1.1 Overview of System Analysis and Design

System development is systematic process which includes phases such as planning, analysis, design, deployment and maintenance. Here, in this, we will primarily focus on:

- System Analysis
- System Design

System Analysis: -

It is a process of collecting and interpreting facts, identifying the problems and decomposition of a system into its components. System analysis is conducted for the purpose of studying a system or its parts in order to identify its objectives. It is a problem-solving technique that improves the system and ensures that all the components of the system work efficient to accomplish their purpose.

System Design: -

It is a process of planning a new business system or replacing an existing system by defining its components or modules to satisfy the specific requirements. Before planning you need to understand the old system thoroughly and determine how we can operate the system efficiently.

1.2 Information Systems and Its Types

” An information system is a set of interrelated components that works together to collect, process, store and breakdown the information to support decision making. ”

Types Of Information System and Its Types: -

- Transaction Processing System
- Management Information System
- Decision Management System
- Experts System

1. Transaction Processing System (TPS):

   Transaction Processing System are information system that processes data resulting from the occurrences of business transactionsTheir objectives are to provide transaction in order to update records and generate reports i.e to perform store keeping functionThe transaction is performed in two ways: Batching processing and Online transaction processing.

   - Example: Bill system, payroll system, Stock control system.

2. Management Information System (MIS):

   Management Information System is designed to take relatively raw data available through a Transaction Processing System and convert them into a summarized and aggregated form for the manager, usually in a report format. It reports tending to be used by middle management and operational supervisors.Many different types of report are produced in MIS. Some of the reports are a summary report, on-demand report, ad-hoc reports and an exception report.

   - Example: Sales management systems, Human resource management system.

3. Decision Support System (DSS):

   Decision Support System is an interactive information system that provides information, models and data manipulation tools to help in making the decision in a semi-structured and unstructured situation.Decision Support System comprises tools and techniques to help in gathering relevant information and analyze the options and alternatives, the end user is more involved in creating DSS than an MIS.

   - Example: Financial planning systems, Bank loan management systems.

4. Experts System:

   Experts systems include expertise in order to aid managers in diagnosing problems or in problem-solving. These systems are based on the principles of artificial intelligence research.Experts Systems is a knowledge-based information system. It uses its knowledge about a specify are to act as an expert consultant to users.Knowledgebase and software modules are the components of an expert system. These modules perform inference on the knowledge and offer answers to a user's question.

1.3 Stakeholders of Information Systems

A stakeholder is any person who has an interest in an existing or proposed information system. Stakeholders can be technical or nontechnical workers. They may also include both internal and external workers. Information workers are those workers whose jobs involve the creation, collection, processing, distribution, and use of information. Knowledge workers are a subset of information workers whose responsibilities are based on a specialized body of knowledge.

- System Owner
- System User
- Project Manager
- System Analyst
- System Designer
- System Builders
- External Service Provider (ESP)

Systems analyst: -

A specialist who studies the problems and needs of an organization to determine how people, data, processes, and information technology can best accomplish improvements for the business. A programmer/analyst (or analyst/programmer) includes the responsibilities of both the computer programmer and the systems analyst. A business analyst focuses on only the non-technical aspects of systems analysis and design.

- By "Problems" that need solving, we mean:
    - **Problems** either real or anticipated, that require corrective action.
    - **Opportunities** to improve a situation despite the absence of complaints.
    - **Directives** to change a situation regardless of whether anyone has complained about the current situation.

Skills needed by System Analyst: -

- Working knowledge of information technology
- Computer programming experience and expertise
- General business knowledge
- General problem-solving skills
- Good interpersonal communication skills
- Good interpersonal relations skills
- Flexibility and adaptability
- Character and ethics

1.4 Systems Development Life Cycle and Life Cycle Models

Definition: -

The Software Development Life Cycle (SDLC) refers to a methodology with clearly defined processes for creating high-quality software. in detail, the SDLC methodology focuses on the following phases of software development:

- Requirement analysis
- Planning
- Software design such as architectural design
- Software development
- Testing
- Deployment

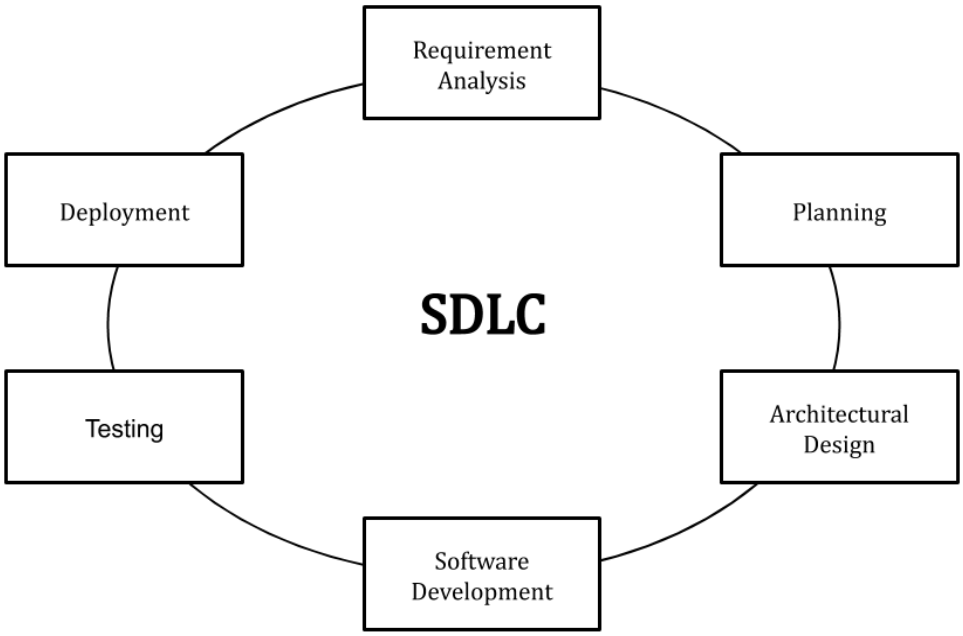


Fig. System Development Life Cycle

Our Simplified System Development Process	**General Problem-Solving Steps**
System initiation	Identify the problem.
System analysis	Analyze and understand the problem. Identify solution requirements or expectations.
System design	Identify alternative solutions and choose the “best” course of action. Design the chosen solution.
System construction	Construct the designed system
System implementation	Implement the chosen solution. Evaluate the results. If the problem is not solved, return to step 1 or 2 as appropriate.

SDLC MODELS: -

- Waterfall Models
- Spiral Models
- Prototype Models

Waterfall Model: -

Winston Royce introduced the Waterfall Model in 1970.This model has five phases: Requirements analysis and specification, design, implementation, and unit testing, integration and system testing, and operation and maintenance. The steps always follow in this order and do not overlap. The developer must complete every phase before the next phase begins. This model is named "**Waterfall Model**", because its diagrammatic representation resembles a cascade of waterfalls.

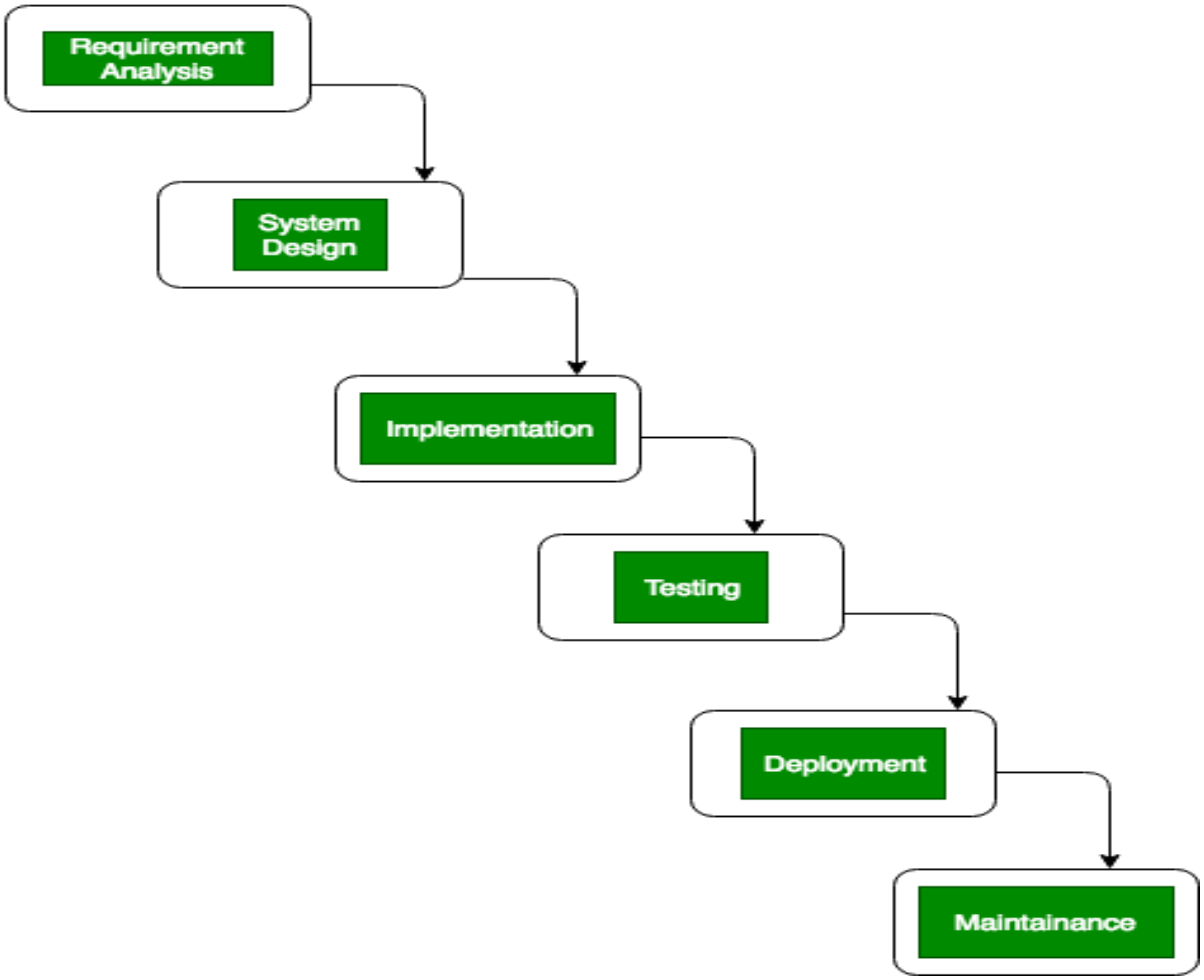


Fig. Waterfall Model

1. Requirements analysis and specification phase:

The aim of this phase is to understand the exact requirements of the customer and to document them properly. Both the customer and the software developer work together so as to document all the functions, performance, and interfacing requirement of the software. It describes the "what" of the system to be produced and not "how."In this phase, a large document called **Software Requirement Specification (SRS)** document is created which contained a detailed description of what the system will do in the common language.

2. System Design: This phase aims to transform the requirements gathered in the SRS into a suitable form which permits further coding in a programming language. It defines the overall software architecture together with high level and detailed design. All this work is documented as a Software Design Document (SDD).

3. Implementation: During this phase, design is implemented. If the SDD is complete, the implementation or coding phase proceeds smoothly, because all the information needed by software developers is contained in the SDD. During testing, the code is thoroughly examined and modified. Small modules are tested in isolation initially. After that these modules are tested by writing some overhead code to check the interaction between these modules and the flow of intermediate output.

4. System Testing and Deployment: This phase is highly crucial as the quality of the end product is determined by the effectiveness of the testing carried out. The better output will lead to satisfied customers, lower maintenance costs, and accurate results. Unit testing determines the efficiency of individual modules. However, in this phase, the modules are tested for their interactions with each other and with the system.

5. Maintenance: Maintenance is the task performed by every user once the software has been delivered to the customer, installed, and operational.

Advantages	Disadvantages
▪ Easy to explain to the users. ▪ Structures approach. ▪ Stages and activities are well defined. ▪ Helps to plan and schedule the project. ▪ Verification at each stage ensures early detection of errors/misunderstandings. ▪ Each phase has specific deliverables.	▪ Assumes that the requirements of a system can be frozen. ▪ Very difficult to go back to any stage after it is finished. ▪ A little flexibility and adjusting the scope are difficult and expensive. ▪ Costly and required more time, in addition to the detailed plan.

Spiral Model: -

The spiral model, initially proposed by Boehm, is an evolutionary software process model that couples the iterative feature of prototyping with the controlled and systematic aspects of the linear sequential model. It implements the potential for rapid development of new versions of the software. Using the spiral model, the software is developed in a series of incremental releases. During the early iterations, the additional release may be a paper model or prototype. During later iterations, more and more complete versions of the engineered system are produced.

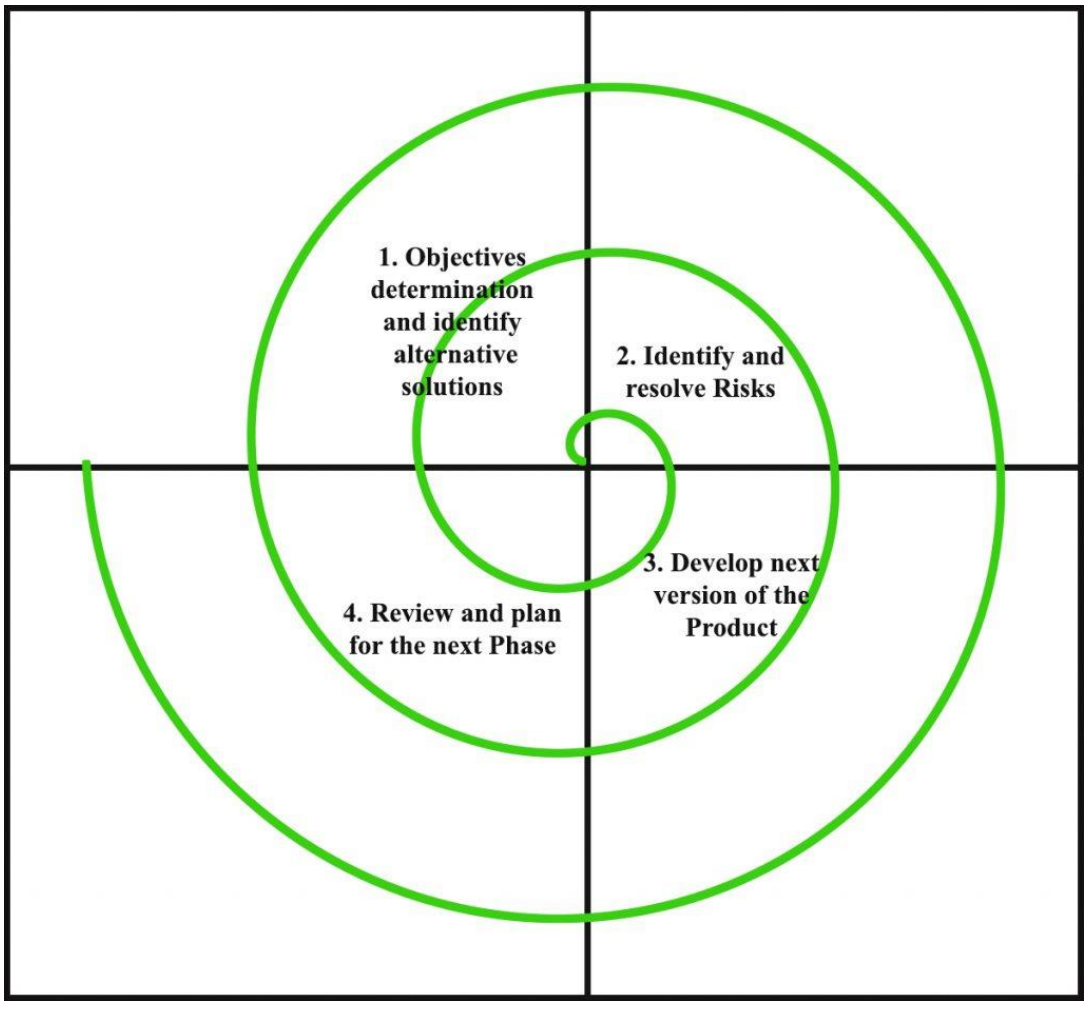


Fig. Spiral Model

Objective setting: Each cycle in the spiral starts with the identification of purpose for that cycle, the various alternatives that are possible for achieving the targets, and the constraints that exists. **Risk Assessment and reduction:** The next phase in the cycle is to calculate these various alternatives based on the goals and constraints. The focus of evaluation in this stage is located on the risk perception for the project.

Development and validation: The next phase is to develop strategies that resolve uncertainties and risks. This process may include activities such as benchmarking, simulation, and prototyping.

Planning: Finally, the next step is planned. The project is reviewed, and a choice made whether to continue with a further period of the spiral. If it is determined to keep, plans are drawn up for the next step of the project.

The development phase depends on the remaining risks. For example, if performance or user-interface risks are treated more essential than the program development risks, the next phase may be an evolutionary development that includes developing a more detailed prototype for solving the risks.

The **risk-driven** feature of the spiral model allows it to accommodate any mixture of a specification-oriented, prototype-oriented, simulation-oriented, or another type of approach. An essential element of the model is that each period of the spiral is completed by a review that includes all the products developed during that cycle, including plans for the next cycle. The spiral model works for development as well as enhancement projects.

Advantages	Disadvantages
▪ Estimates (i.e. budget, schedule, etc.) become more realistic as work progressed because important issues are discovered earlier. ▪ Early involvement of developers. ▪ Manages risks and develops the system into phases.	▪ High cost and time to reach the final product. ▪ Needs special skills to evaluate the risks and assumptions. ▪ Highly customized limiting re-usability

Prototype Model: -

The prototype model requires that before carrying out the development of actual software, a working prototype of the system should be built. A prototype is a toy implementation of the system. A prototype usually turns out to be a very crude version of the actual system, possible exhibiting limited functional capabilities, low reliability, and inefficient performance as compared to actual software. In many instances, the client only has a general view of what is expected from the software product. In such a scenario where there is an absence of detailed information regarding the input to the system, the processing needs, and the output requirement, the prototyping model may be employed.

Steps of Prototype Model

1. Requirement Gathering and Analyst
2. Quick Decision
3. Build a Prototype
4. Assessment or User Evaluation
5. Prototype Refinement
6. Engineer Produc

Advantages	**Disadvantages**
▪ Reduced time and costs, but this can be a disadvantage if the developer loses time in developing the prototypes. ▪ Improved and increased user involvement.	▪ Insufficient analysis. User confusion of prototype and finished system. ▪ Developer misunderstanding of user objectives. ▪ Excessive development time of the prototype. ▪ It is costly to implement the prototypes

1.5 Introduction to Analysis and Design Tools

Software analysis and design includes all activities, which help the transformation of requirement specification into implementation. Requirement specifications specify all functional and non-functional expectations from the software. These requirement specifications come in the shape of human readable and understandable documents, to which a computer has nothing to do.

Software analysis and design is the intermediate stage, which helps human-readable requirements to be transformed into actual code.

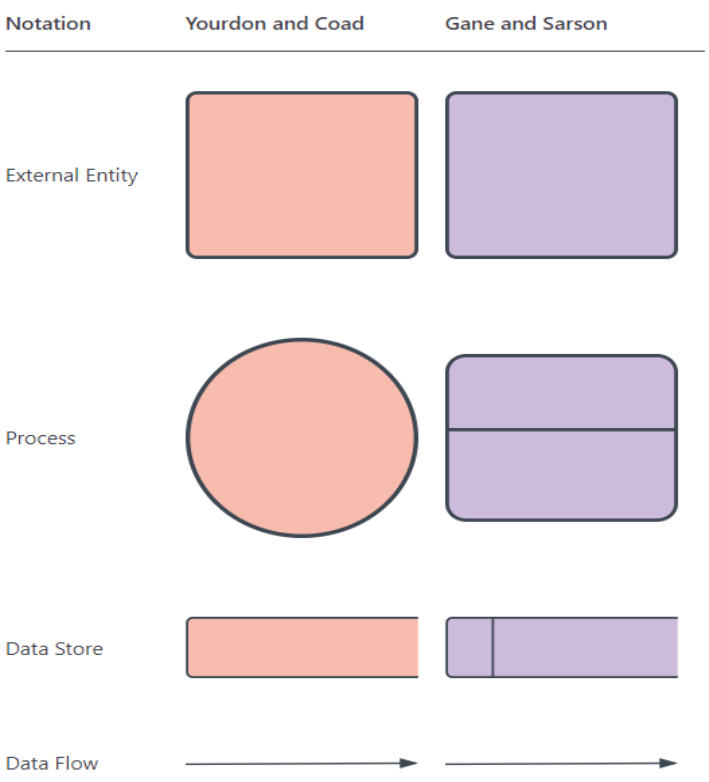
Let us see few analysis and design tools used by software designers:

- Data Flow Diagram
- Structure Chart
- Hippo Diagram
- Structure English
- Pseudo Code
- E-R diagram
- Decision Table

Unit 2: Process and Conceptual Modeling

Conceptual models are abstract, psychological representations of how tasks should be carried out. People use conceptual models subconsciously and intuitively as a way of systematizing processes.

2.1 Introduction to Data Flow Diagram (DFD)



A data flow diagram (DFD) maps out the flow of information for any process or system. It uses defined symbols like rectangles, circles and arrows, plus short text labels, to show data inputs, outputs, storage points and the routes between each destination. Data flowcharts can range from simple, even hand-drawn process overviews, to in-depth, multi-level DFDs that dig progressively deeper into how the data is handled. They can be used to analyze an existing system or model a new one. Like all the best diagrams and charts, a DFD can often visually "say" things that would be hard to explain in words, and they work for both technical and nontechnical audiences, from developer to CEO. That's why DFDs remain so popular after all these years. While they work well for data flow software and systems, they are less applicable nowadays to visualizing interactive, real-time or database-oriented software or systems.

2.2 Concept used in drawings (DFD)

Using any convention's DFD rules or guidelines, the symbols depict the four components of data flow diagrams.

1. **External entity:** An outside system that sends or receives data, communicating with the system being diagrammed. They are the sources and destinations of information entering or leaving the system. They might be an outside organization or person, a computer system or a business system. They are also known as terminators, sources and sinks or actors. They are typically drawn on the edges of the diagram.

2. **Process:** Any process that changes the data, producing an output. It might perform computations, or sort data based on logic, or direct the data flow based on business rules. A short label is used to describe the process, such as "Submit payment."

3. **Data store:** Files or repositories that hold information for later use, such as a database table or a membership form. Each data store receives a simple label, such as "Orders."

4. **Data flow:** The route that data takes between the external entities, processes and data stores. It portrays the interface between the other components and is shown with arrows, typically labeled with a short data name, like "Billing details."

Rule of Data Flow

One of the rules for developing DFD is that all flow must begin with and end at a processing step. This is quite logical because data can't transform on its own with being process. By using the thumb rule, it is quite easy to identify the illegal data flows and correct them in a DFD.

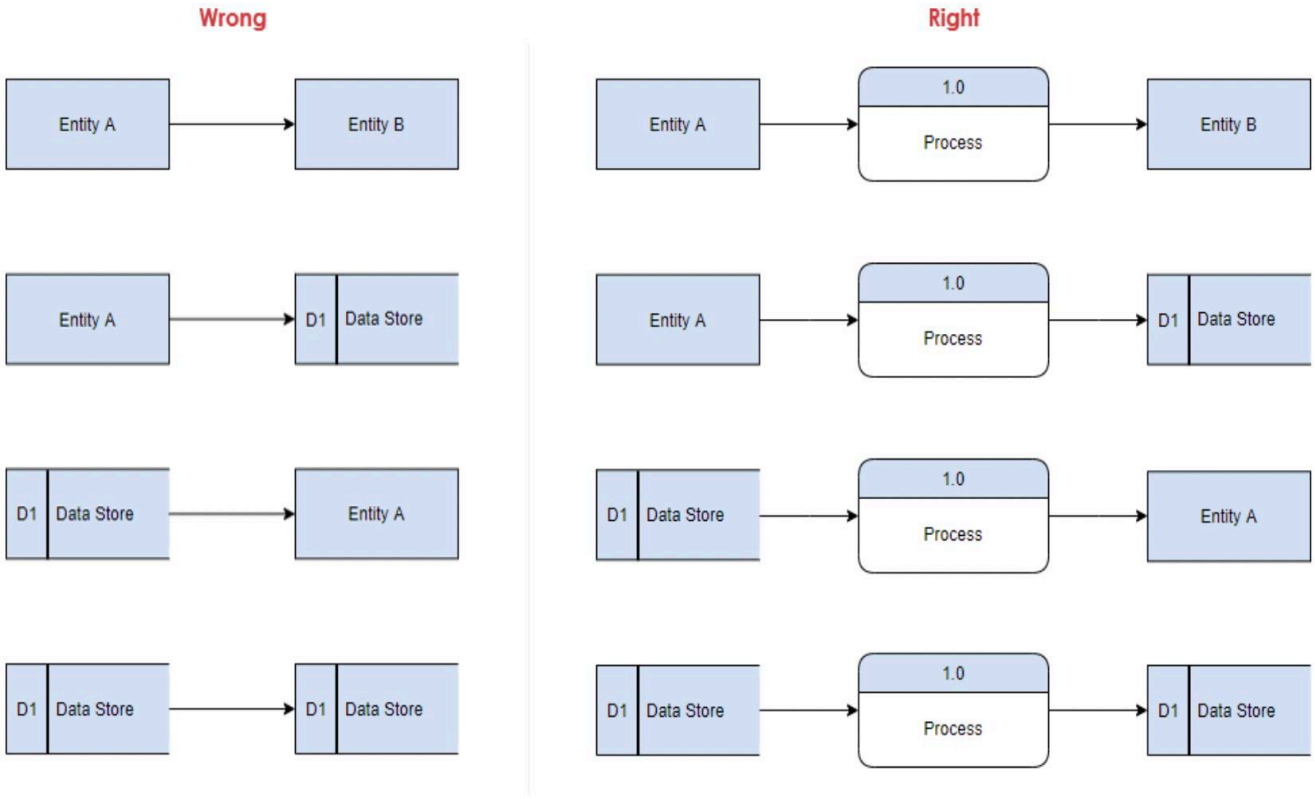
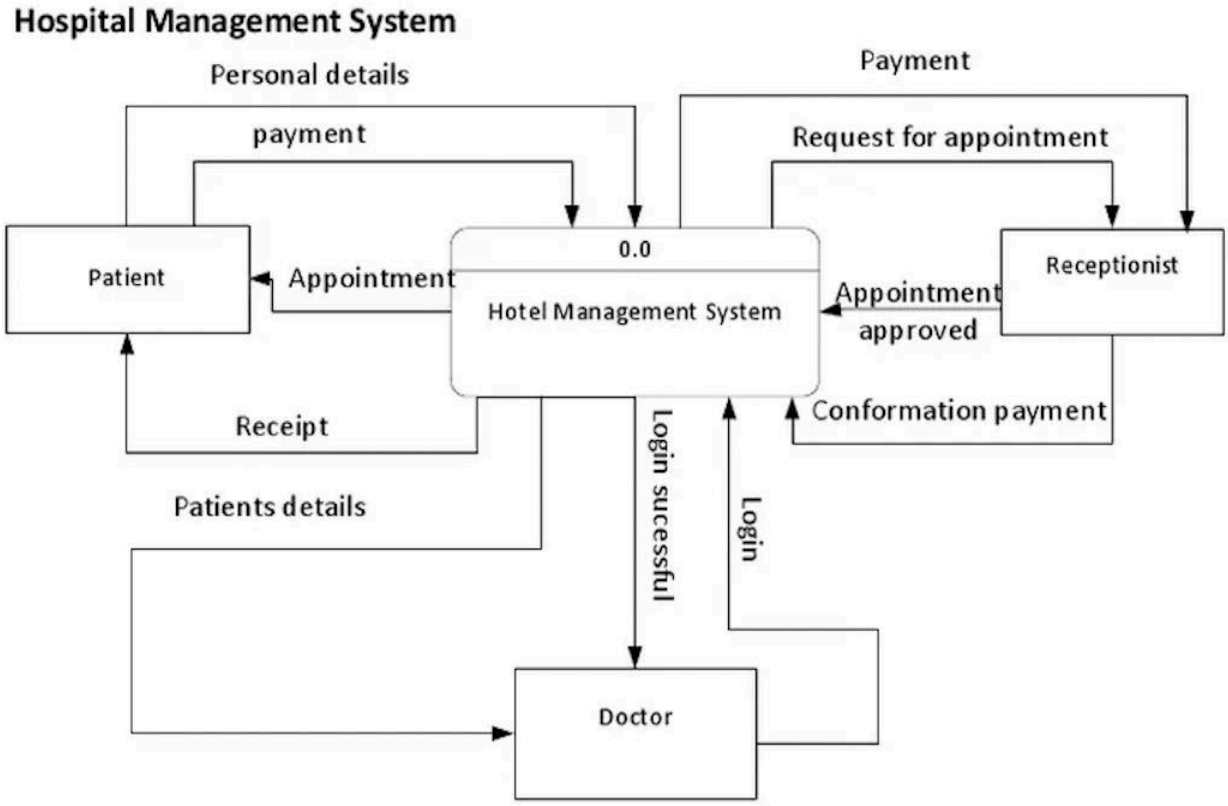


Fig. Rules of DFD

2.3 DFD design (up to level 2)

Levels or layers are used in DFDs to represent progressive degrees of detail about the system or process. These levels include:

- **Level 0:** Also known as a "context diagram," this is the highest level and represents a very simple, top-level view of the system being represented.
- **Level 1:** Still a relatively broad view of the system, but incorporates subprocesses and more detail.
- **Level 2:** Provides even more detail and continues to break down subprocesses as needed.

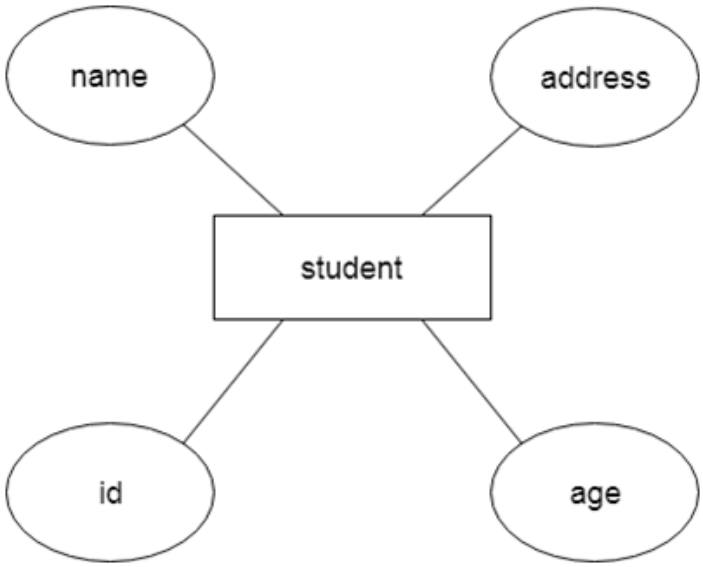

Hospital Management System
Personal details
payment
Payment
Request for appointment
Patient
Appointment
0.0
Hotel Management System
Appointment
approved
Receptionist
Receipt
Conformation payment
Patients details
Login sucessful
Login
Doctor

DFD Level 0

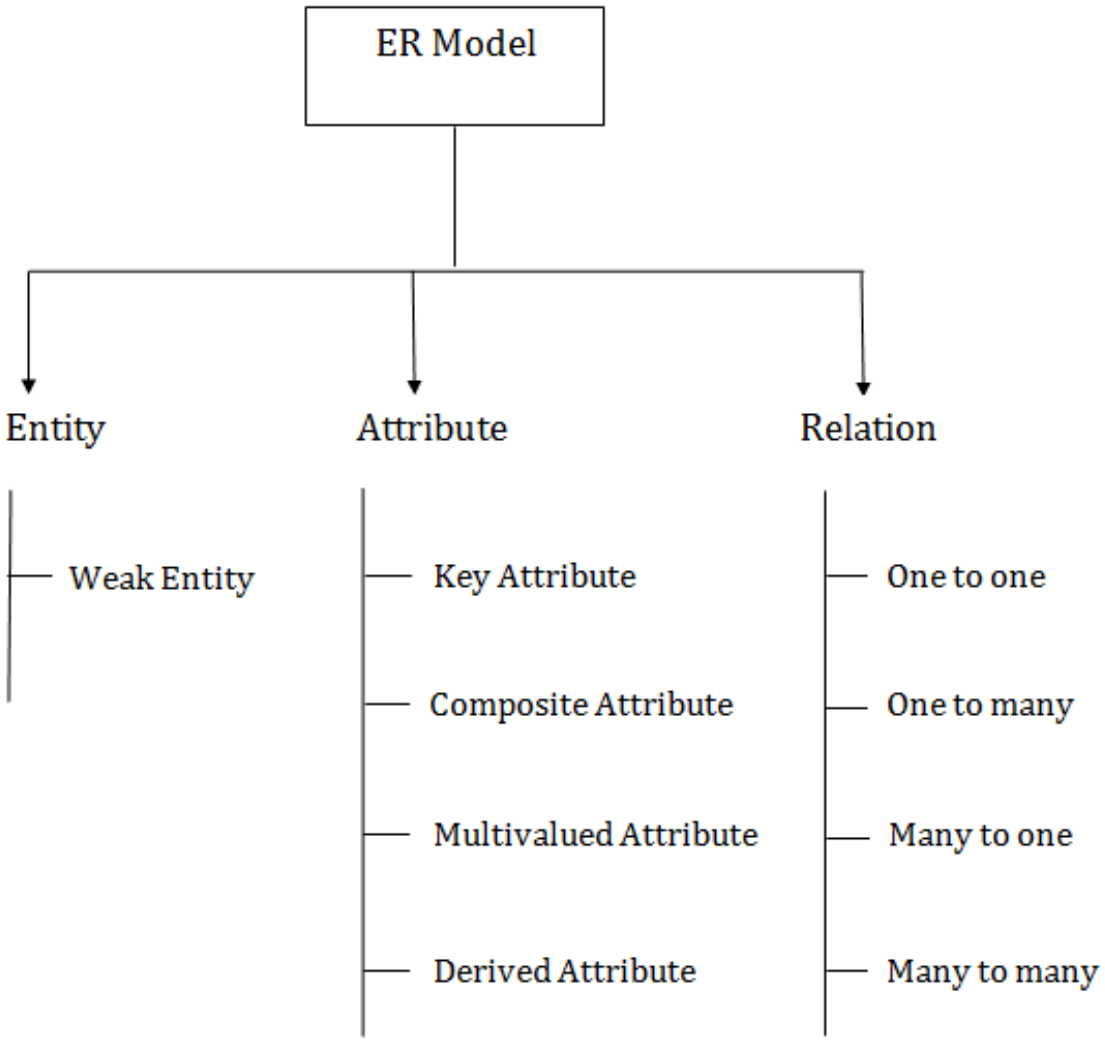
2.5 Entity Relationship Diagram

- ER model stands for an Entity-Relationship model. It is a high-level data model. This model is used to define the data elements and relationship for a specified system.
- It develops a conceptual design for the database. It also develops a very simple and easy to design view of data.
- In ER modeling, the database structure is portrayed as a diagram called an entity-relationship diagram.

For example, Suppose we design a school database. In this database, the student will be an entity with attributes like address, name, id, age, etc. The address can be another entity with attributes like city, street name, pin code, etc and there will be a relationship between them.



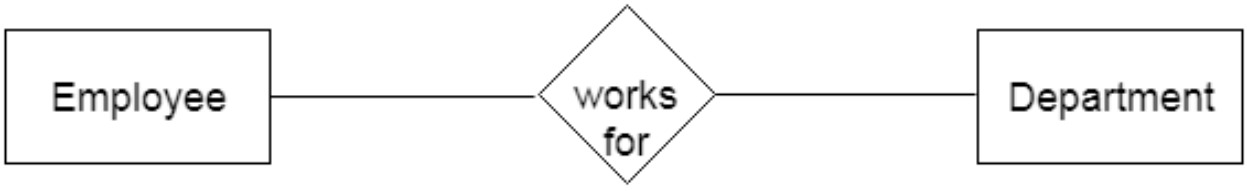
Component of ER Diagram



1. Entity:

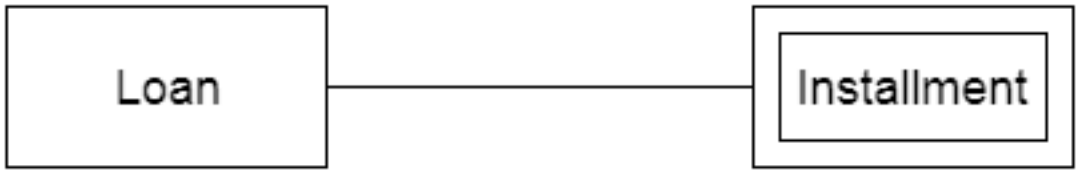
An entity may be any object, class, person or place. In the ER diagram, an entity can be represented as rectangles.

Consider an organization as an example- manager, product, employee, department etc. can be taken as an entity.



a. Weak Entity

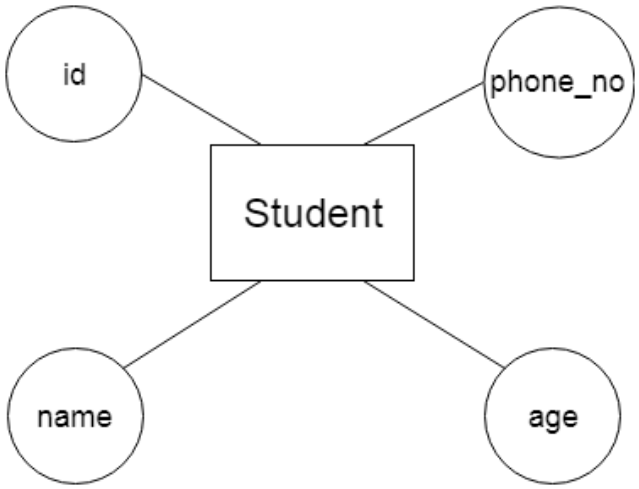
An entity that depends on another entity called a weak entity. The weak entity doesn't contain any key attribute of its own. The weak entity is represented by a double rectangle.



2. Attribute

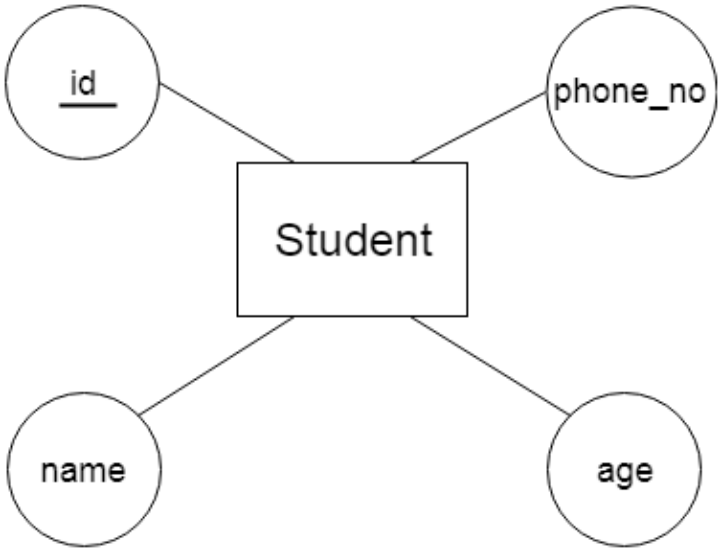
The attribute is used to describe the property of an entity. Eclipse is used to represent an attribute.

For example, id, age, contact number, name, etc. can be attributes of a student.



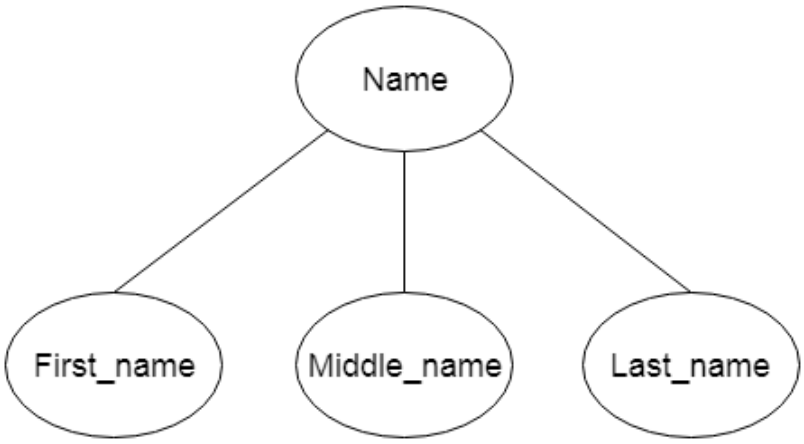
a. Key Attribute

The key attribute is used to represent the main characteristics of an entity. It represents a primary key. The key attribute is represented by an ellipse with the text underlined.



b. Composite Attribute

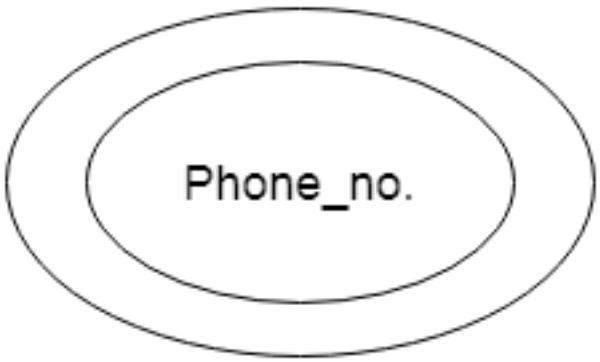
An attribute that composed of many other attributes is known as a composite attribute. The composite attribute is represented by an ellipse, and those ellipses are connected with an ellipse.



c. Multivalued Attribute

An attribute can have more than one value. These attributes are known as a multivalued attribute. The double oval is used to represent multivalued attribute.

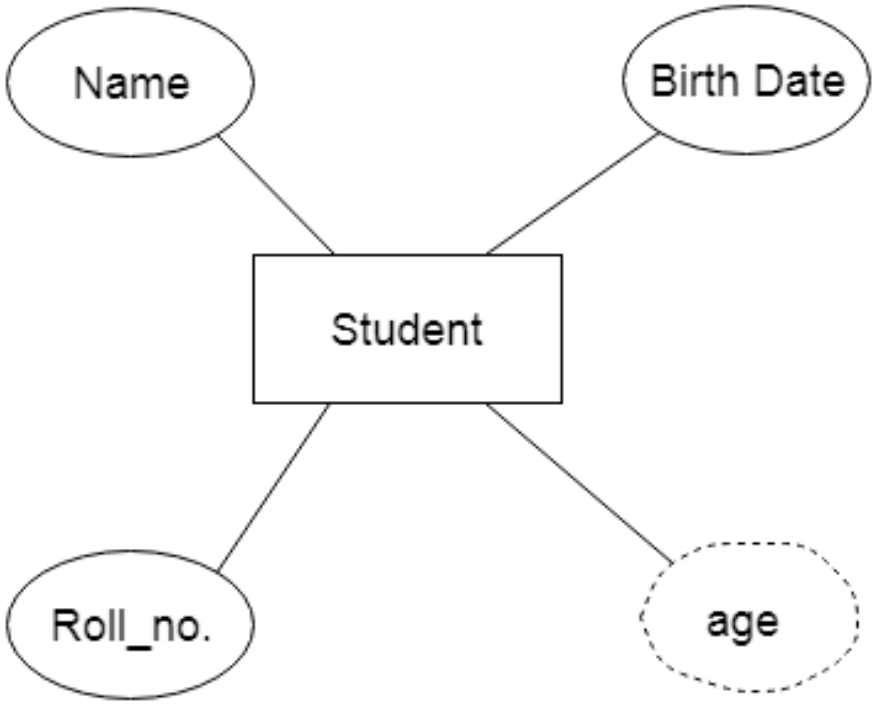
For example, a student can have more than one phone number.



d. Derived Attribute

An attribute that can be derived from other attribute is known as a derived attribute. It can be represented by a dashed ellipse.

For example, A person's age changes over time and can be derived from another attribute like Date of birth.



3. Relationship

A relationship is used to describe the relation between entities. Diamond or rhombus is used to represent the relationship.

Types of relationship are as follows:

a. One-to-One Relationship

When only one instance of an entity is associated with the relationship, then it is known as one to one relationship.

For example, A female can marry to one male, and a male can marry to one female.

b. One-to-many relationship

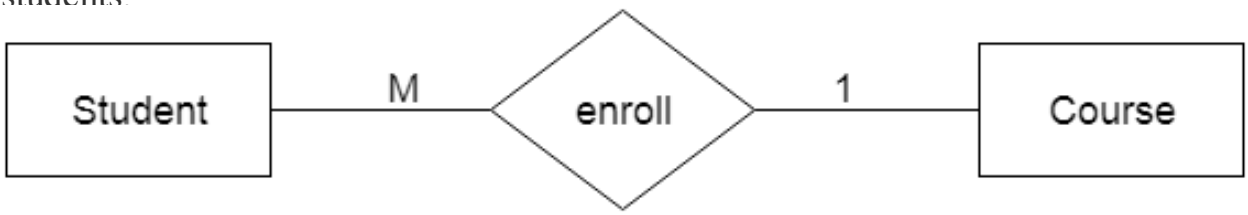
When only one instance of the entity on the left, and more than one instance of an entity on the right associates with the relationship then this is known as a one-to-many relationship.

For example, Scientist can invent many inventions, but the invention is done by the only specific scientist.

c. Many-to-one relationship

When more than one instance of the entity on the left, and only one instance of an entity on the right associates with the relationship then it is known as a many-to-one relationship.

For example, Student enrolls for only one course, but a course can have many students.



d. Many-to-many relationship

When more than one instance of the entity on the left, and more than one instance of an entity on the right associates with the relationship then it is known as a many-to-many relationship.

For example, Employee can assign by many projects and project can have many employees.

E-R Diagram example

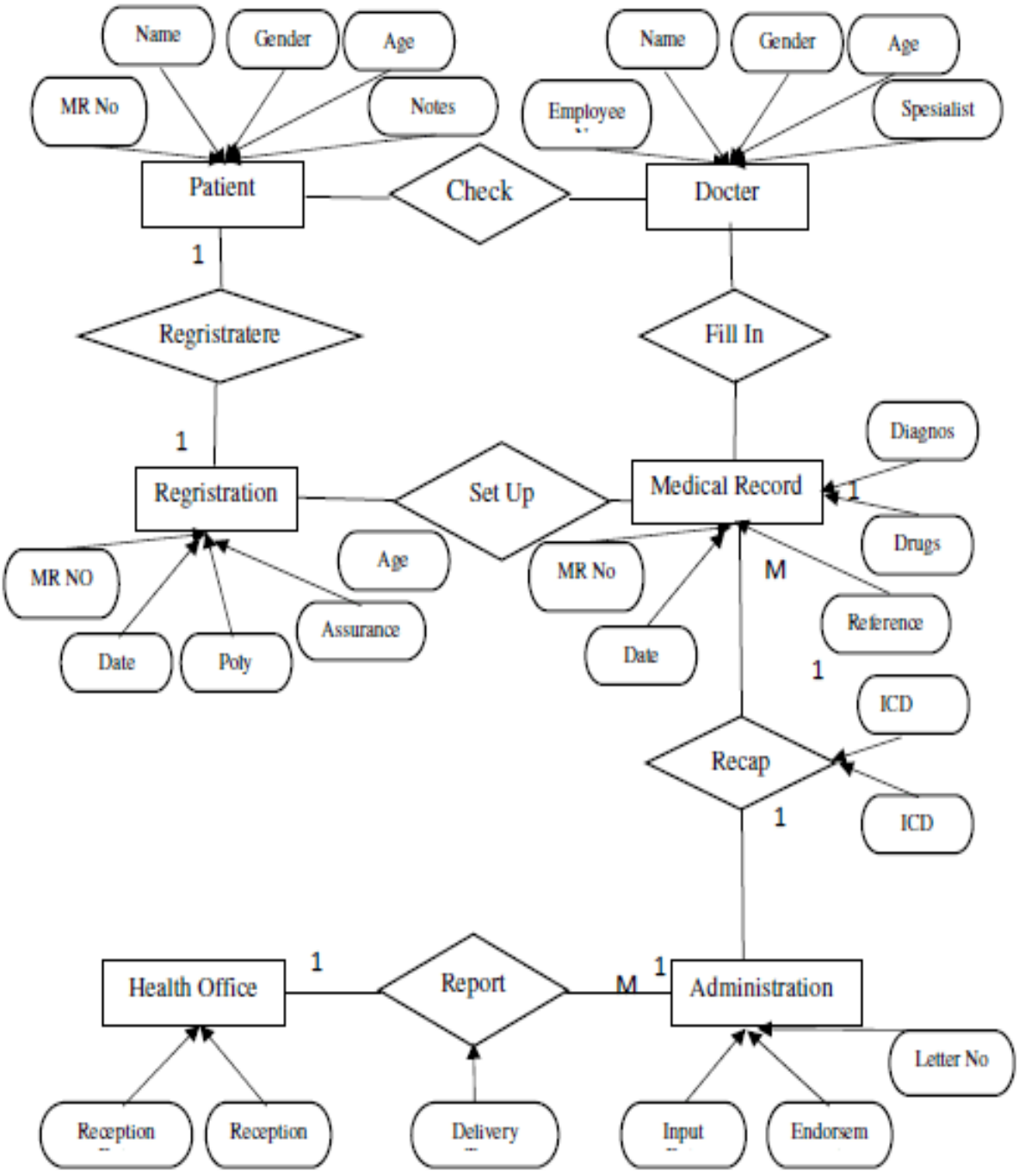


Fig. E-R Diagram

UNIT 3 Logical Modeling

3.1 Decision Table

A decision table is a scheduled rule logic entry, in table format, that consists of conditions, represented in the row and column headings, and actions, represented as the intersection points of the conditional cases in the table. Decision tables are best suited for business rules that have multiple conditions. Adding another condition is done by simply adding another row or column.

Like the if/then rule set, the decision table is driven by the interaction of conditions and actions. The main difference is that in a decision table, the action is decided by more than one condition, and more than one action can be associated with each set of conditions. If the conditions are met, then the corresponding action or actions are performed.

		Rule 1	*Rule 2*	*Rule 3*	*Rule 4*
IF	Condition 1	Y	Y	N	N
AND	Condition 2	Y	N	Y	Y
AND	Condition 3	-	N	Y	-
AND	Condition 4	-	-	Y	N
THEN	Action 1	X		X	
AND	Action 2	X			X
AND	Action 3		X		
AND	Action 4		X	X	

Fig. Decision table

3.2 Decision Tree

Decision Tree is the most powerful and popular tool for classification and prediction. A Decision tree is a flowchart-like tree structure, where each internal node denotes a test on an attribute, each branch represents an outcome of the test, and each leaf node (terminal node) holds a class label.

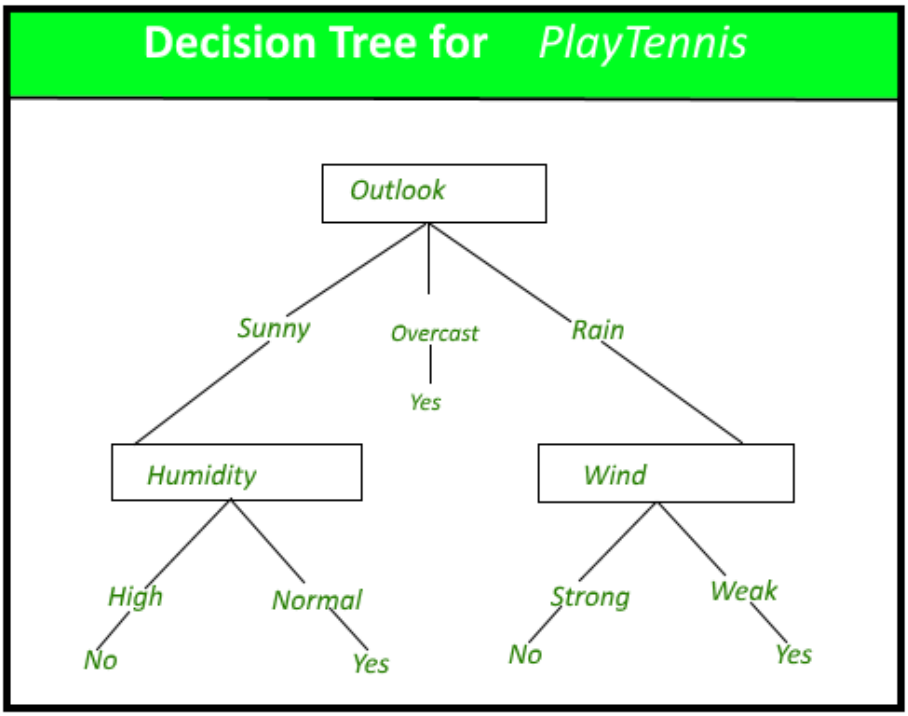


Fig.A decision tree for the concept PlayTennis.

Construction of Decision Tree: A tree can be *"learned"* by splitting the source set into subsets based on an attribute value test. This process is repeated on each derived subset in a recursive manner called *recursive partitioning*. The recursion is completed when the subset at a node all has the same value of the target variable, or when splitting no longer adds value to the predictions. The construction of a decision tree classifier does not require any domain knowledge or parameter setting, and therefore is appropriate for exploratory knowledge discovery. Decision trees can handle high-dimensional data. In general decision tree classifier has good accuracy.

3.3 Structure English

It is similar to our structured programming. It uses logical construction and imperative sentences to carry out instructions for action. Decisions are made through IF, THEN, ELSE and SO statements. The structured English for the above ex: the publishers discount policy is

We can actually make structured English more compact by using terms defined in the data dictionary. For ex: the process ORDER may have the data element ORDER-SIZE that defines four values.

MINIMUM: 5 or fewer copies per book title.
SMALL: 6 to 19 copies
MEDIUM: 20 to 49 copies
LARGE: 50 or more

Using these values, the Structured English will be

COMPUTE-DISCOUNT
IF order is from bookstore
  And-IF ORDER-SIZE is SMALL
    THEN: Discount is 25%
  ELSE (ORDER-SIZE is MINIMUM)
    SO: no discount is allowed
  ELSE (order is from libraries or individual customer)
    So-if ORDER-SIZE is LARGE
      Discount is 15%
    ELSE IF ORDER-SIZE is MEDIUM
      Discount is 10%
    ELSE IF ORDER-SIZE is SMALL
      Discount is 5%
    ELSE (ORDER-SIZE is MINIMUM)
      SO: no discount is allowed

3.4 Data Dictionary

A data dictionary is a file or a set of files that includes a database's metadata. The data dictionary hold records about other objects in the database, such as data ownership, data relationships to other objects, and other data. The data dictionary is an essential component of any relational database. Ironically, because of its importance, it is invisible to most database users. Typically, only database administrators interact with the data dictionary.

The data dictionary, in general, includes information about the following:

- Name of the data item
- Aliases
- Description/purpose
- Related data items
- Range of values
- Data structure definition/Forms

The **name of the data item** is self-explanatory.

Aliases include other names by which this data item is called DEO for Data Entry Operator and DR for Deputy Registrar.

Description/purpose is a textual description of what the data item is used for or why it exists.

Related data items capture relationships between data items e.g., total_marks must always equal to internal_marks plus external_marks.

Range of values records all possible values, e.g. total marks must be positive and between 0 to 100.

Data structure Forms: Data flows capture the name of processes that generate or receive the data items. If the data item is primitive, then data structure form captures the physical structures of the data item. If the data is itself a data aggregate, then data structure form capture the composition of the data items in terms of other data items.

The mathematical operators used within the data dictionary are defined in the table:

Notations	Meaning
x=a+b	x includes of data elements a and b.
x=[a/b]	x includes of either data elements a or b.
x=a x	includes of optimal data elements a.
x=y[a]	x includes of y or more occurrences of data element a
x=[a]z	x includes of z or fewer occurrences of data element a
x=y[a]z	x includes of some occurrences of data element a which are between y and z.