(CT-21015)
Software Engineering
Mini Project Stage- II

Unit II-Requirement Engineering

Unit II-Requirement Engineering

Contents

- Requirements Engineering
- Requirement engineering process
- Introduction to Analysis model

Requirements Engineering

First step of SDLC

Software Requirements,

Business Requirements,

Functional & Non-functional Requirements,

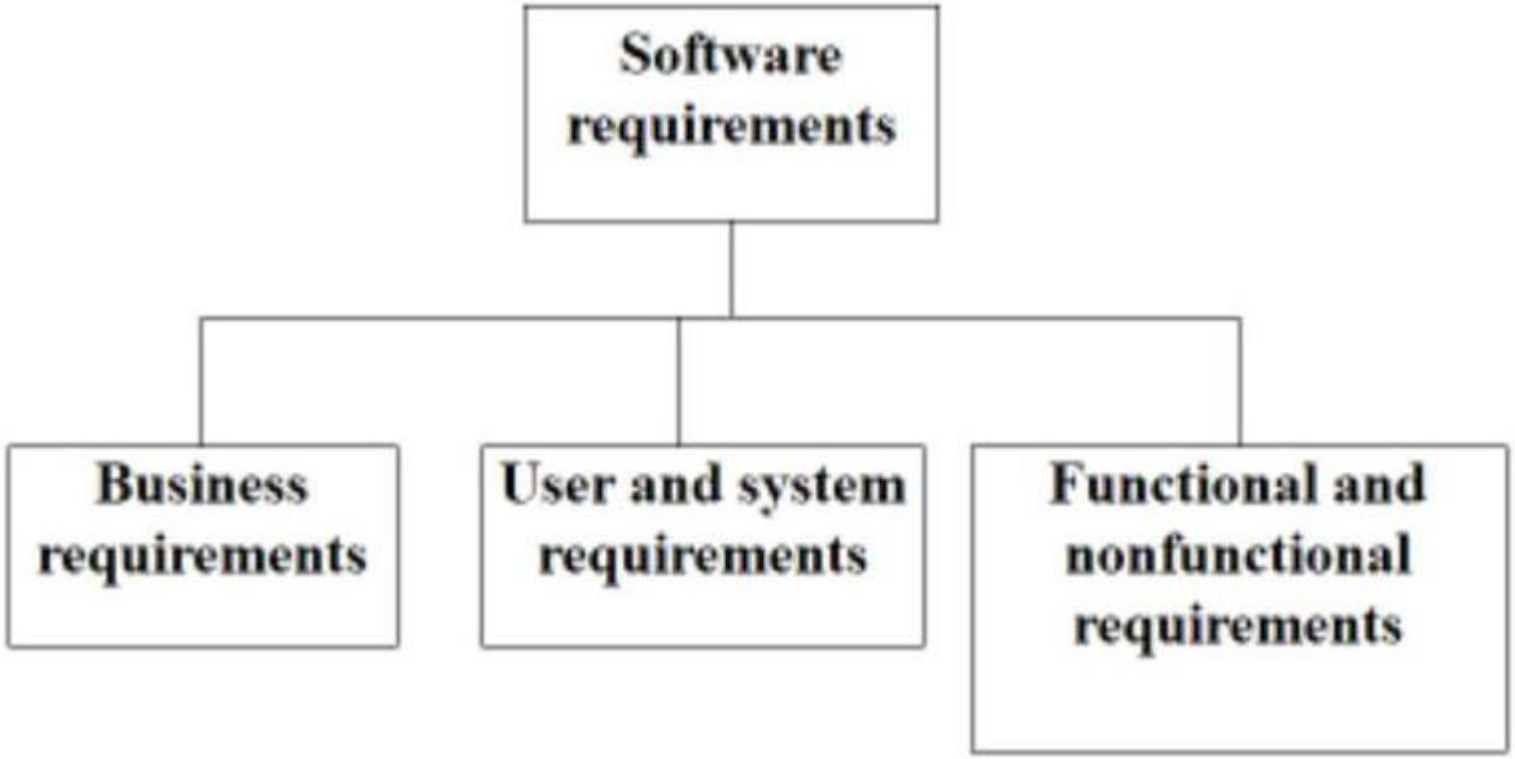
System & User Requirements

Introduction

- **Requirements engineering** is the process of gathering, analyzing, documenting, validating, and managing requirements.
- The **main goal** of requirements engineering is **to** clearly understand the customer requirements and systematically organize these requirements in the SRS.
- The requirements collected from the customer are organized in some systematic manner and presented in the formal document called software requirements specification **(SRS)** document.

Software Requirements

- A software requirement is a detailed, formal description of system functionalities.
- It specifies a function that a system or component must be able to perform for customer satisfaction.



Business Requirements

- Business requirements **define the project goal** and the expected business benefits for doing the project.
- Understanding the **business rules** or **the processes of organization** is vital to software development.
- The enterprise mission, values, priorities, and strategies must be known to understand the business requirements that cover higher level data models and scope of the models.
- The **business analyst** guides the client through the complex process that elicits the requirements of their business.

User Requirements

- User requirements are the **high-level abstract statements** supplied by the customer, end users, or other stakeholders.
- These requirements are **translated into** system requirements keeping in mind user's views.
- These requirements are generally **represented in** some natural language with pictorial representations or tables to understand the requirements.
- User requirements **may be ambiguous or incomplete** in description with less product specification and little hardware/software configurations are stated in the user requirements.
- In an ATM machine, user requirements allow users to withdraw and deposit cash.

System Requirements

- These are considered as a **contract between** the client and the development organization.
- System requirements are the **detailed and technical functionalities** written in a systematic manner that are implemented in the business process to achieve the goal of user requirements.
- The system requirements mean a **more detailed description** of the system services and the operational constraints such as how the system will be used and development constraints such as the programming languages.
- This level of detail is needed by those who are involved in the system development, like engineers, system architects, testers, etc.
- The system requirements consider customer ID, account type, bank name, consortium, PIN, communication link, hardware, and software. Also, an ATM will service one customer at a time.

Functional Requirements

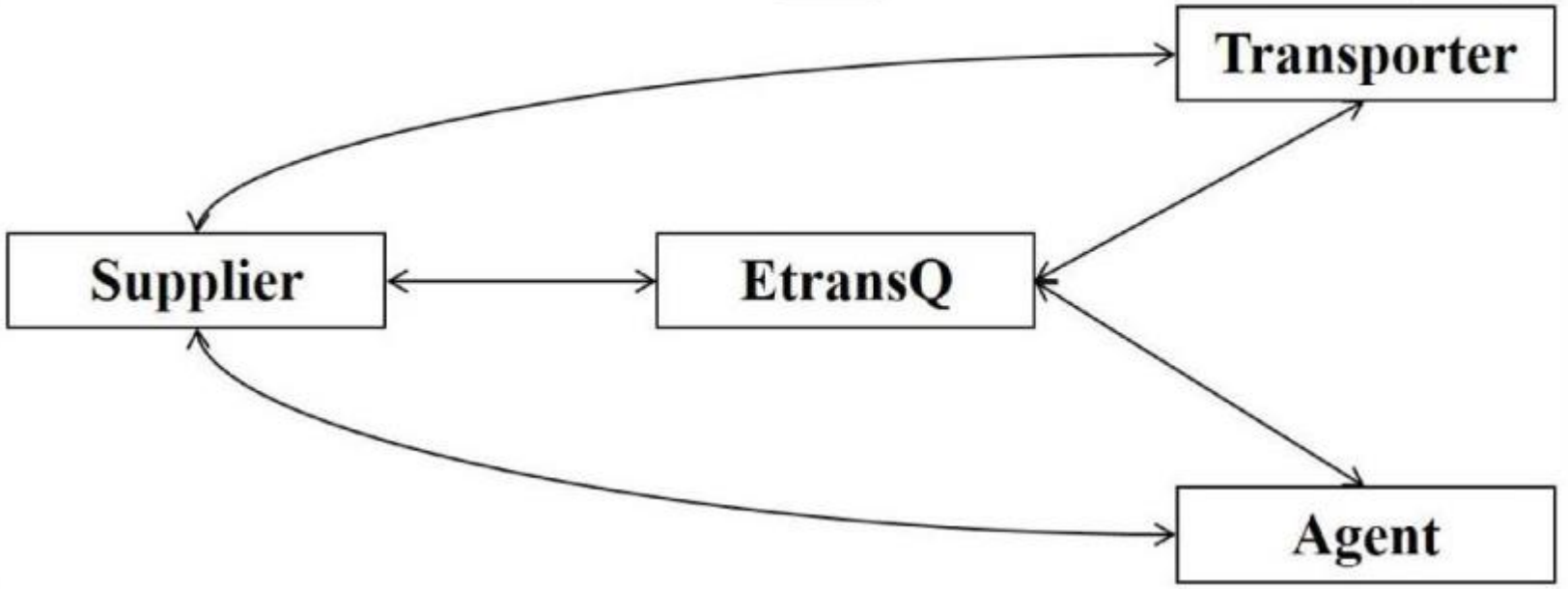
- Functional requirements are the **behavior or functions** that the system must support.
- It **covers the main functions** that should be provided by the system. When expressed as user requirements, they are usually described in an abstract way.
- However, more specific functional system requirements describe the system functions, its inputs, processing; how it’s going to react to a particular input, and what’s the expected output.

Non-functional Requirements

- Non-functional requirements **specify how a system must behave**.
- Non-functional requirements are related to functional requirements, i.e., how efficiently, by how much volume, how fast, at what quality, how safely, etc., a **function is performed** by a particular system.
- The examples of non-functional requirements are reliability, maintainability, performance, usability, security, scalability, capacity, availability, recoverability, serviceability, manageability, integrity, and interoperability.

Example: ETransQ

Fig: Business Architecture for EtransQ



It is a web-based information system for suppliers, transporters, and agents to share a common platform and help in gaining profit.

Business requirements for ETransQ

- The objectives of EtransQ are to
  - ✓ provide the common platform for suppliers, transporters, and agents
  - ✓ provide an easy way to search for tenders and offers
  - ✓ build trust and relationships between consumers and service providers
  - ✓ provide easy and improved way of communication; and
  - ✓ provide enough options to get the best deal available in the market

User and system requirements for ETransQ

- EtransQ system should **provide a common platform** for suppliers, agents, and transporters **to share knowledge** that will **help them to grow their business**.
- The **system requirements** in this system are:
  - The user should be able to communicate with other users connected with the system irrespective of their physical location
  - The system should be flexible enough to provide personalized search results for every registered user.
  - The system should work in a secured manner.
  - The information available should be genuine and reliable.

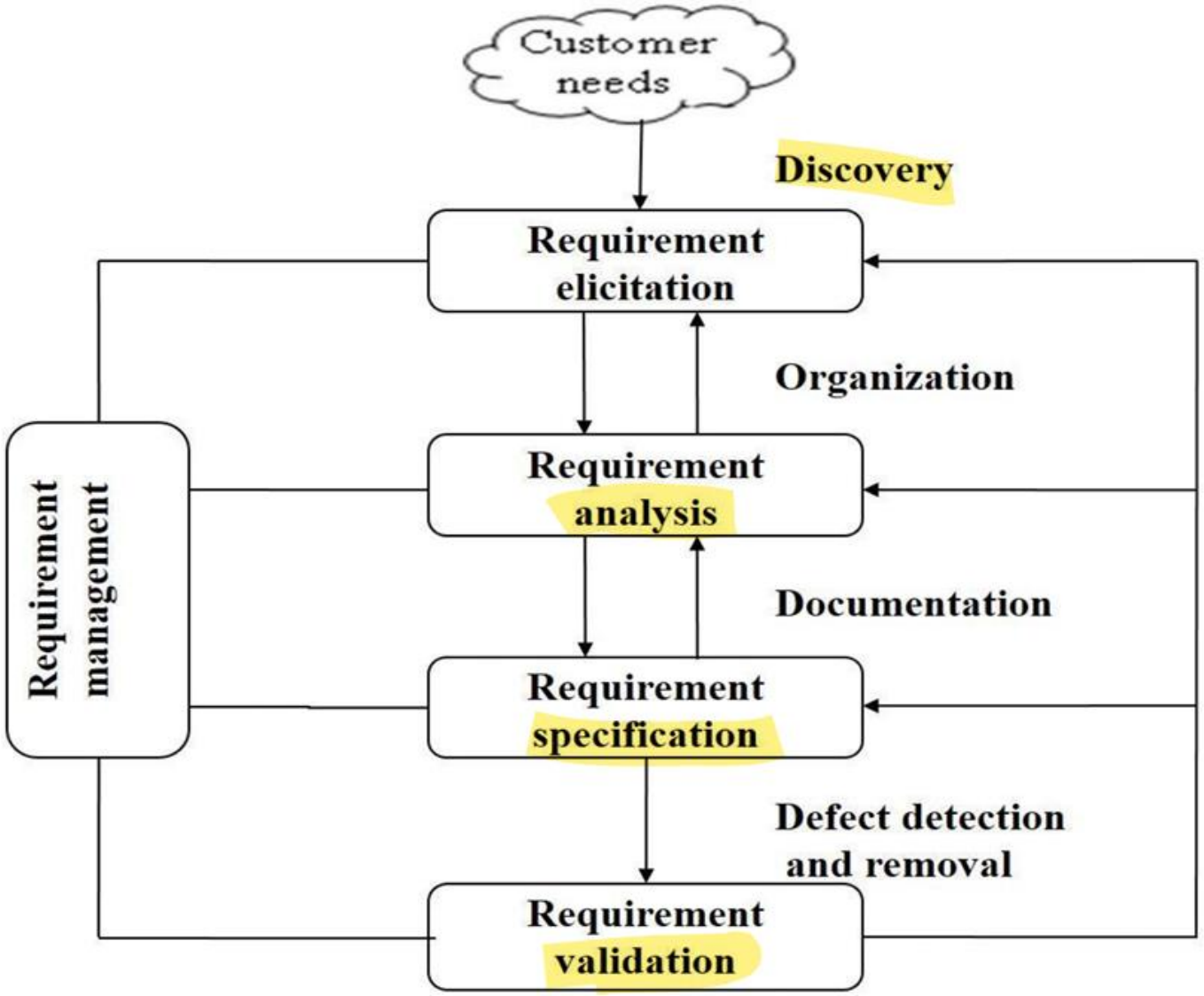
Functional Requirements for ETransQ

- EtransQ system provides
  - registration,
  - association,
  - quick search,
  - tendering,
  - offering, and
  - testimonial services to its user.
  - etc.

Non-functional Requirements for ETransQ

- Availability
- Recoverability
- Extensibility
- Maintainability
- Privacy
- Security
- Compatibility
- Usability
- Stability
- Reusability
- Robustness

Requirement Engineering Process...



The main issues involved in requirement engineering are

- ✓ **Discovering** the requirements
- ✓ Technical **organization** of the requirements
- ✓ **Documenting** the requirements
- ✓ Ensuring **correctness and completeness** of the requirements
- ✓ **Managing** requirements that changes over time.

- A typical requirement engineering process has **two main aspects**:
  - Requirement development
  - Requirement management

- **Requirement Development** includes various activities, such as elicitation, analysis, specification and validation of requirements.
- **Requirement Management** is concerned with managing requirements that change dynamically, controlling the baseline requirements, **monitoring the commitments and consistency of requirements throughout software development.**

1. Requirements Elicitation

- The **developers work** closely with the customers and stakeholders to **gather the requirements** needed for the development of the system.

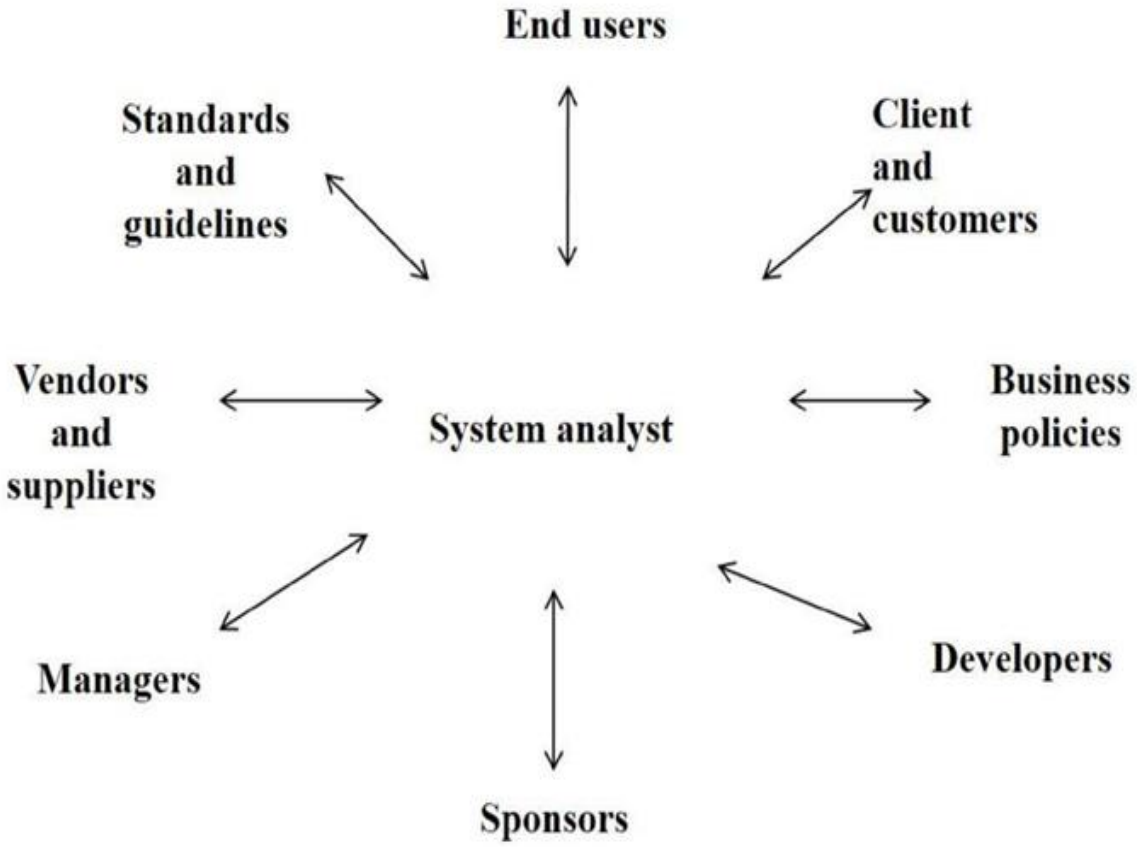
- There are several **Fact-Finding techniques** that can be used to elicit requirements, including:
  - **Interviews:** These are one-on-one conversations with stakeholders to gather information about their needs and expectations.
  - **Surveys:** These are questionnaires that are distributed to stakeholders to gather information about their needs and expectations.
  - **Focus Groups**: These are small groups of stakeholders who are brought together to discuss their needs and expectations for the software system.
  - **Onsite Observation:** This technique involves observing the stakeholders in their work environment to gather information about their needs and expectations.
  - **Prototyping:** This technique involves creating a working model of the software system, which can be used to gather feedback from stakeholders and to validate requirements.

A **system analyst** is the person who interacts with different people, understands the business needs, and has knowledge of computing.

System analyst

- The **role of the system analyst** is multifunctional, fascinating, and challenging as compared to other people in the organization.
- A system analyst is the **person who interacts** with different people, understands the business needs, and has knowledge of computing.
- The **skills of the system analyst** include programming experience, problem solving, interpersonal savvy, IT expertise, and political savvy. He acts as a broker and needs to be a team player.
- He should also have good communication and decision-making skills.
- He should be able to motivate others and should have sound business awareness.

Fig: System Analyst Interaction



Challenges in requirements elicitation

- The **main challenges** of requirements elicitation are
  - ✓Identification of problem scope
  - ✓Identification of stakeholders
  - ✓Understanding of problem
  - ✓Volatility of requirements.
- Stakeholders are generally unable to express the complete requirement at a time and in an appropriate language.
- It may happen that the analyst is not aware of the problem domain and the business scenario.
- To face these challenges, analyst will **follow some techniques** to gather the requirements, which are called as **Fact-Finding Techniques**.

2. Requirements Analysis

- The requirement engineers **process the information** gathered from the stakeholders, classify them in various categories, and relate the customer needs to possible requirements.
- It **includes various activities**, such as classification, organization, prioritization, negotiation, and modeling requirements.
- The **draft requirements are analyzed to** verify their correctness and completeness
- The following **analysis techniques** are generally used for the modeling of requirements:
  - Structured analysis
  - Data-oriented analysis
  - Object-oriented analysis
  - Prototyping

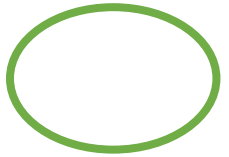
1. Structured Analysis

- Also **referred as** process modeling or data flow modeling.
- It focuses on transforming the documented requirements in the form of processes of the system.
- During transformation, it **follows a top-down functional decomposition process** in which the system is considered a single process and it is further decomposed into several sub-processes to solve the problem.
- Thus, the **aim** of structured analysis is to **understand the work flow of the system** that the user performs in the existing system of the organization.
- Structured analysis uses a graphical tool called **data flow diagrams (DFD),** which represent the system behavior.

Data Flow Diagram (DFD)

- A DFD is a graphical tool that describes the flow of data through a system and the functions performed by the system.
- It shows the processes that receive input, perform a series of transformations, and produce the desired outcomes.
- It does not show the control information (time) at which processes are executed.
- DFD is also called a bubble chart or process model or information flow model.
- **A DFD has four different symbols:**

✓ **Process:** A process is represented by a circle and it denotes transformations of the input data to produce the output data.

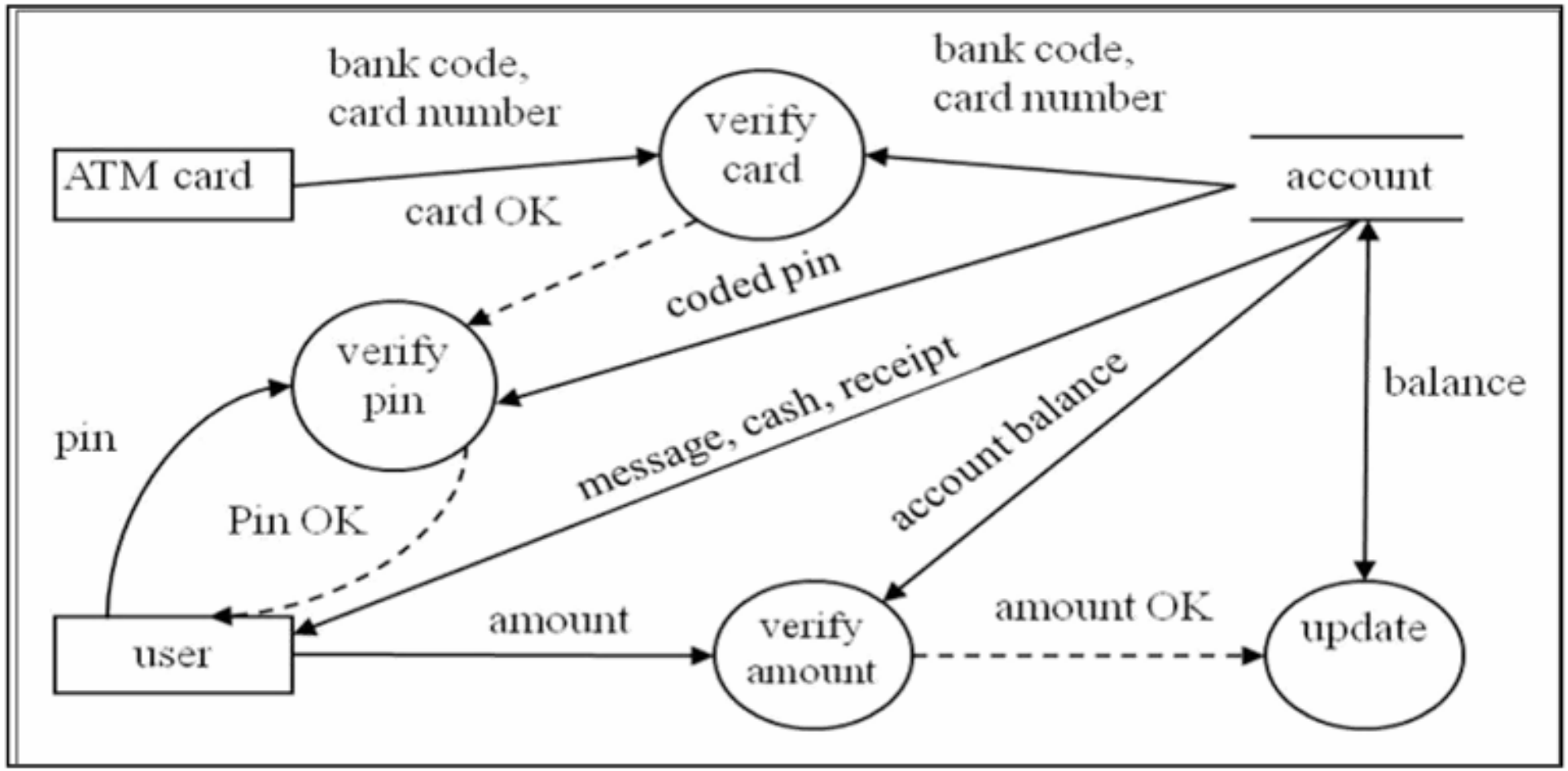


✓ **Data flow:** represent the movement of data, i.e., leaving one process and entering into another process. Data flows are represented by arrows, connecting one data transformation to another.

✓ **Data store:** Data store is the data at rest. It is represented in parallel lines.

✓ **Entity/Actor:** It is the external entity that represents the source or sink (destination of data). It is represented by a rectangle.

Fig: **DFD for ATM Withdrawal**



DFD Levels

- DFDs can range from simple overviews to complex, granular representations of a system or process with multiple levels, starting with level 0.
- The most common and intuitive DFDs are level 0 DFDs, also called context diagrams. They're digestible, high-level overviews of the flow of information through a system or process, so almost anyone can understand it.
- On the other extreme, level 3+ diagrams contain lots of detail and complexity.

- **Level 0: Context Diagram**

This DFD level focuses on high-level system processes or functions and the data sources that flow to or from them. Level 0 diagrams are designed to be **simple, straightforward overviews of a process** or system.

- **Level 1: Process Decomposition**

While level 1 DFDs are still broad **overviews of a system or process**, they're also more detailed — they break down the system's single process node into subprocesses.

- **Level 2: Deeper Dives**

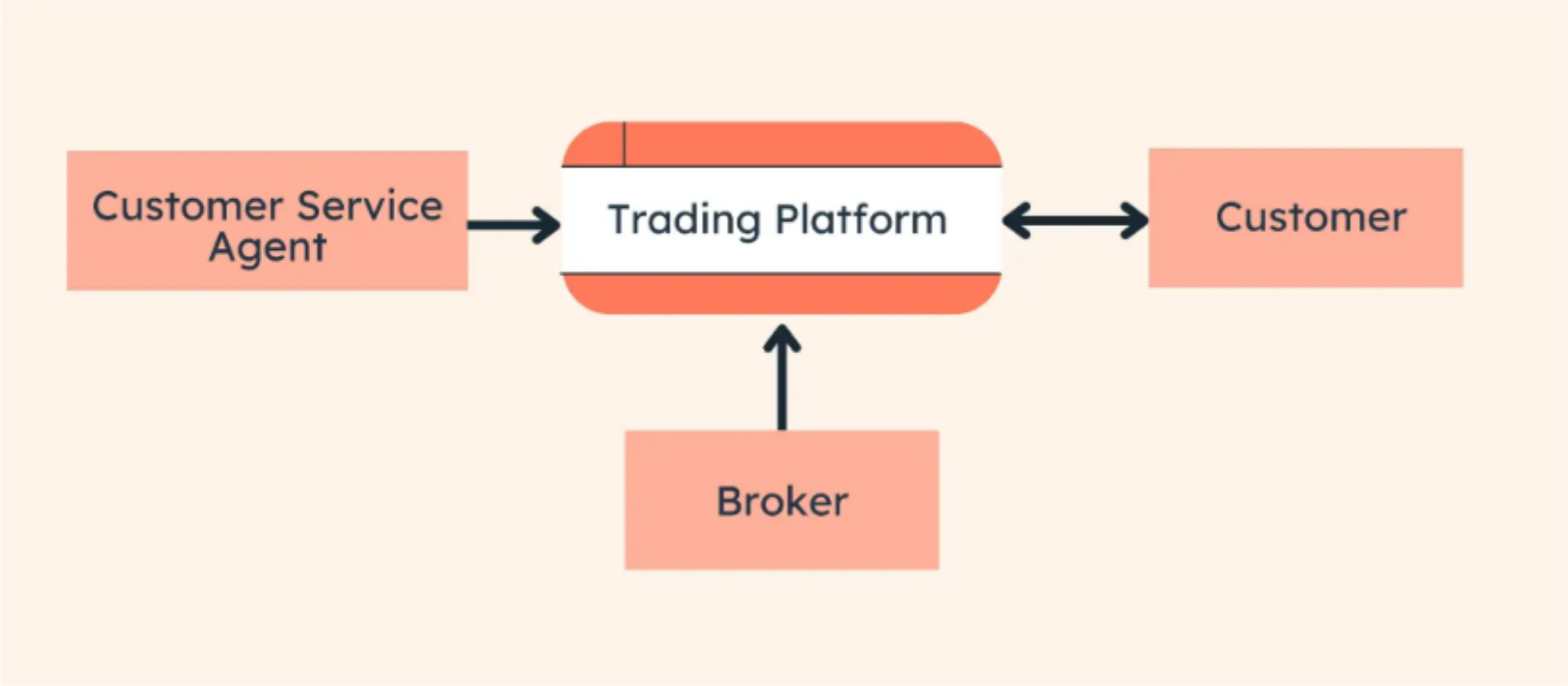
The next level of DFDs dives even deeper into **detail by breaking down each level 1 process** into granular subprocesses.

- **Level 3: Increasing Complexity**

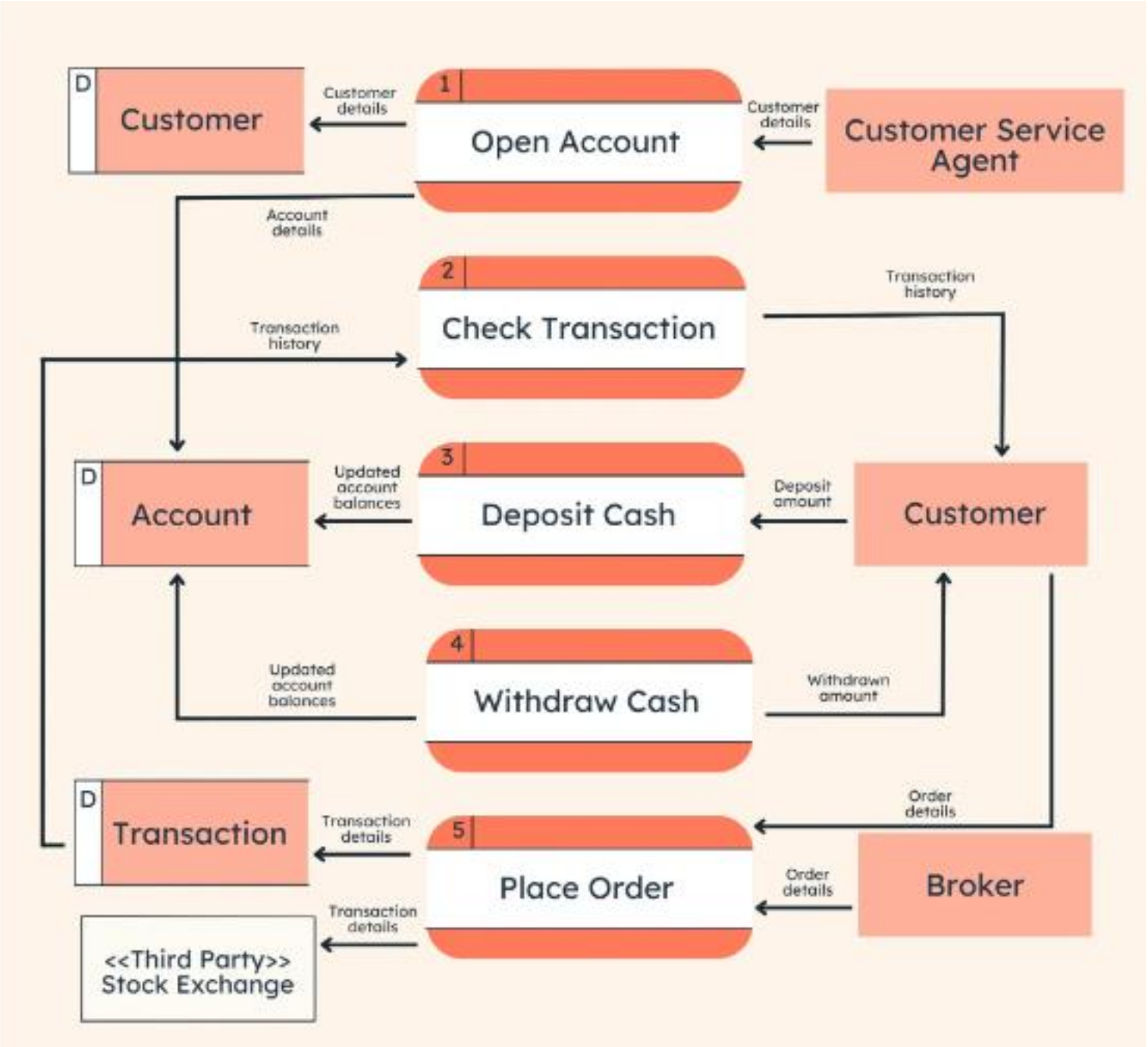
Level 3 and higher-numbered DFDs are uncommon. This is largely due to the amount of detail required, which **defeats its original purpose** of being easy to understand.

Data Flow Diagram Example-1

1. Level 0 DFD



This Level 0 DFD provides a contextual map of a securities trading platform.
Data flows in one direction from the customer service assistant and the broker to the platform. It also flows in two directions from customers to the platform and back again.

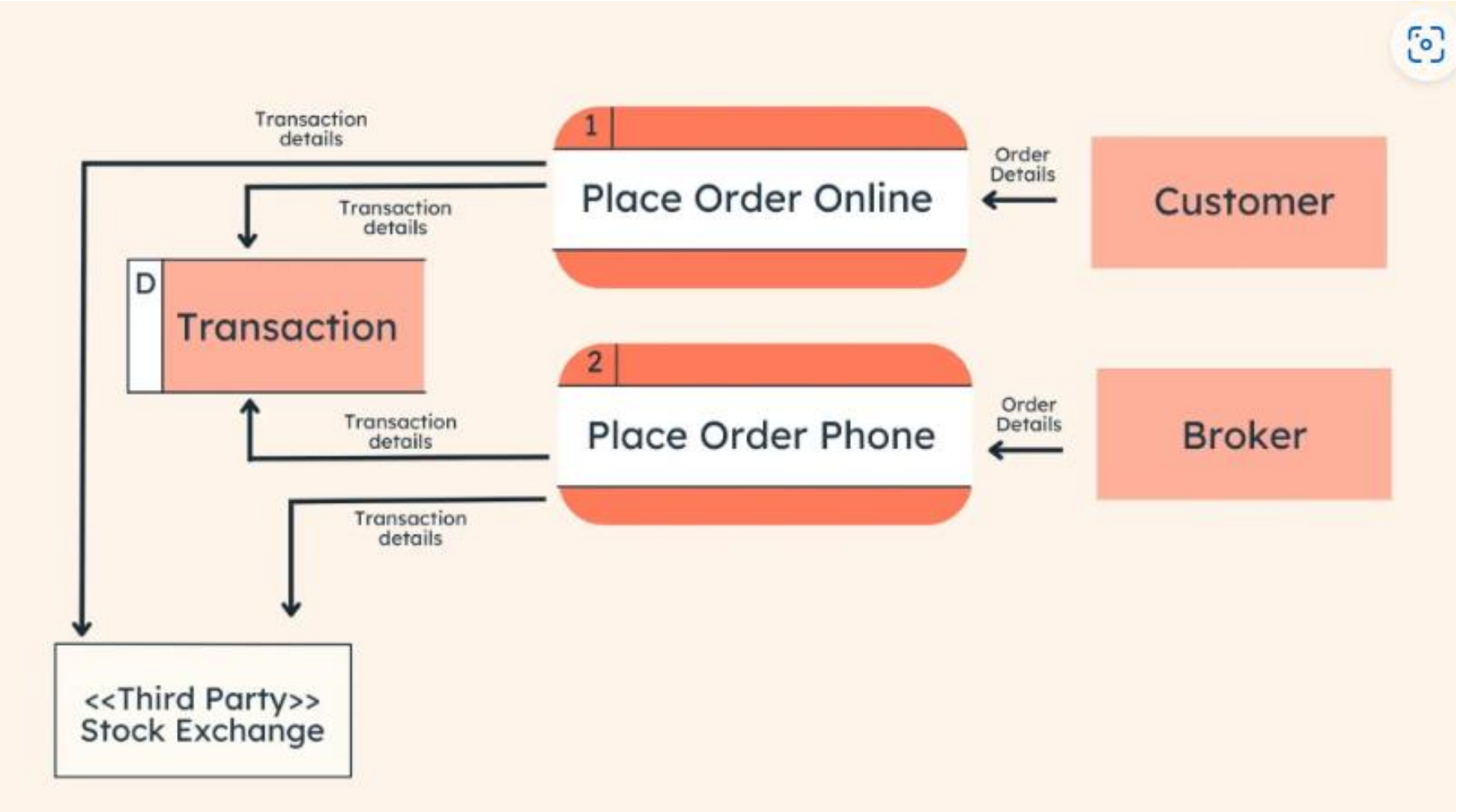


2. Level 1 DFD

This Level 1 DFD breaks down the customer process in more detail, expanding it to include account creation, cash withdrawals, and eventual securities transactions.

As you can see, it breaks down the customer interactions into more specific actions, allowing viewers to grasp what the whole process looks like.

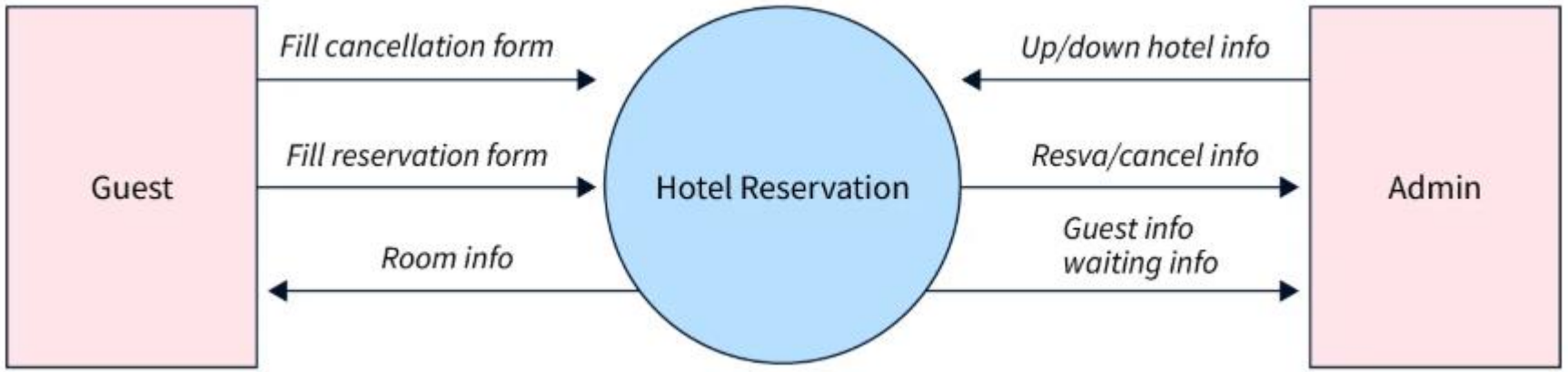
3. Level 2 DFD



This Level 2 DFD decomposes the “Place Order” process to contextualize the steps required to place an order — either by a customer or by a broker.
It even accounts for a third-party stock exchange center where transaction details are forwarded after an order is placed. This provides a more granular depiction of a specific process.

Data Flow Diagram Example-2

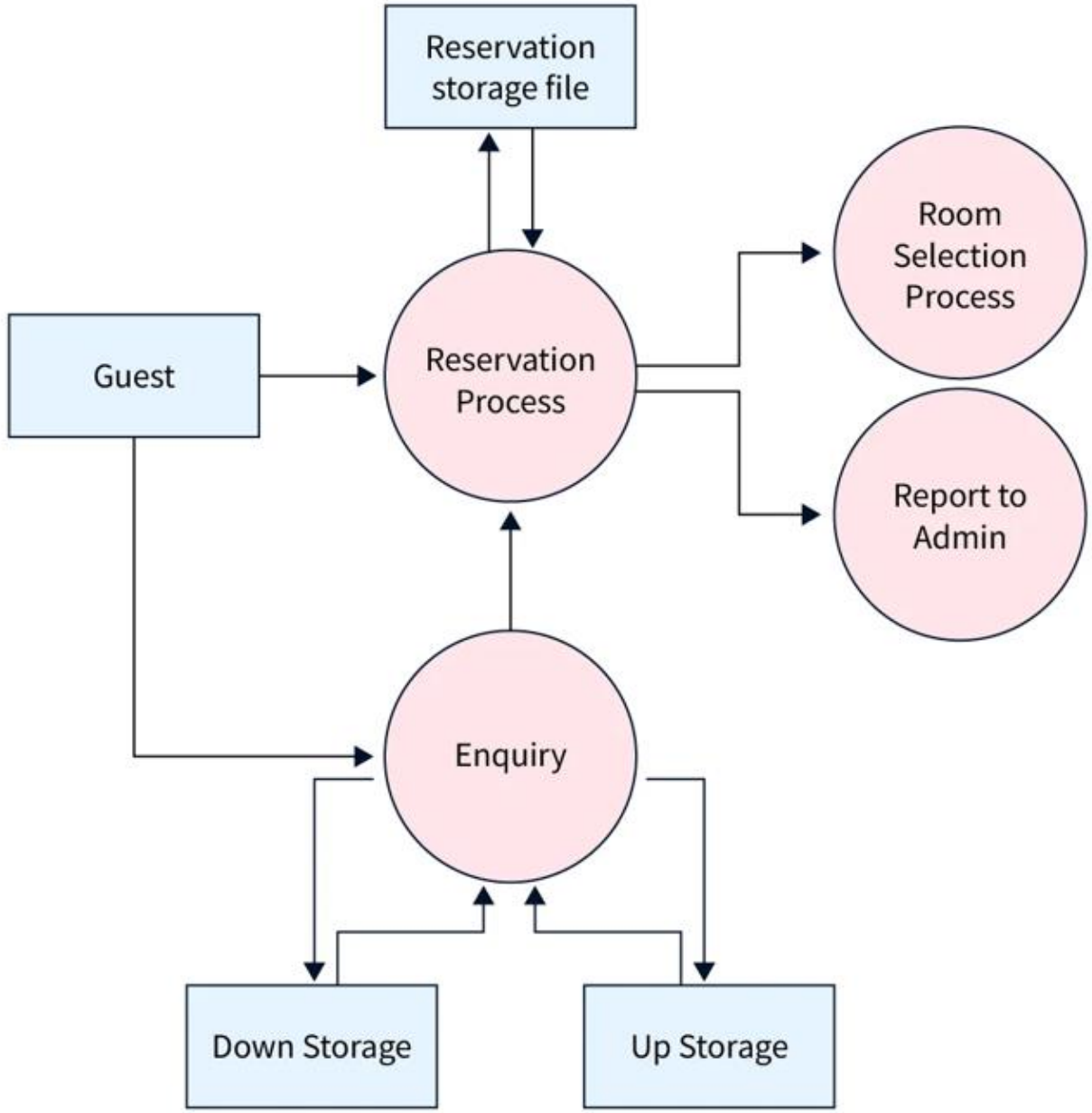
0-Level DFD



It represents the complete system as a single bubble with incoming and outgoing arrows indicating input and output data. Here's an example of a 0-level DFD.

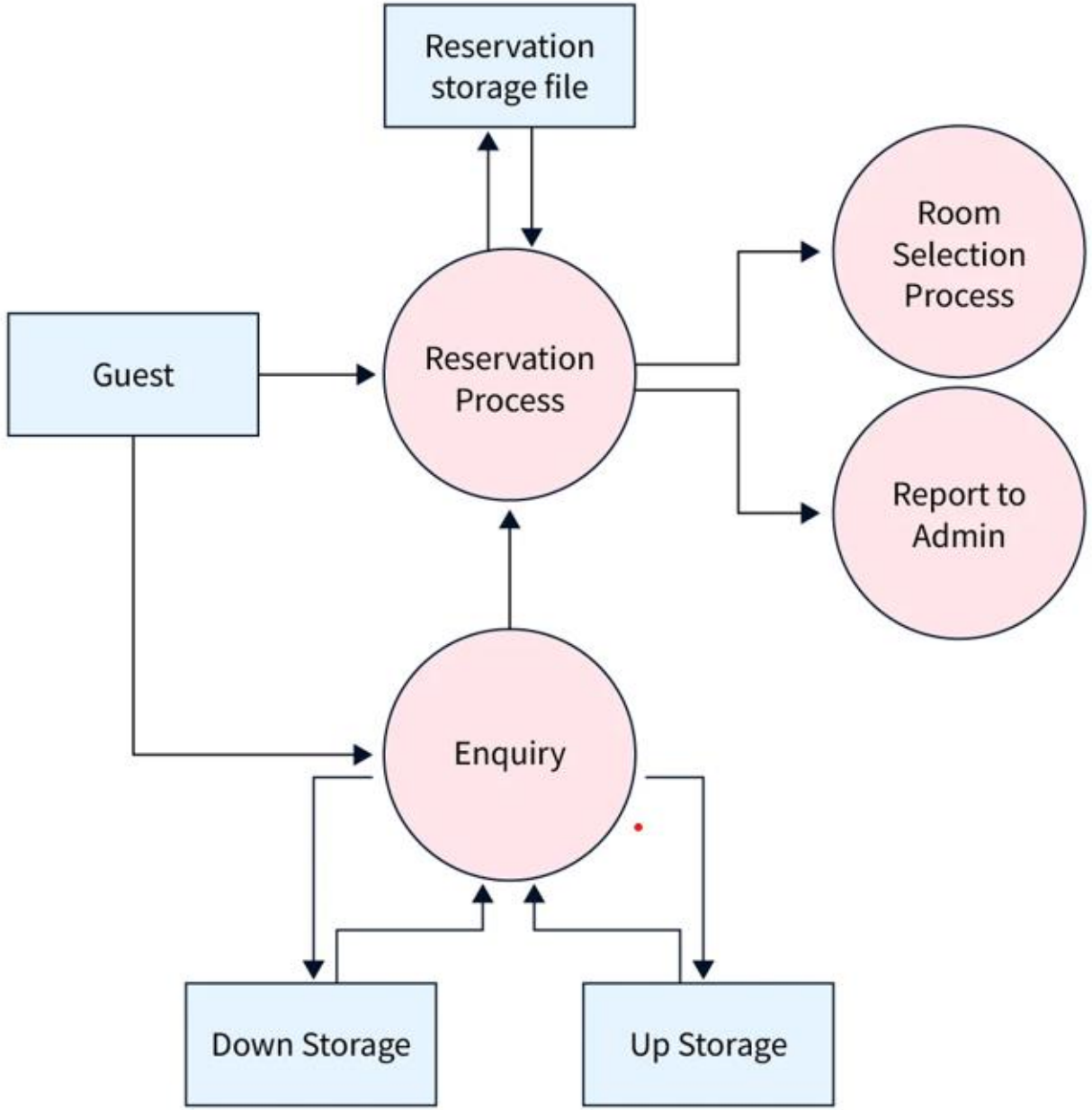
1-Level DFD

At this level, we highlight the system's essential functions and divide the high-level process of 0-level DFD into subprocesses. Here's an example of a 1-level DFD.



2-Level DFD

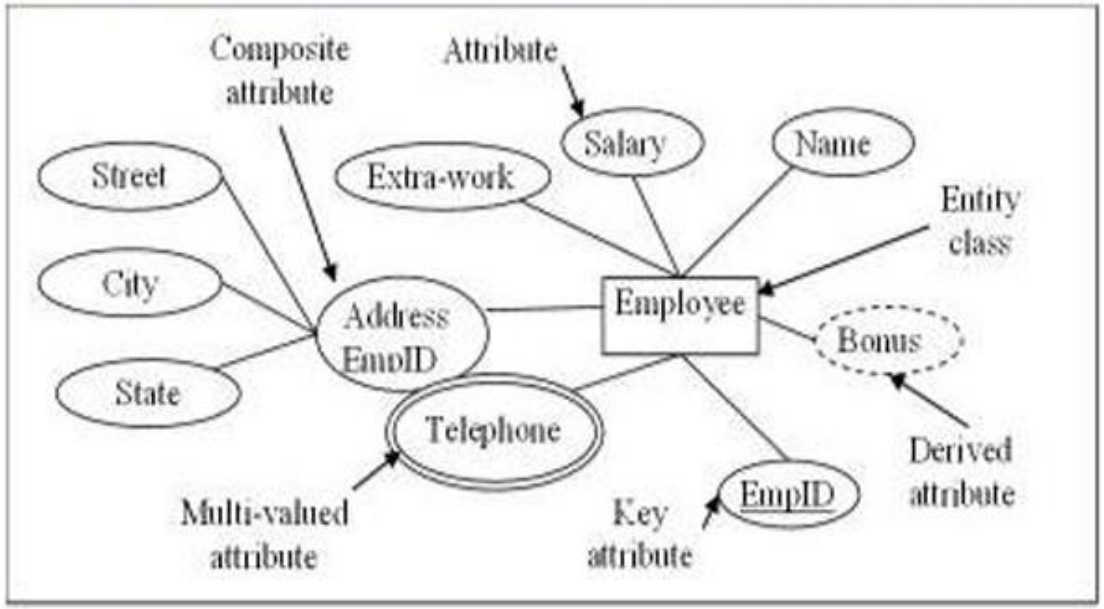
2-level DFD delves deeper into aspects of 1-level DFD. It can be used to design or record specific/necessary details about how the system works. Here's an example of a 2-level DFD.



2. Data-Oriented Analysis

- Also referred to as data-oriented modeling, which aims at conceptual representation of the business requirements.
- Data-oriented analysis is **performed using** entity relationship modeling (ERM).
  - Entity relationship modeling (ERM) is a **pictorial method of data representation**.
  - ERM is **represented by the E-R diagram** that **represents data** and **organizes them** in such a graphical manner that helps to design the final database.
  - **An entity** may represent a group of people, places, things, events, or concepts.
  - **A relationship** represents the association between two or more entities.

Entity and Attributes Examples



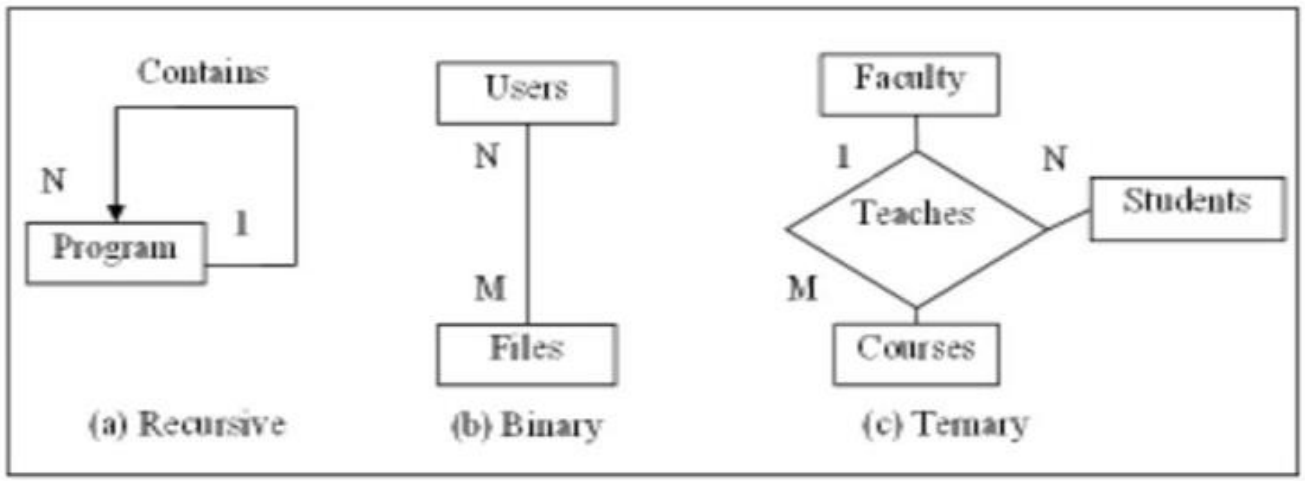
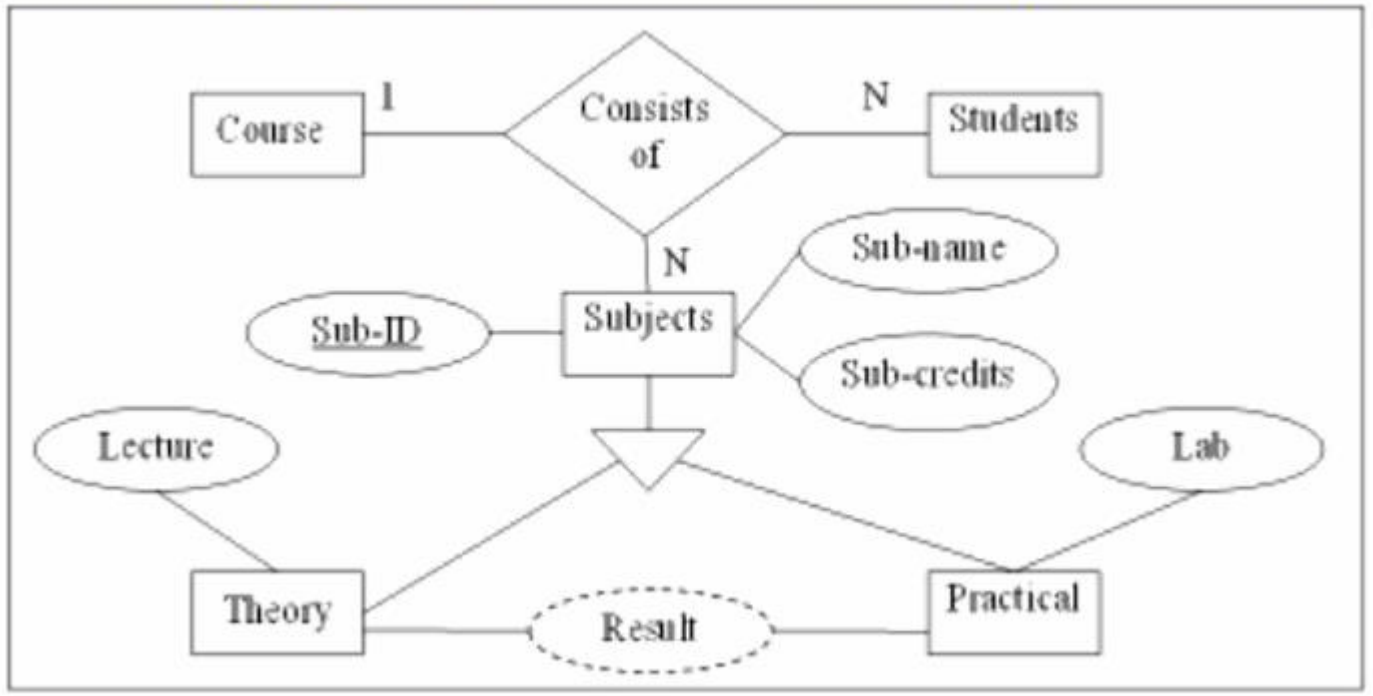


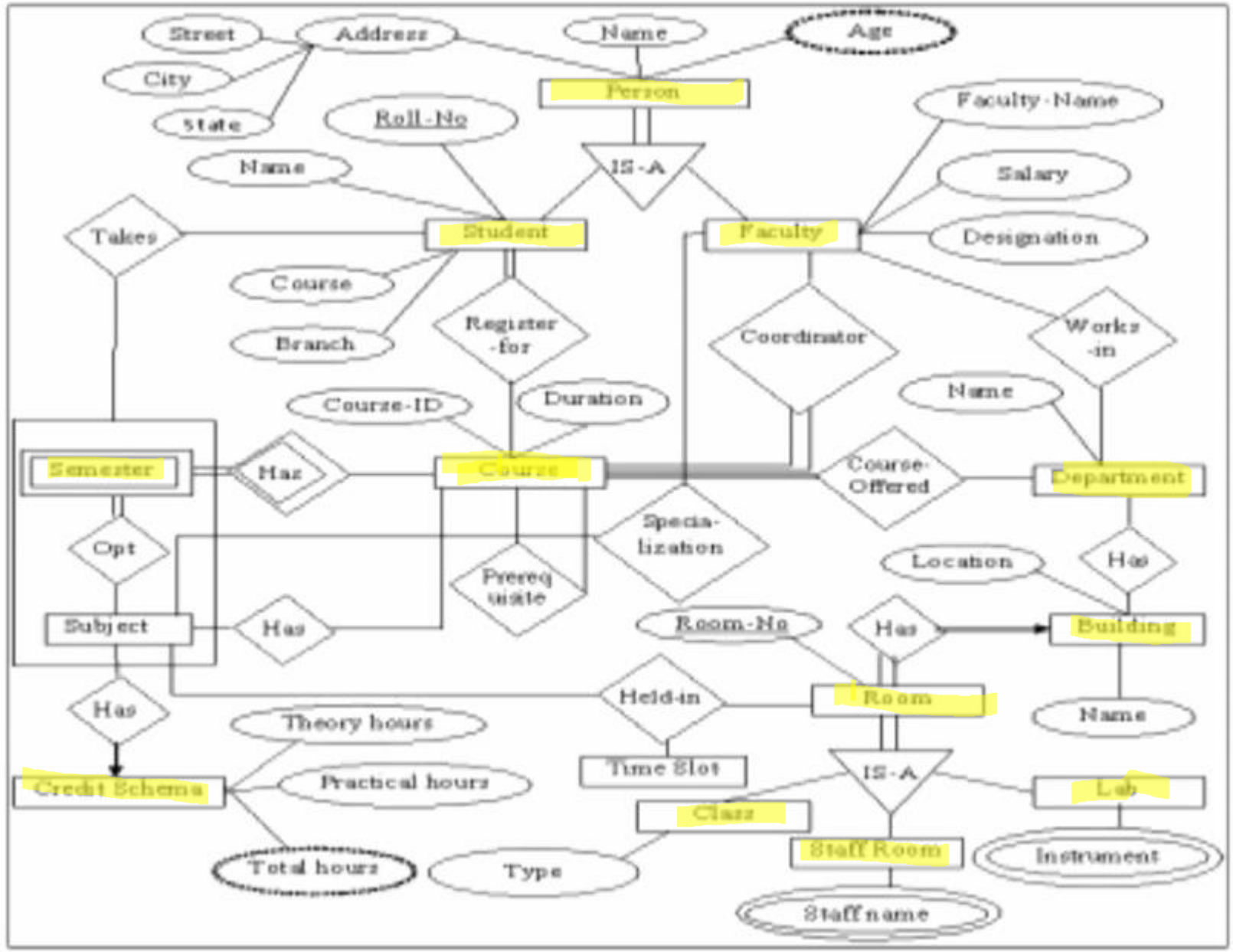
Fig: Relationships with Cardinalities

ERM: Aggregation, Generalization, and Specialization



Data-Oriented analysis Method

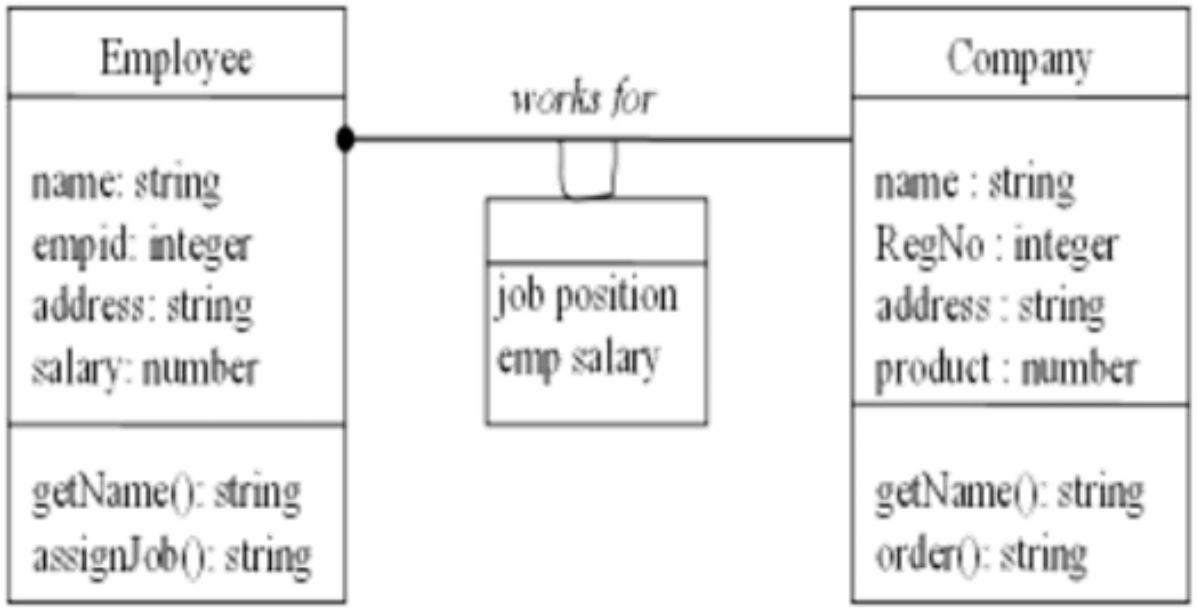
1. Identification of entity and relationships
2. Construct the basic E-R diagram
3. Add key attributes to the basic E-R model
4. Add non-key attributes to the basic E-R model
5. Apply hierarchical relation
6. Perform normalization
7. Adding integrity rules to the model


Street
Address
Name
Age
City
Person
State
Roll-No
Faculty-Name
Name
IS-A
Salary
Takes
Student
Faculty
Designation
Course
Register-for
Coordinator
Works-in
Branch
Name
Course-ID
Duration
Semester
Has
Course
Course-Offered
Department
Opt
Specia-lization
Location
Has
Prereq uisite
Subject
Has
Room-No
Has
Building
Held-in
Room
Has
Theory hours
Name
Time Slot
IS-A
Credit Schema
Practical hours
Lab
Class
Staff Room
Total hours
Type
Instrument
Staffname

3. Object-Oriented Analysis

- Object-oriented approach also **combines both data and processes** into single entities called objects.
- The object-oriented approach **has two aspects**, object-oriented analysis (OOA) and object-oriented design (OOD).

Fig: Class Diagram and Association of an Employee in a Company



Object modeling Method:

1. *Identifying object and classes*
2. *Prepare a data dictionary*
3. *Identifying associations*
4. *Identifying attributes*
5. *Refining with inheritance*
6. *Grouping classes into modules*

Data dictionary for the EtransQ system

User_credential = Username + Password

Offer_detail = Offer_id + Offer_owner + Cost + Offer_date + Branch_id + Summary

Tender_detail = Tender_id + Start_date + End_date + Cost + City

User_detail = Name + Password + Organization_name + Address + [Office_number + Mobile_number + Fax_number]*

Search_detail = [Offer_detail | Tender_details | User_detail]*

Registration_detail = Name + Password + Organization_name + Address + [Office_number + Mobile_number + Fax_number]*

Search_result = [Offer_detail | Tender_detail | User_detail]*

Registered_user = User_id + Password

User = [Supplier | Transporter | Agent | Visitor | Administrator | Client]

Admin_credential = Admin _name + Password Logistic_detail = [News |

Event]*

Testimonial_detail = Testimonial_id + Testimonial_for + Testimonial_by + Publish_status + Request_date

Data dictionary is metadata that describe composite data structures defined in the DFD. • Data dictionary is written using special symbols, such as “+” for composition, “*” for repetition, and “|” for selection. • The combinations of repeated data are written using “* +” symbol.

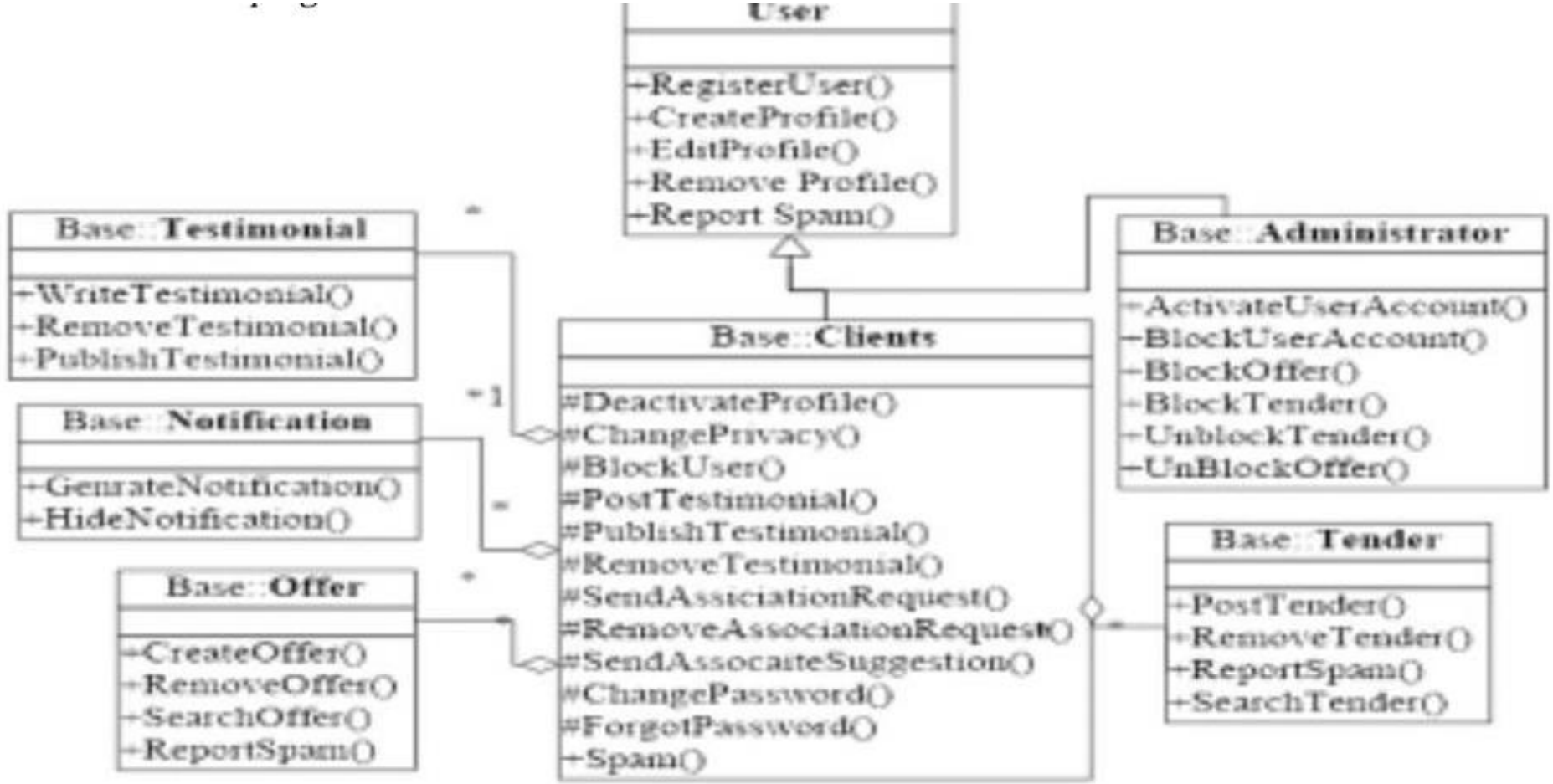


Fig: Class Diagram for ETransQ System

4. Prototyping

- There are two types of prototyping approaches widely used for elicitation, analysis, and requirement validation:
  - ➢ Throwaway prototyping
  - ➢ Evolutionary prototyping

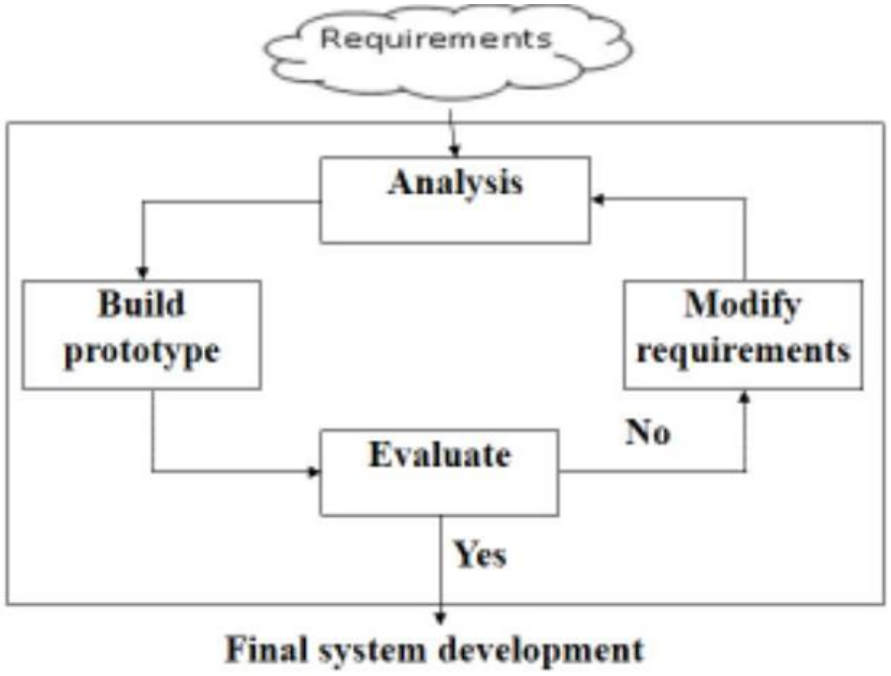
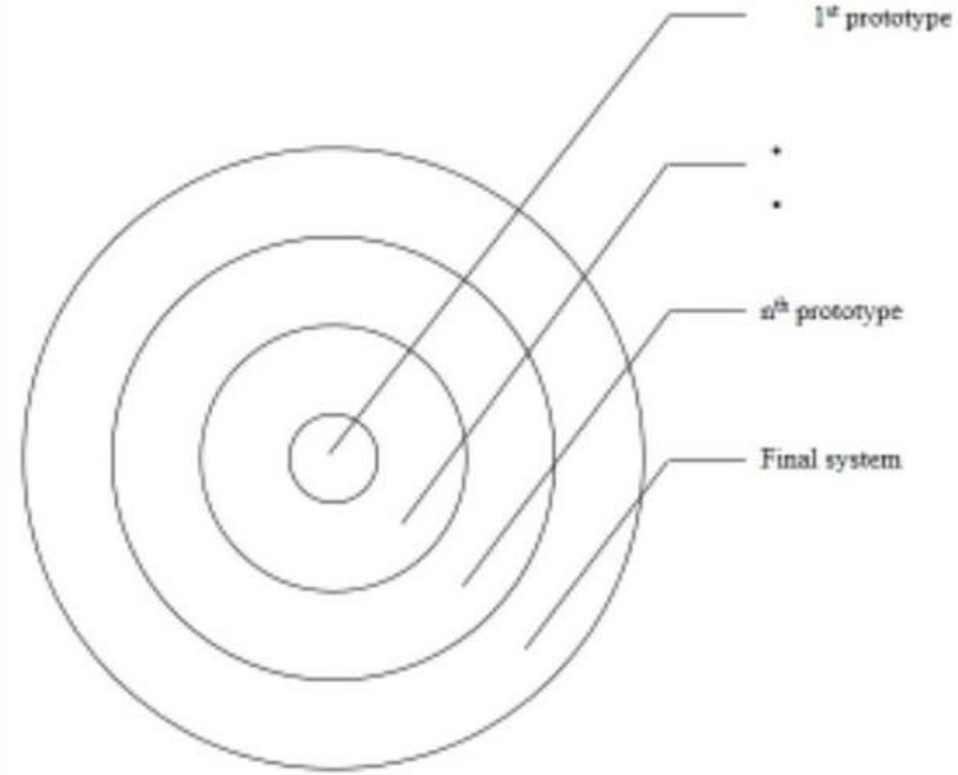


Fig: Throwaway protoyping

Fig: Evolutionary Prototyping



3. Requirements Specification

- The main focus of the problem analysis approaches is to understand the **internal behavior of the software**
- Software requirements specification is one of the **important documents required** in the software development.
- Requirements are described in a formal document called software requirement specification **(SRS).**
- **SRS is a description of a software system to be developed**.
- It **lays out functional and non-functional requirements** of the software to be developed.
- It may include a set of **use cases** that describe user interactions that the software must provide to the user for perfect interaction.

SRS Structure

1. Introduction

- 1.1 Purpose
- 1.2 Intended Audience
- 1.3 Scope
- 1.4 Definitions
- 1.5 References

2. Overall Description

- 2.1 User Interfaces
- 2.2 System Interfaces
- 2.3 Software & Hardware Requirements
- 2.4 Constraints, assumptions and dependencies
- 2.5 User Characteristics

3. System Features and Requirements

- 3.1 Functional Requirements
- 3.2 Use Cases
- 3.3 External Interface Requirements
- 3.4 Logical database requirement
- 3.5 Nonfunctional Requirements

4. Deliver for Approval

IEEE Structure of SRS

1. Introduction
1.1 Purpose
1.2 Scope
1.3 Definitions, acronyms, and & abbreviations
1.4 References
1.5 Document overview
2. General description
2.1 Product perspective
2.2 Product functions
2.3 User characteristics
2.4 General constraints
2.5 Assumptions and& dependencies
3. Specific requirements
3.1 Functional requirements
3.1.1 Functional requirement 1
3.1.1.1 Introduction
3.1.1.2 Input
3.1.1.3 Processing
3.1.1.4 Output
3.1.N Functional requirement M
3.2 External interface requirements
3.3 Performance requirements
3.4 Design constraints
3.5 Security requirements
3.6 Maintainability requirements
3.7 Reliability requirements
3.8 Availability requirements
3.9 Database requirements
3.10 Documentation requirements
3.11 Safety requirements
3.12 Operational requirements
3.13 Site adaptation

4. Requirements Validation

- After SRS developed, the **requirements discussed in SRS document are validated or tested.**
- Requirement validation **done through Requirement reviews taken by customers, After developing prototyping** check with customers & **Test case generations**.

Requirements can be the check against the following conditions -

- If they can practically implement.
- If they are correct and as per the functionality and specially of software.
- If there are any ambiguities.
- If they can describe
- Any changing or additional requirement

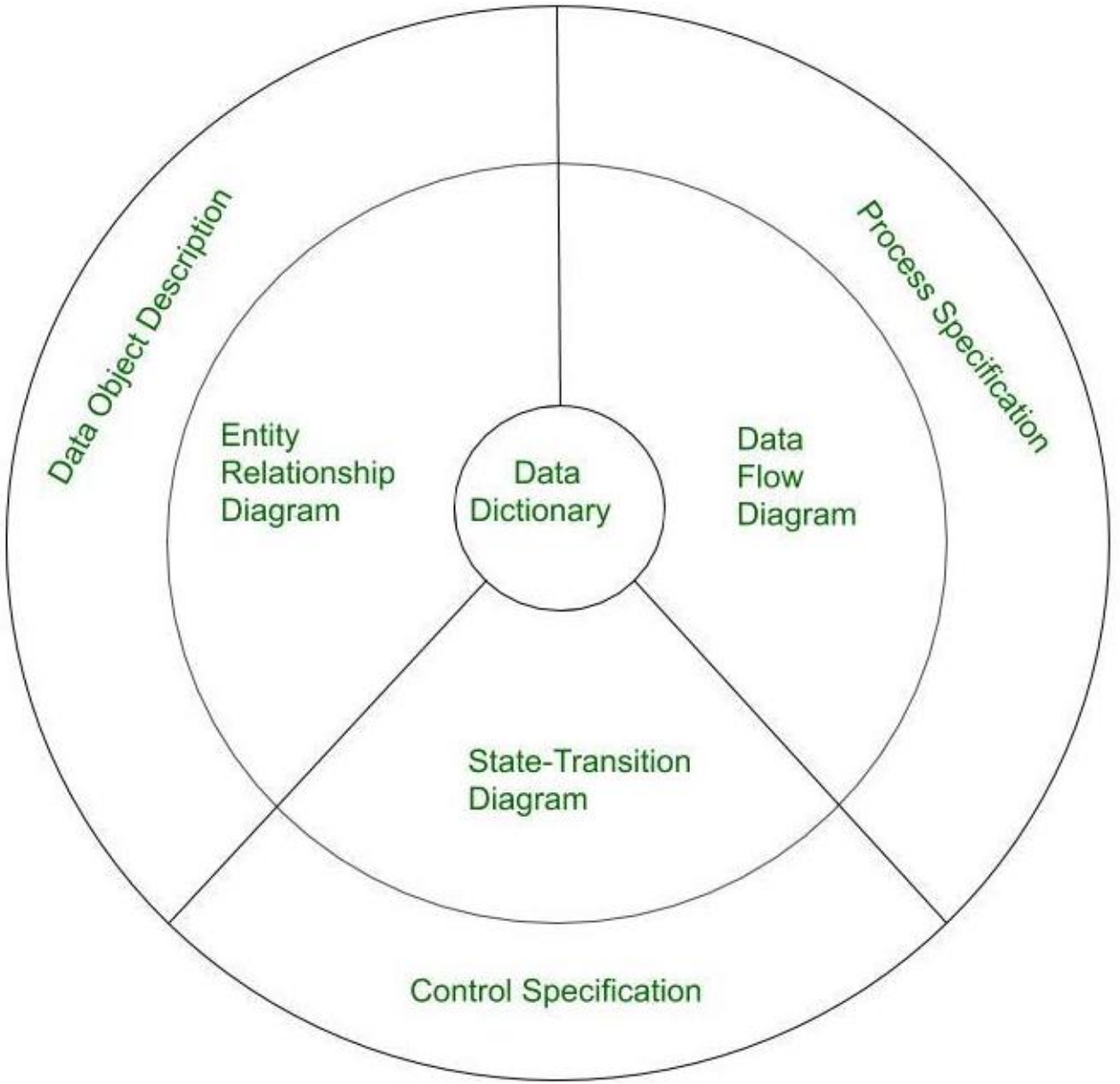
5. Requirement Management

- Requirement management is the **process of managing changing requirements** during the requirements engineering process.
- Activities involved in Requirement Management:
  - **Tracking & Controlling Changes:** Handling changing requirements from customers.
  - **Version Control:** Keeping track of different versions of system.
  - **Traceability:** Trace to software is design, develop or test as per requirements or not.
  - **Communication:** If changing requirements has any issues contact with clients within time.
  - **Monitoring & Reporting:** Monitoring development process & report the status.

Analysis model....

- Analysis Model is a technical representation of the system.
- It acts as a link between the system description and the design model.
- In Analysis Modelling, information, behavior, and functions of the system are defined and translated into the architecture, component, and interface level design in the design modeling.

Elements of Analysis Model



1. **Data Dictionary:**
   It is a repository that consists of a description of all data objects used or produced by the software. It stores the collection of data present in the software. It is a very crucial element of the analysis model. It acts as a centralized repository and also helps in modeling data objects defined during software requirements.

2. **Entity Relationship Diagram (ERD):**
   It depicts the relationship between data objects and is used in conducting data modeling activities. The attributes of each object in the Entity-Relationship Diagram can be described using Data object description. It provides the basis for activity related to data design.

3. **Data Flow Diagram (DFD):**
   It depicts the functions that transform data flow, and it also shows how data is transformed when moving from input to output. It provides the additional information that is used during the analysis of the information domain and serves as a basis for the modeling of function. It also enables the engineer to develop models of functional and information domains at the same time.

4. **State Transition Diagram:**
   It shows various modes of behavior (states) of the system and also shows the transitions from one state to another state in the system. It also provides the details of how the system behaves due to the consequences of external events. It represents the behavior of a system by presenting its states and the events that cause the system to change state. It also describes what actions are taken due to the occurrence of a particular event.

5. **Process Specification:**
   It stores the description of each function present in the data flow diagram. It describes the input to a function, the algorithm that is applied for the transformation of input, and the output that is produced. It also shows regulations and barriers imposed on the performance characteristics that apply to the process and layout constraints that could influence how the process will be implemented.

6. **Control Specification:**
   It stores additional information about the control aspects of the software. It is used to indicate how the software behaves when an event occurs and which processes are invoked due to the occurrence of the event. It also provides the details of the processes which are executed to manage events.

7. **Data Object Description:**
   It stores and provides complete knowledge about a data object present and used in the software. It also gives us the details of attributes of the data object present in the Entity Relationship Diagram. Hence, it incorporates all the data objects and their attributes.