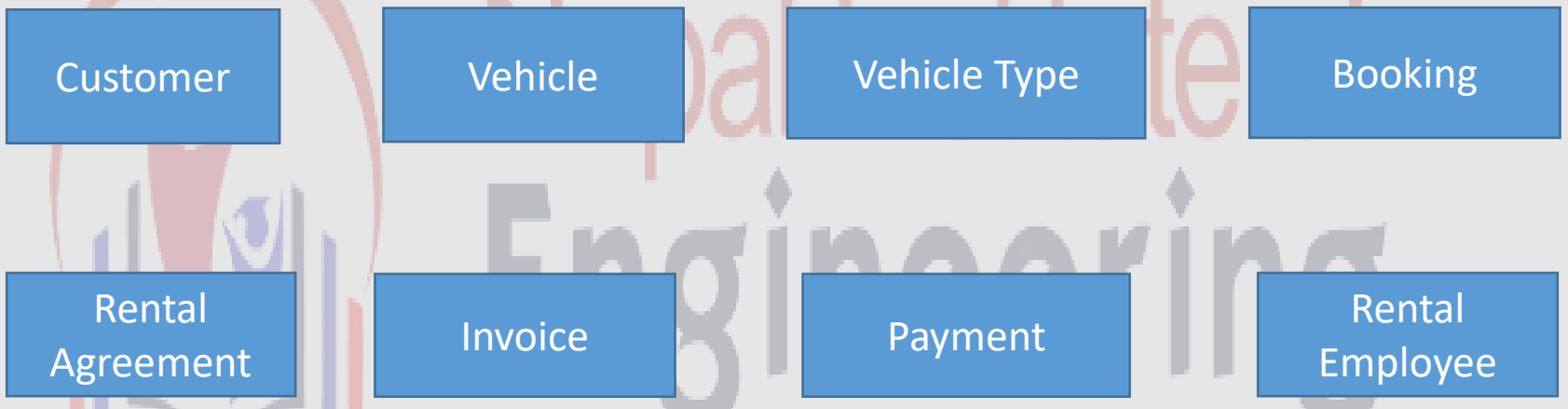
The problem statement of this project is to develop an *online process of vehicle rental service in Pune*. The current system is manual and it is time consuming as well as ineffective in terms of returns. Currently, customers have to call manually in order to rent or reserve a vehicle. The staff of the vehicle rental service company will check their file to see which vehicle is available for rental. *The goal of this project is to automate vehicle rental and reservation so that customers do not need to call and spend unnecessary time in order to reserve a vehicle.*

They can go online and reserve any kind of vehicle they want and that is available. Even when a customer chooses to visit the booking centre to personally hire a vehicle, computers are available for him to go online and perform his reservation. When he choose to reserve by phone, any of the customer service representatives can help him reserve the vehicle speedily and issue him a reservation number.

Car Rental System

- **Conceptual Classes List:**

  Customer, Vehicle, Vehicle Type, Booking, Rental Agreement, Invoice, Payment

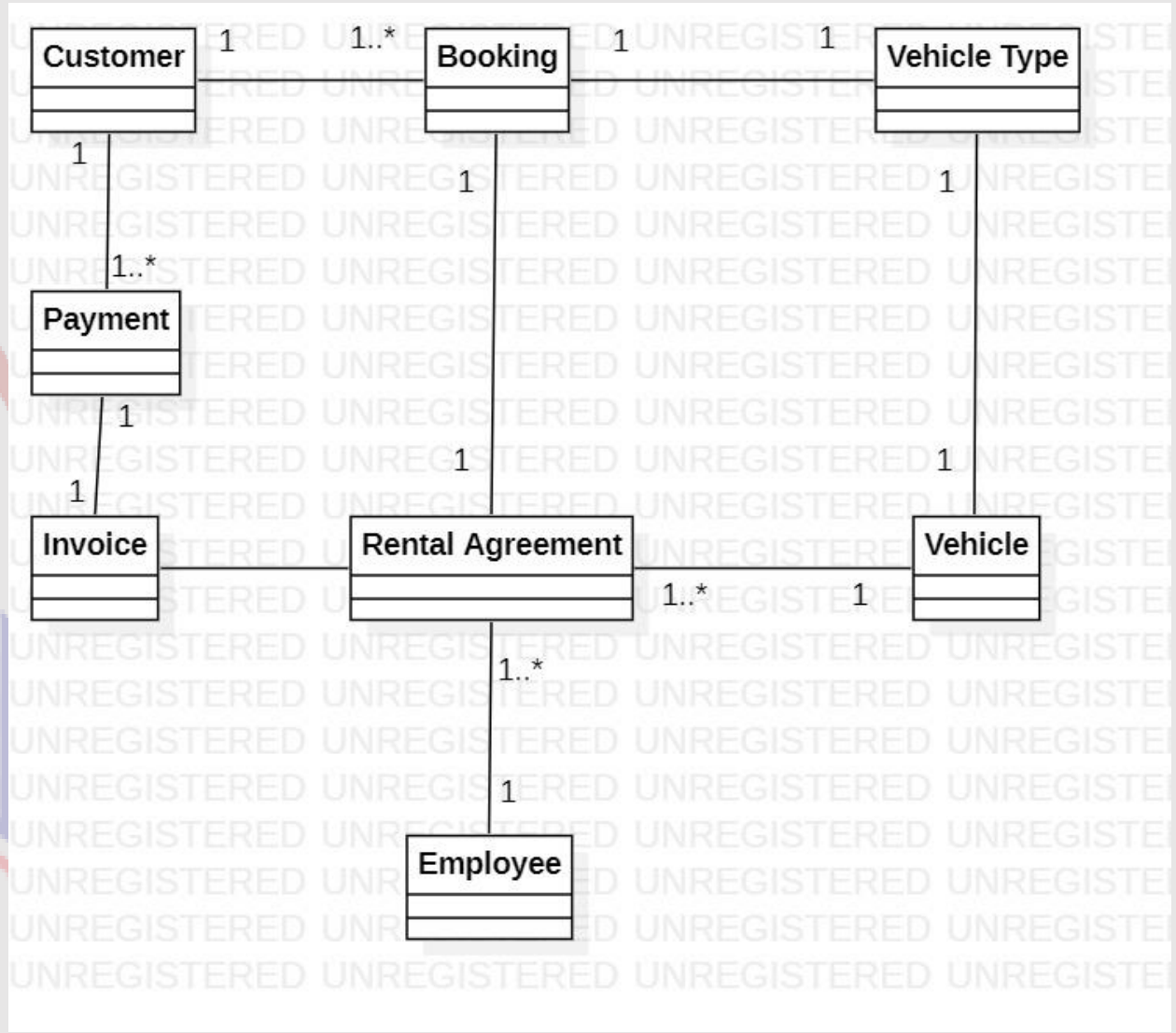


Building Conceptual Model

Add Associations :

- After listing out the conceptual classes, visualize the connection of listed classes with each other through association relationships.
- Add multiplicity in the relationships
- You can add the connection name if required

Car Rental System

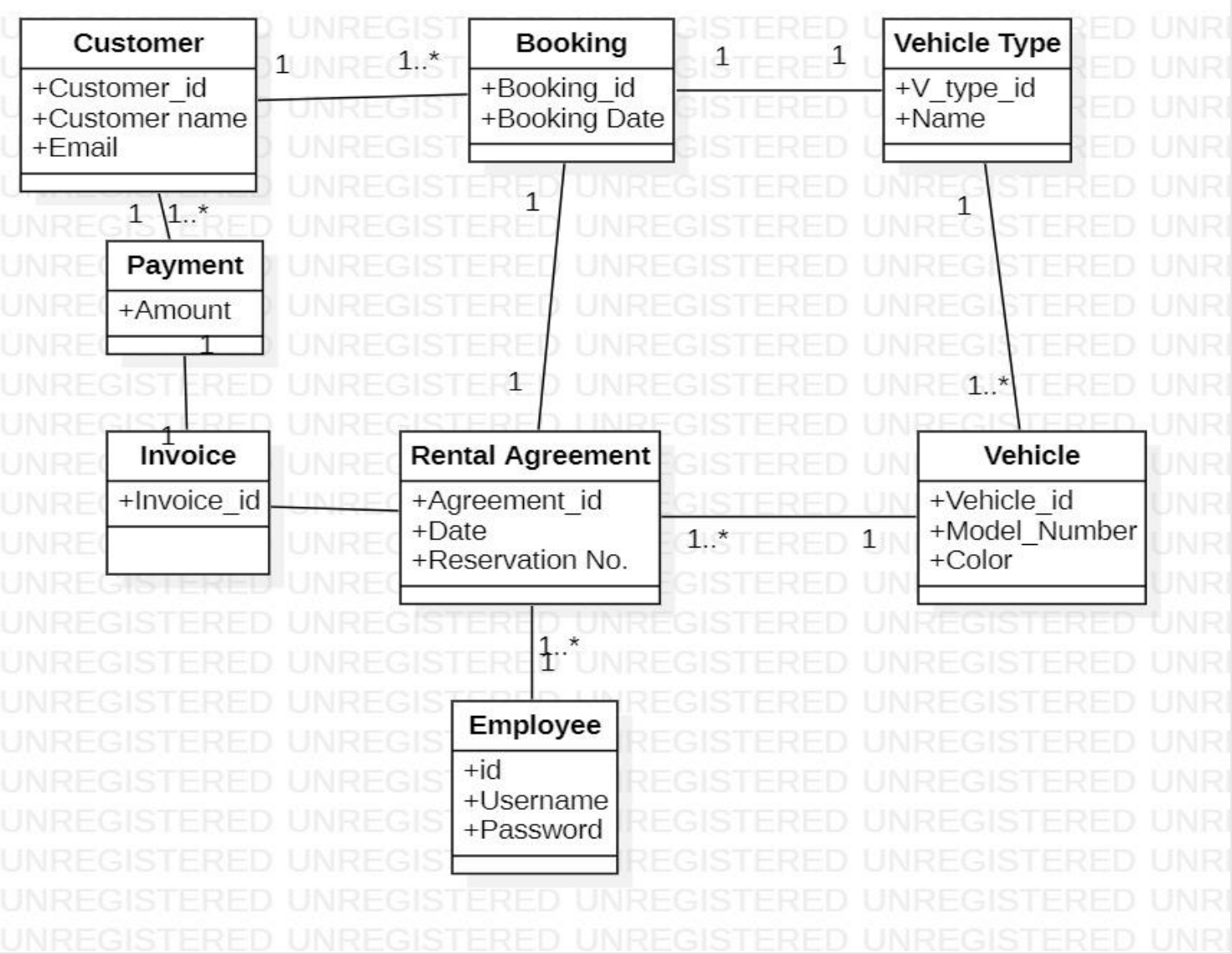


Building Conceptual Model

Add Attributes:

- It is useful to identify those attributes of conceptual classes that are needed to satisfy the information requirements of the current scenarios under development.
- An attribute is a logical data value of an object

Car Rental System



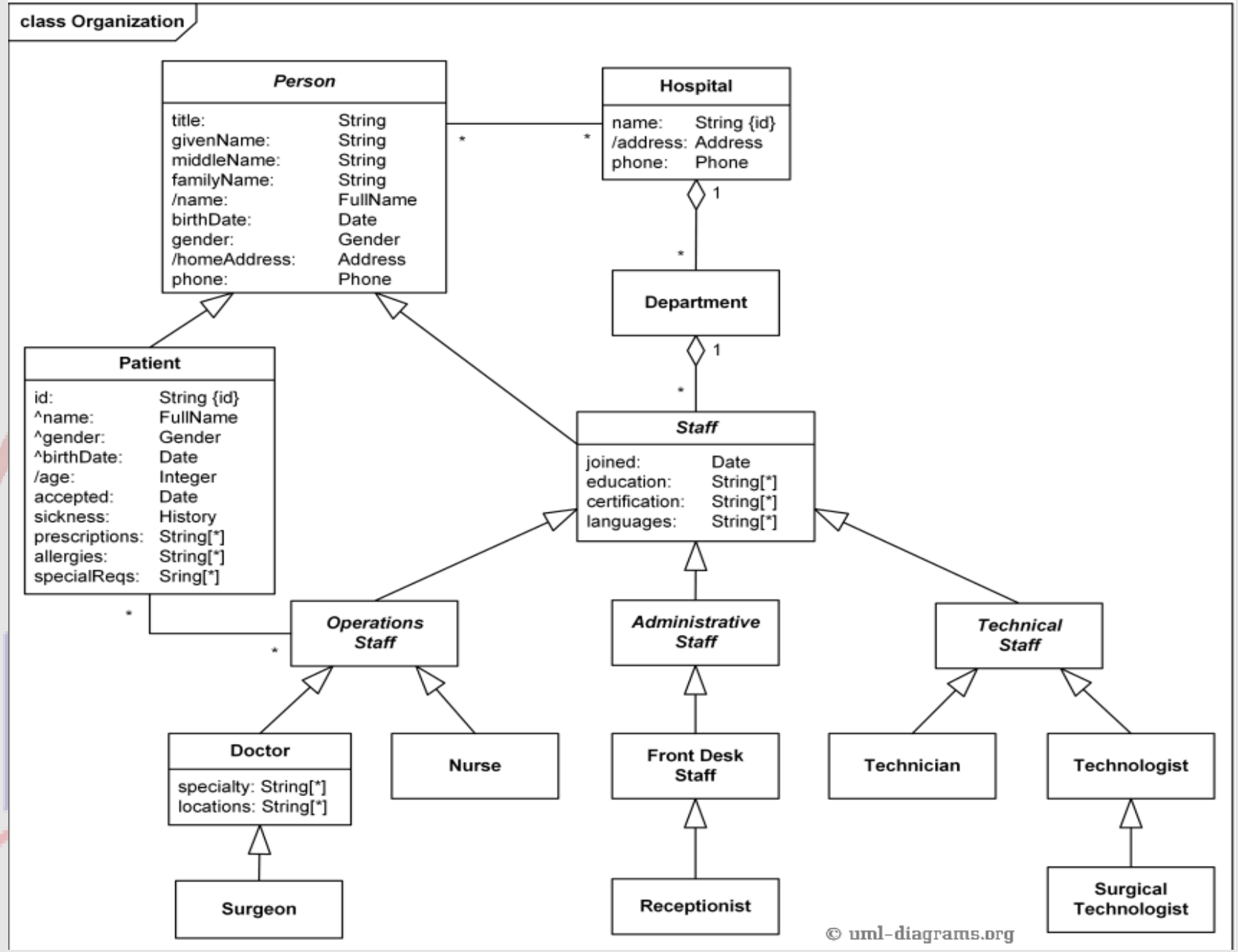


Fig: Hospital Management System


THANK
YOU