Software Engineering

Lecture 4

Lectures	Topics
1	Introduction to Software Engineering
2	Software Development Process (SDLC Activities) - SDLC Activity: Specification or Requirement Engineering - SDLC Activity: System Modeling/Design - SDLC Activity: Implementation - SDLC Activity: Testing - SDLC Activity: Evolution - SDLC Activity: Deployment/Installation - SDLC Activity: Maintenance
3	SDLC Activity: Requirement Engineering - Requirement Elicitation - Requirement Analysis and Management - Requirement Validation
4, 5, 6	SDLC Activity: System Modeling/Design - Context Modeling - Data Modeling - Structural/Architectural Modeling - Process Modeling - UI/UX Modeling
7,8,9	SDLC Activity: Implementation (Coding, tools, GIT – Version management, IDE, RESTFUL architecture)
10	SDLC Activity: Testing
11	SDLC Activity: Deployment (tools to deploy, cloud computing)
12	SDLC Activity: Maintenance

Agenda

- Different Programming Paradigms
  - Procedural Programming
  - Object Oriented Programming
- System Modeling
  - Context Model
    - Use Case Diagram
    - Context Diagram
    - Functional Decomposition Diagram (FDD)
  - Data Model
    - Entity Relationship Diagram (ERD)
    - Data Dictionary

(P) Pre-requisite Learning

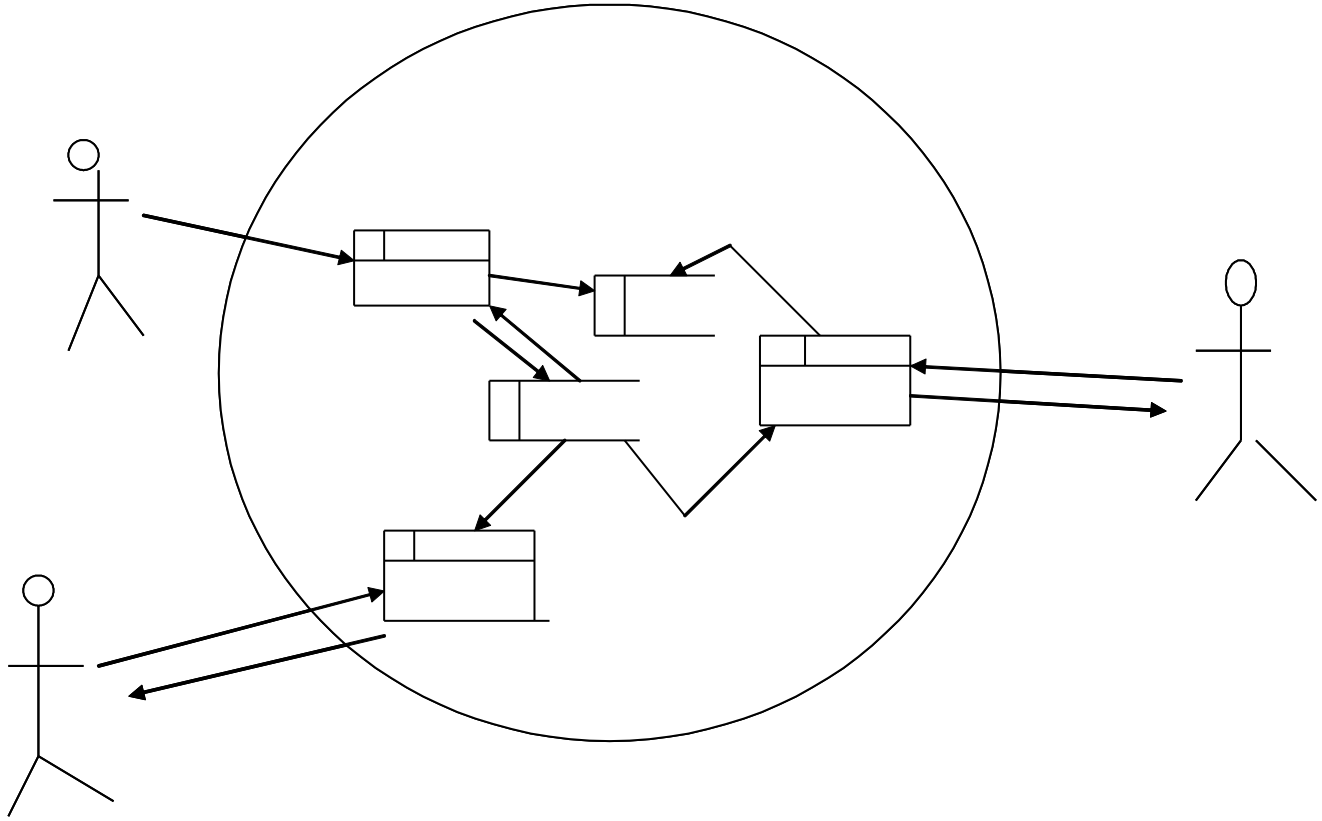
- Before we move into the ‘Modelling’ of the system, we should know the difference between object oriented programming and procedural programming. Different approach of programming takes different types of modeling techniques.

- Programming Approach/Paradigm
  - Procedural Programming
  - Object Oriented Programming

(P1) Structured / Conventional / Procedural Paradigm

- Structured/'conventional' paradigm
  - A system is viewed as a 'collection' of processes which operate on data entities ('stored' in data stores/files/database).
  - These processes correspond to various functional requirements, e.g. when we model a typical library system we are going to have such processes: *Borrow Book*, *Return Book*, *Reserve Book* etc. These processes operate on data entities Book, Loan, Reservation, etc.

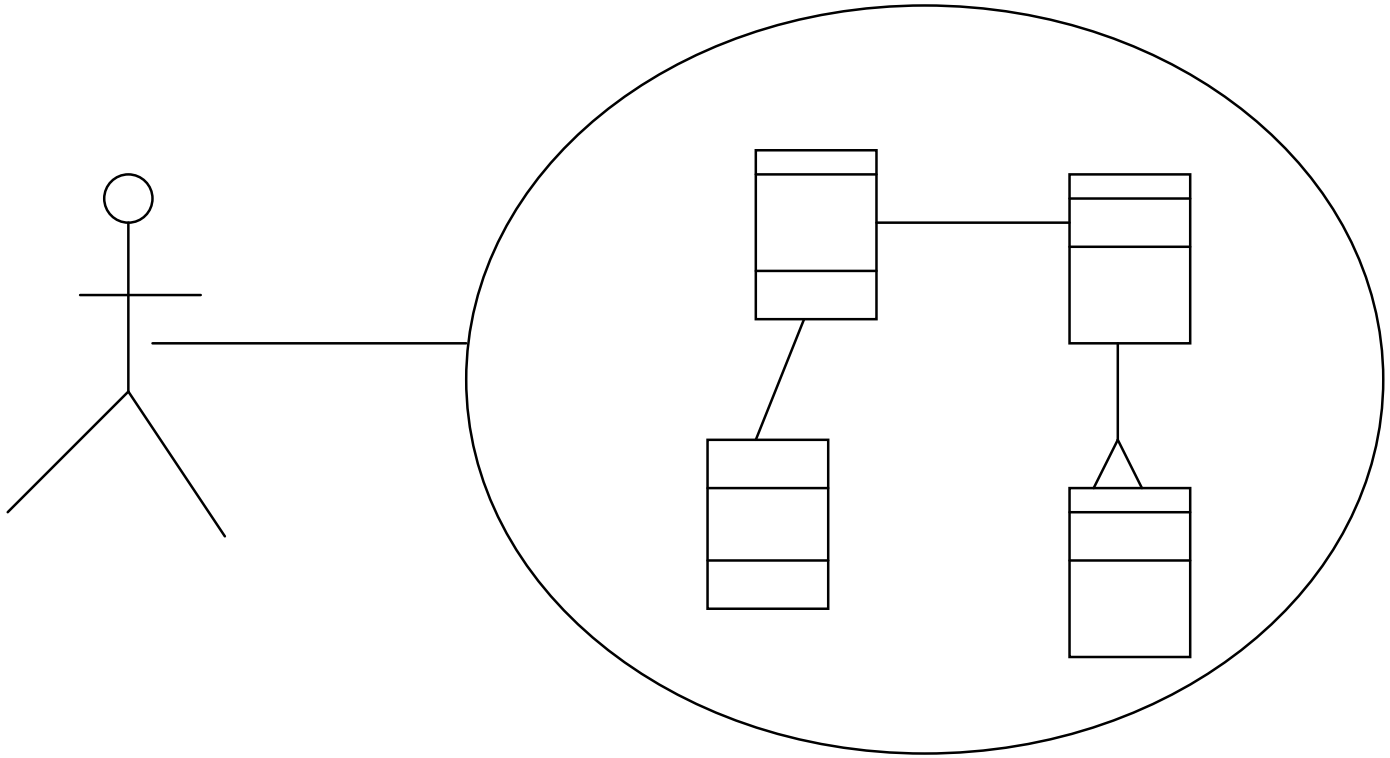
(P1) Structured/'Conventional' View of a System



(P2) Object Oriented Paradigm

- Object-oriented paradigm:
  - A system is viewed as a ‘collection’ of objects (class instances). Each object/class encapsulates its attributes (i.e. data) and operations.
  - Functional requirements (i.e. system functions) are ‘realised’ by collaborating objects. For example when we model a typical library system we are going to have such classes of objects: Book, Loan, Reader, etc. Function ‘Borrow Book’ will be ‘realised’ by corresponding operations of Book Object, Loan Object and possibly Reader Object. All these objects will collaborate to ‘realise’/process ‘Borrow Book’.

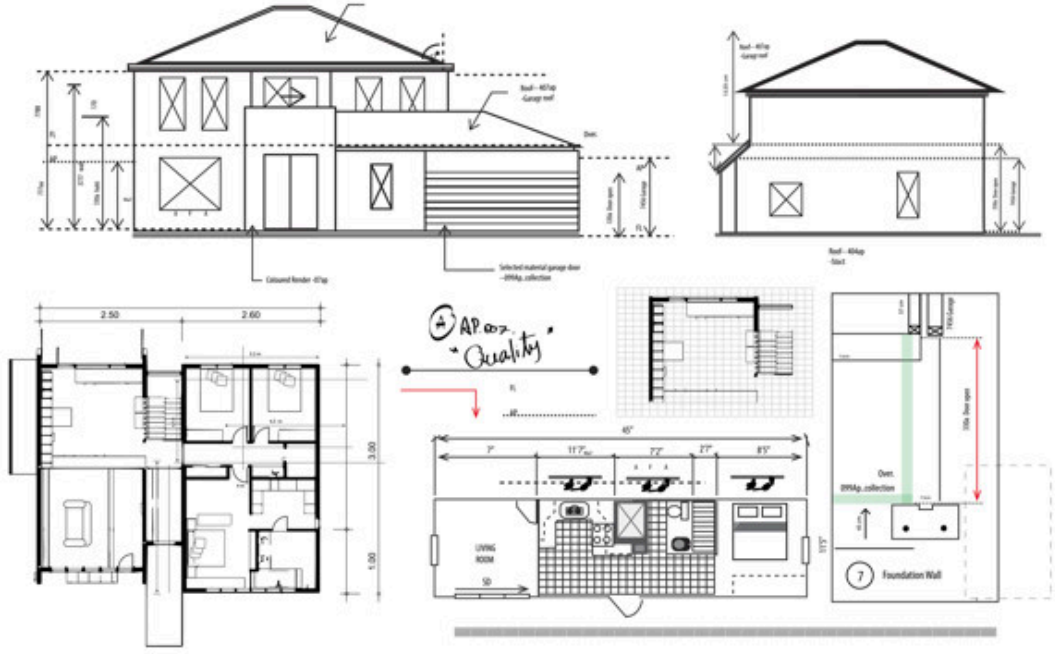
(P2) Object Oriented ‘View’ of a System



(P3) Procedural Programming vs OO Programming

4.0 System Design / Modeling

- Expansion of (Design Phase – part of SDLC)
- Converting textual requirements into graphical representation



4.0 System Design / Modeling

- System modeling is the process of **developing abstract models of a system**, with each model presenting a **different view** (data, process, interface) of the system.
- System modelling helps the analyst to understand the functionality of the system and models are used to communicate with customers and developers.

4.0 System Design / Modeling

System Modeling paradigms

- Structured/Conventional Paradigm (System viewed as a collection of process)
- Object Oriented Paradigm (system viewed as a collection of objects) [OUR FOCUS]
  - Representation in Graphical view using some standard notation. One of such **widely used graphical notation is UML** (due to the popularity of object oriented Design pattern)

Each of different paradigms have its own set of modelling techniques

4.0 System Design / Modeling

Following are the different types of modeling in software engineering (All of the modeling results in graphical representation that are in-line with the textual requirement gathered during Requirement Engineering phase of the generic SDLC activities)

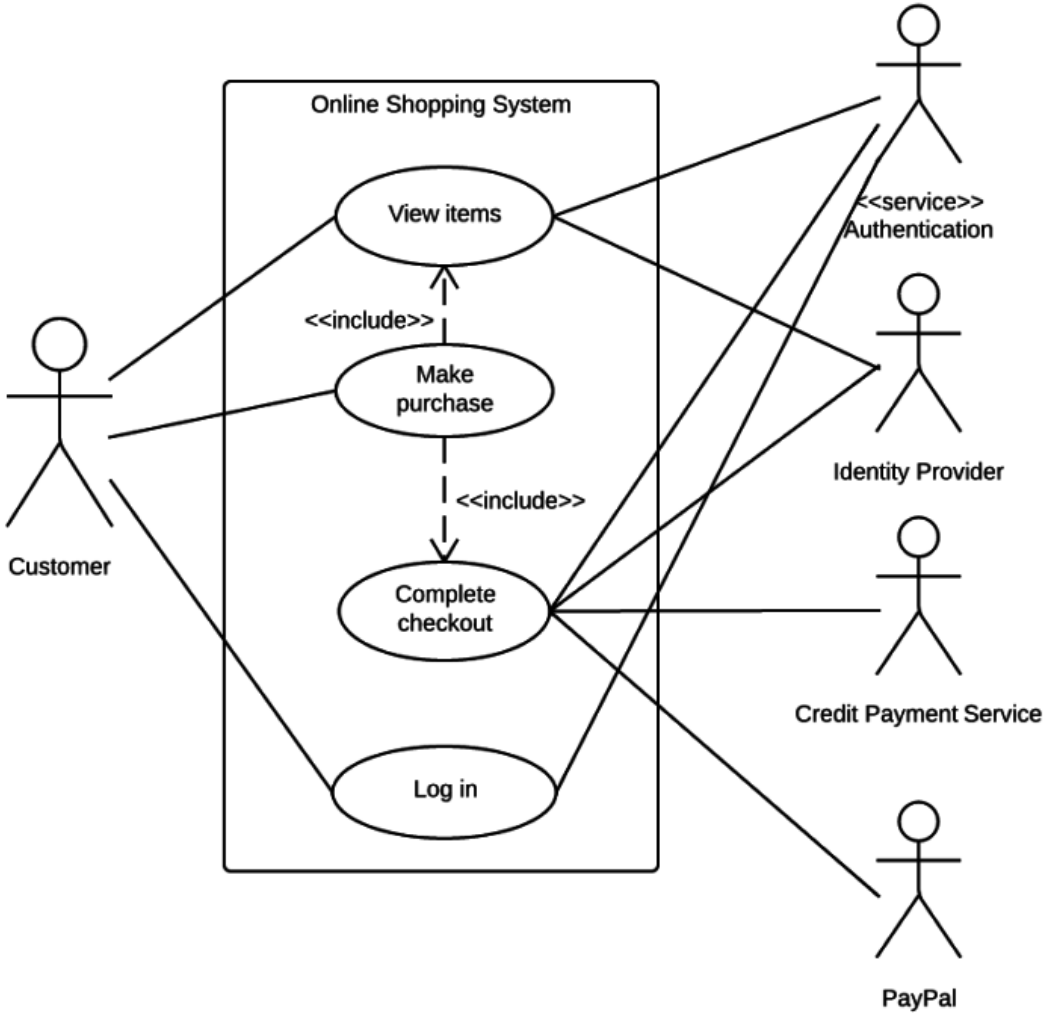
1. Context Modeling
2. Data Modeling
3. Structural / Architectural Modeling
4. Process Modeling
5. UI/UX Modeling

4.1 Context Models

- Context models are used **to illustrate the context of the system**. Context model consist of two main models:
  1. **Defines system boundaries** and establishes the relation with external entities
     - With the help of context Diagram (structured Paradigm) and Use case Diagram (Object Oriented Paradigm)
  2. **Shows the sub-system of the main system**
     - With the help of Functional Decomposition Diagram (FDD)

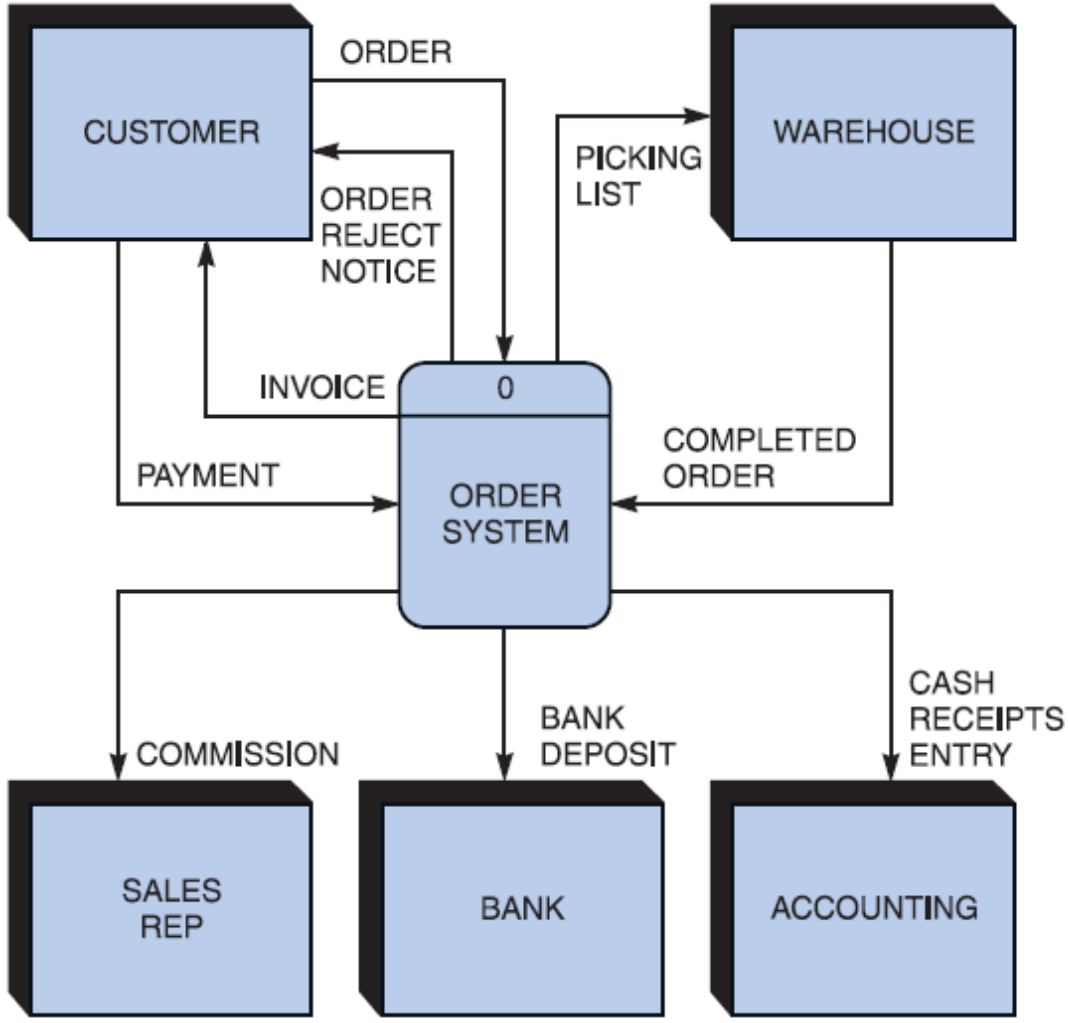
4.1.1 Context Model: Use Case Diagram

- Use Case Diagram: use case diagram visually represents the interaction between users and the information system
- Use case diagram is a modeling technique for context model for object oriented paradigm



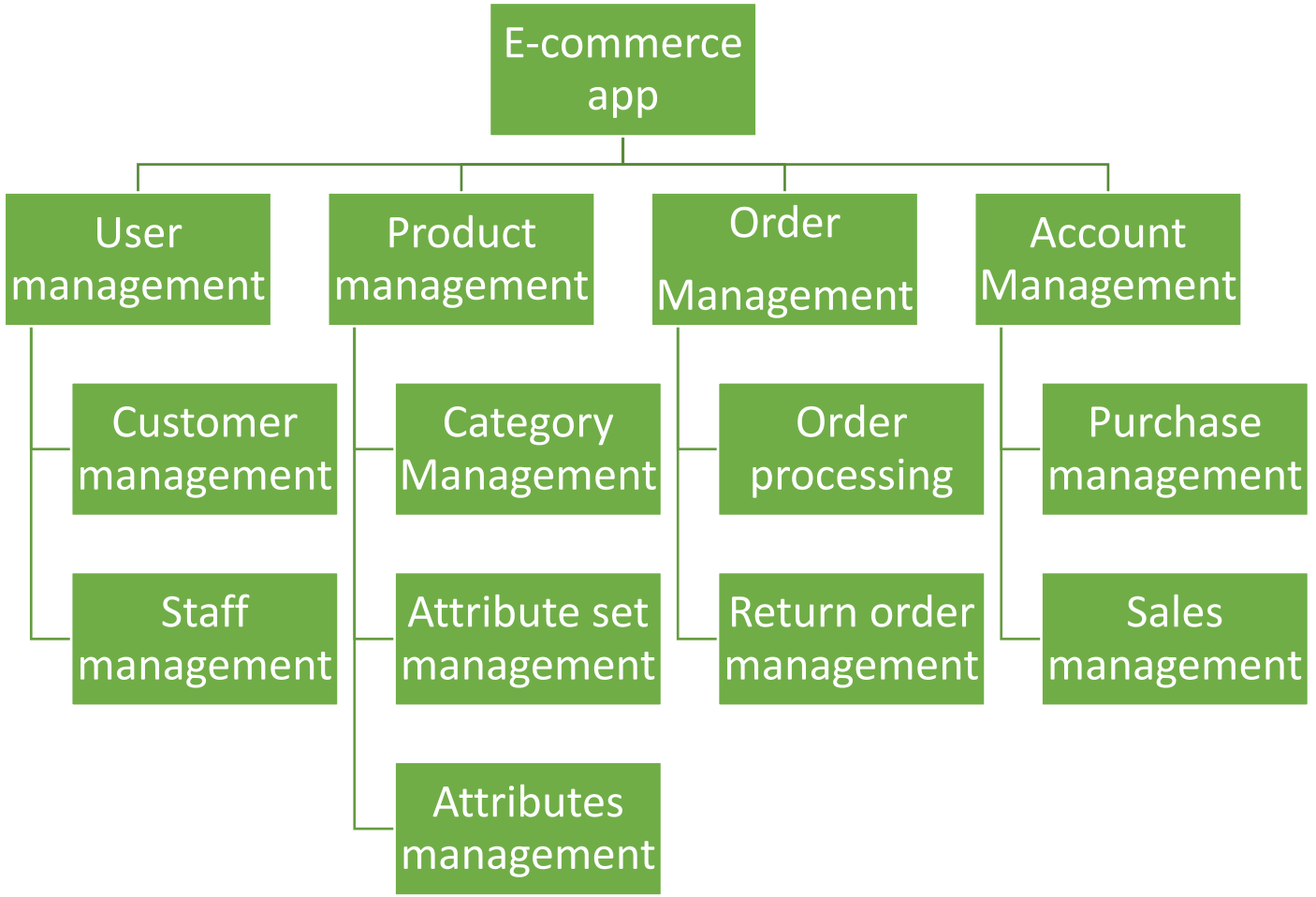
4.1.2 Context Model: Context Diagram

- Context Diagram
- Context diagram is a modeling technique for context model for Structured paradigm



4.1.3 Context Model: FDD

- Functional Decomposition Diagram
- Showing sub-system of the main systems
- Example of FDD
- FDD can be drawn during the Requirement Engineering phase too.



4.1 Context Models

In Summary,

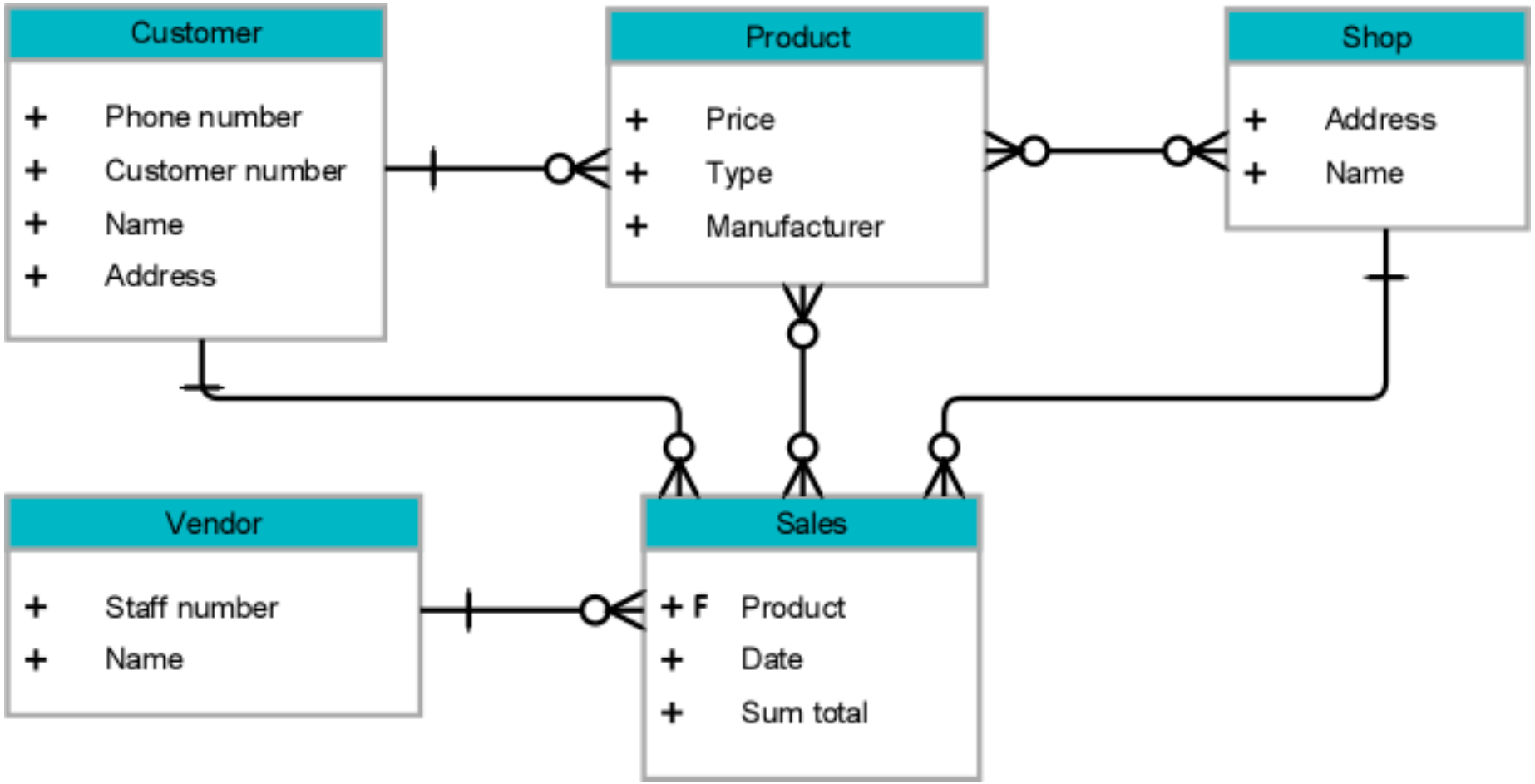
Paradigm	Context Modeling Technique
Structured Paradigm	- Context Diagram - FDD
Object Oriented Paradigm	- Use Case Diagram - FDD

4.2 Data Modeling / Data Design

Modeling Technique

- For both Paradigm (Structured and O.O): ERD, Data Dictionary

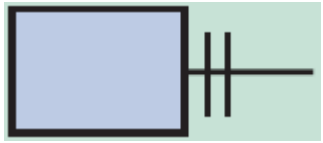
4.2.1 Data Modeling: ERD



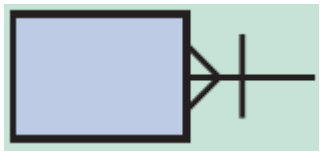
4.2.1.1 Data Modeling: ERD

Cardinality in ERD

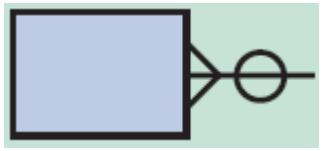
Crow's foot notation is a common method of indicating cardinality. The four examples show how you can use various symbols to describe the relationships between entities



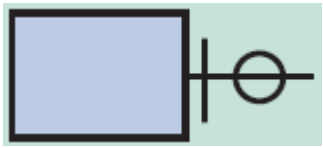
One and only One



One or Many



Zero, or one, or many



Zero or one

4.2.2 Data Modeling: Data Dictionary

- A set of information describing the contents, format, and structure of a database
- Summary, Describing the schema of a table

4.2.2 Data Modeling: Data Dictionary Format

- Table Name: Employee
  - Fields: name, address, phone_number, gender, dob

Fields	Description	Data Type	Character Length	Default Value	Required	Accept Null Value
name	Stores the name of employee.	VARCHAR	9		Yes	No
gender	Stores the gender information	ENUM(“M”,”F”)	1		No	Yes
dob	Stores the date of birth	DATE	15		Yes	no

Workshop Excercise

- Context Model
  - Use Case Diagram
  - Context Diagram
  - Functional Decomposition Diagram (FDD)
- Data Model
  - Entity Relationship Diagram (ERD)
  - Data Dictionary