SPM

Software Process Model

Software Project

A Software Project is the complete process of software development starting from requirement analysis to testing and maintenance which are carried out according to the execution methodologies, in a specified period of time to develop the desired software product.

Software Project Manager

A software project manager is a person who undertakes the responsibility of executing the software project. Software project manager is thoroughly aware of all the phases of SDLC that the software would go through.

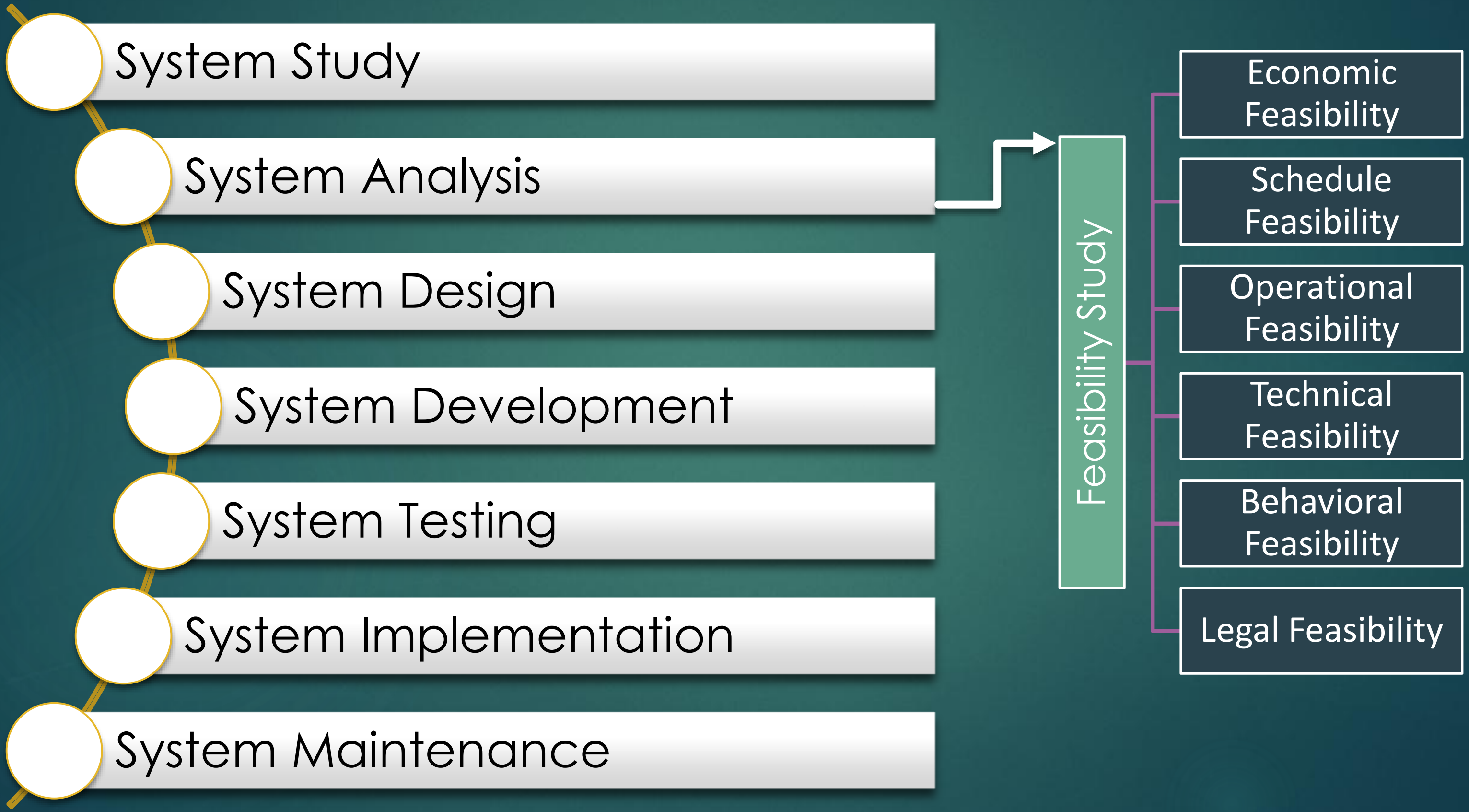
Software Development Process

Developing software is a complex and challenging task. It requires a lot of effort and dedication of the software project team in order to develop the software as requested by the client. Software development refers to the set of activities performed in the process of creating, designing, deploying and supporting software.

SDLC Life Cycle

SDLC stands for Software Development Life Cycle. Developing software is a complex and challenging task. It requires a lot of effort and dedication of the software project team in order to develop the software as requested by the client. The software development life cycle is a process of planning, creating, testing, and deploying system across hardware and software. SDLC is also called Software Development Process.

Stages of SDLC Life Cycle



System Study:

Without f finding the actual problem no one can develop the system or software. To find the problem, the programmer and user sit together and exchange their problem. This is an important and first step of software development where the developer collects the data about the need and problem of the user from the field and analysis them to get the way of solution. Every developer develops software to satisfy the requirement of the user

System Analysis:

System analysis is:

- ❑ The survey and planning of the system and project
- ❑ The study and analysis of the existing system
- ❑ The definition of requirements and priorities for a new or improved system.

Feasibility study is the most important activity in the system analysis phase. It is the study of whether the system is feasible or not to design. It consists of technical feasibility, Economical feasibility, operational feasibility, legal feasibility, behavioral feasibility, time (Schedule) feasibility.

Technical Feasibility

It concerns with the availability of the hardware, software and the support equipment for the complete development of the system.

Economical Feasibility

It concerns with cost effectiveness of the system. The main objective of economical feasibility is to calculate approximate cost-both the development cost and the operational cost and the benefits from the system. Developers always develop suitable system for an organization that should be economically feasible. So it is popularly knowns as cost benefit analysis.

Operational Feasibility

It concerns with smooth operation of the system. It is all about the problems that may occur during operation of the system after its development.

Legal Feasibility

It concerns with legal issue of the system. If the system is illegal then the system designing is meaningless. Everything is measured whether it is legal or illegal.

Behavioral Feasibility

It concerns with behavior of the users and the society towards the new system. Generally, most of the traditional employees are not easily ready to upgrade them with the new system. They may feel that their jobs are unsecured. The behavior feasibility is the key factor for the effective operation of the system.

Time (Schedule) Feasibility

Time feasibility is a determination of whether a proposed system can be implemented fully within a stipulated time frame or not. It concerns system deadlines and time frame for system completion.

System Design:

After analyzing the problem the developer prepares the general programming technique and logic. It is the stage at which programmers follows the different techniques or methods to achieve the goa. Developer develops many probable solutions to solve the problem and finally selects the best solutions. During system design different tools are used such as Algorithm, Flow Chart, DFD (Data Flow Diagram), ERD (Entity Relationship Diagram) etc.

System Development:

Once the system design phase is over, the next phase is coding. In this phase, developers start build the entire system by writing code using the chosen programming language. In the coding phase, tasks are divided into units or modules and assigned to the various developers. Inn this phase, developer needs to follow certain predefined coding guidelines.

System Testing:

It is an important state of software development. Testing tells the developer about the program. The validation of the program, testing ensures that the program performs the required tasks correctly. It is a process of running the program to find the incorrectness and logical errors containing on the program. Using the test data following testing process is carried out:

a) Unit Testing: It is the process of testing the individual component of the system after the component is developed but before the components are combined together to form a complete system.
b) Integration Testing: Integration testing is the process of testing the individual component of the system after all the individual components are combined together.
c) System Test: It is the process of testing the complete system after all the components of the system are combined together. The

System Testing:

system can be tested in two ways:

i) Black Box Testing
- Internal code of the program is tested.
- Test cases are totally hidden from the users.
- Also known as Functional Test.

ii) White Box Testing
- Structure of the program is tested.
- Test cases are totally visible to the users and they can also make test cases.
- Also known as Glass box Test.

d) User Acceptance Testing: It is the process of testing the complete system by the user to identify if the developed system meets the user's requirement or not.

System Implementation:

This phase is started after the system has been tested and accepted by the user. System performance is compared to performance objectives established during the planning phase. Implementation includes user notification, user training, installation of hardware, installation of software.

System Maintenance:

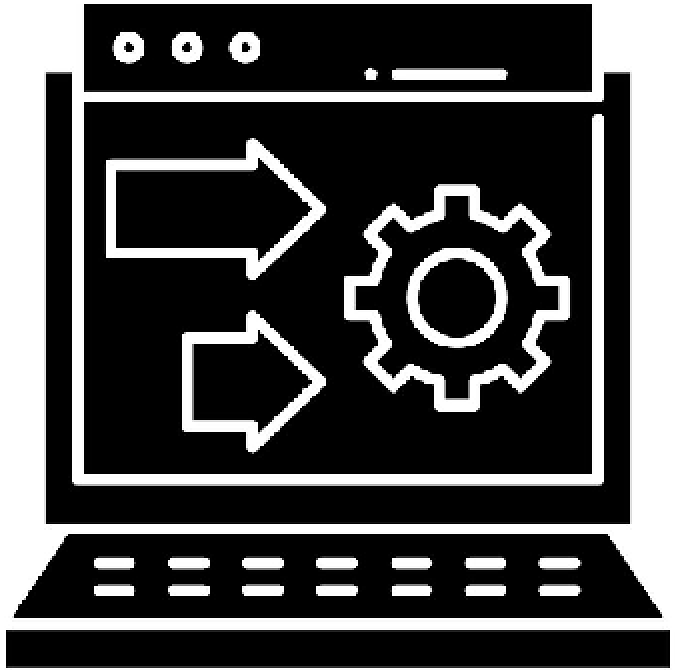
The SDLC doesn’t end when software reaches the market. Developers must now move into a maintenance mode and begin practicing any activities required to handle issues reported by end users. The maintenance phase involves making changes to hardware, software and documentation to support its operational effectiveness. This correcting and upgrading of system process is called system maintenance.

Importance and the Necessity of SDLC

The Software Development Life Cycle (SDLC) is crucial for building software applications systematically and efficiently. It outlines the stages that a project goes through from conception to completion, ensuring that each step is well-defined and followed to ensure quality, timely delivery, and alignment with customer needs.

- It helps to determine the needs of the user.
- It supports constant communication between the developer and the user.
- SDLC helps for easy identification of missing requirements.
- It ensures that the software meets the needs of its users.
- It supports proper analysis and design of the software.
- It ensures proper development and testing.
- Proper documentation support for future upgrades and maintenance.
- It provides flexibility for adding features even after the software is developed.

The necessity of SDLC lies in its ability to streamline the software development process, manage risks, control costs, and ensure the delivery of high-quality software that meets the needs of clients and users. Whether following a traditional waterfall approach or agile methodology, the SDLC provides a blueprint for success in the complex and ever-evolving world of software development.



SYSTEMS ENGINEER

System Engineer

- ❑ Software Engineer is an IT professional who have extensive knowledge of programming languages, software development and apply engineering principles in the creation of software.
- ❑ A Software Engineer is a person who analyzes and designs software based on user needs.
- ❑ Software engineer follows a systematic and disciplined approach for software design, development, deployment and maintenance of software applications.

Role and Responsibilities of System Engineer:

- ❑ Developing a software system and testing
- ❑ Analyze client’s requirements
- ❑ Overseeing the development of documentation
- ❑ Managing the software development lifecycle.
- ❑ Monitoring system performance
- ❑ Communicating to team members and building system.
- ❑ Testing new software and fixing bugs

Qualities/ Attributes of System Engineer:

- ❑ Smart programming talents.
- ❑ Smart knowledge of engineering principles
- ❑ High motivation
- ❑ Intelligence
- ❑ Ability to figure in team
- ❑ Discipline

System Analyst

System Analyst:

- ❑ A system analyst is a person who uses system analysis and design techniques to solve system problems.
- ❑ A system analyst analyzes, designs and implements system to fulfill organizational needs.
- ❑ System Analyst conducts system study, identifies requirements and determines the procedures to achieve system objectives.
- ❑ System Analyst designs and implements the system to suit organizational requirements for effective results.
- ❑ System analysts carry the responsibilities of researching problems, finding solutions, recommending courses of actions and coordinating with other members in order to meet specified requirements.
- ❑ System Analyst is responsible for the system from its start to end.

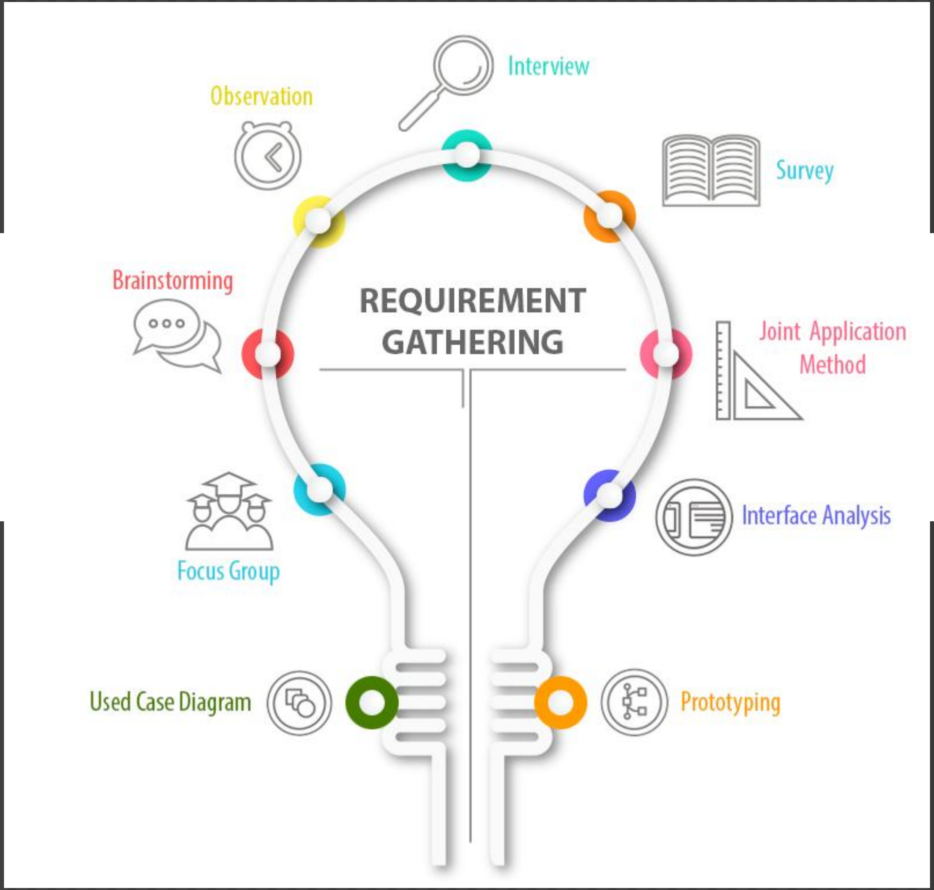
Role and Responsibilities of System Analyst:

- ❑ Research and evaluate new technologies
- ❑ Identify the organizational needs
- ❑ Analyze costs and benefits
- ❑ Add new functionality to systems
- ❑ System Analysis
- ❑ System Design
- ❑ System Upgradation
- ❑ System Implementation
- ❑ System Maintenance

Qualities/ Attributes of System Analyst

- ❑ Knowledge of the Organization
- ❑ Knowledge of Computer Hardware and Software
- ❑ Problem-solving and critical thinking
- ❑ Analytical Skills
- ❑ Motivator Skill
- ❑ Good understanding, communication and teaching abilities.

Requirement Collection Methods



Requirement Collection Methods

Interview

Interview is conducted to understand the stakeholders and curstomer's expectations from the software. Interview turns out to be one of the most effective techniques for requirement gathering. Without understanding the goals and expectations of the users and stakeholders, we are very unlikely to satisfy them.

Survey

Survey and questionnaires another effective method to collect information and requirements within a short frame of time. The survey can force users to select from choices, or have open ended questions allowing free-form responses.

Requirement Collection Methods

Joint Application Method

This method where all the stakeholders like end users, system analyst and software engineers come together and attend workshops for working on a system in greater detail.

Interface Analysis

Interface analysis helps create functional, effective and popular software for a client, group or consumer. You review the how the program or software interacts with other external systems to make sure and notice any concerns that are not widely visible to users.

Requirement Collection Methods

Prototyping

Prototyping is a newer technique used in requirement gathering. In this approach, you gather primary requirements that you use to build an initial version of the solution - a prototype. You show this to the client, who then gives you additional requirements. You change the application and cycle around with the client again. This repetitive process continues until the product meets of business needs.

Use Case Diagram

Use case analysis is a document that defines relationship between actor and system, Define how system will behave in particular situation. Use case can be used to represent system functionality.

Requirement Collection Methods

Focus Group

A focus group is a gathering of people who are representative of the users or customers of a product to get feedback. The feedback can be gathered about needs/opportunities/ problems to identify requirements, or can be gathered to validate and refine already elicited requirements.

Brainstorming

Brainstorming is used in requirement gathering to get as many ideas as possible from group of people. Generally used to identify possible solutions to problems, and clarify details of opportunities.

Requirement Collection Methods

Observation

Under the observation method, the responsible person observes the team in working environment and gets ideas about the software. Observation can be invisible, the person simply observers the working and does not interact or making himself visible, and hence, the concerned person observes and asks relevant questions.

Concept of System Design

The tools which are used to design the system is known as system design tools. Some of these tools are an Algorithm, Flowchart, Pseudo Code, Context Diagram, Data Flow Diagram, Decision Table, Decision Tree, E-R Diagram and Case Tool etc.

Concept of System Design > Algorithm

An algorithm is a procedure for solving a program. Advantages of algorithm:

- They are simple to understand.
- Errors can be pointed out very easily.
- They do not depend on any of the programming languages.
- They are compatible with computer languages.

Concept of System Design > Algorithm

Example:

Write an Algorithm to add two numbers.

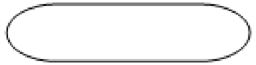
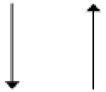
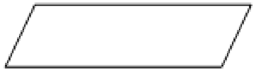
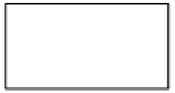
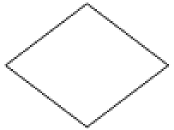
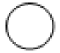
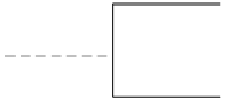
```
Step 1: START
Step 2: Input number1
Step 3: Input Numner2
Step 3: Add Number1 and Number2 and store on Sum
Step 4: Print/ Output Sum
Step 5: END
```

Concept of System Design > Flowchart

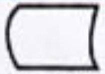
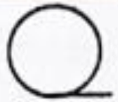
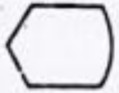
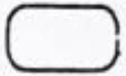
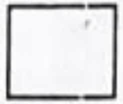
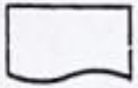
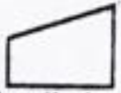
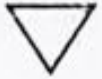
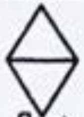
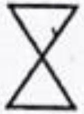
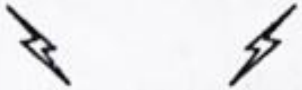
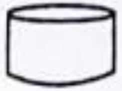
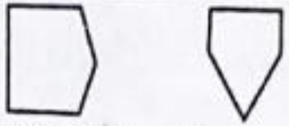
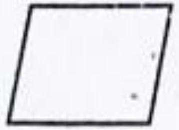
It is the pictorial representation of an algorithm. It used different predefined graphical symbols to show the data flow within a program. Advantages of Flowchart:

- ❑ It is easier for a programmer to explain the logic of a program through the flowchart.
- ❑ It helps in effective analysis of the program.
- ❑ Efficient means of communication
- ❑ With the help of flowchart, coding becomes effective and faster.

Concept of System Design > Flowchart

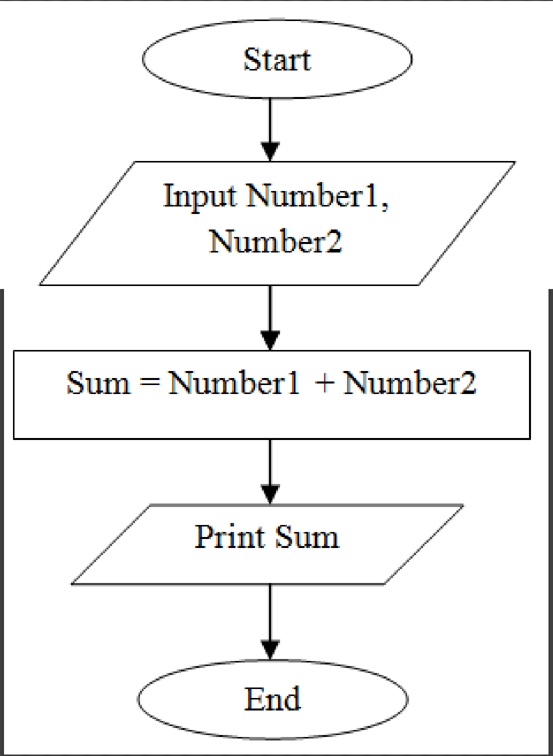
Flowchart Symbol	Symbol Name	Description
	Terminal (Start or Stop)	Terminals (Oval shapes) are used to represent start and stop of the flowchart.
	Flow Lines or Arrow	Flow lines are used to connect symbols used in flowchart and indicate direction of flow.
	Input / Output	Parallelograms are used to read input data and output or display information
	Process	Rectangles are generally used to represent process. For example, Arithmetic operations, Data movement etc.
	Decision	Diamond shapes are generally used to check any condition or take decision for which there are two answers, they are, yes (true) or no (false).
	Connector	It is used connect or join flow lines.
	Annotation	It is used to provide additional information about another flowchart symbol in the form of comments or remarks.

Symbols used in System Flow Chart

 Online Storage (Files stored on hard discs, etč)	Connector on page (a link to or from another part of diagram)	 Magnetic Tape
 Visual Display Unit (Moniter Screen)	 Terminator (Start and end of this page's diagram)	 Data processing operation
 Document (Printed hard copy)	 Maually input (especially keyboard)	 Merge (e.g. join two files of names, one file after other)
 Sort (alphabetical, chronological, numeric)	 Collate	 Communication line (e.g. telephone line, cat 5 network cable)
 Hard disk (file)	 Connector, off page (link from this diagram to another diagram)	 Input/Output operation

Concept of System Design > Flowchart

Example: Flowchart to add two numbers.



Pseudo Code

Pseudocode is an informal high-level representation of the actual code that shows how an algorithm or a computer program works in plain English. *Pseudocode does not have a specific syntax.*

Example:

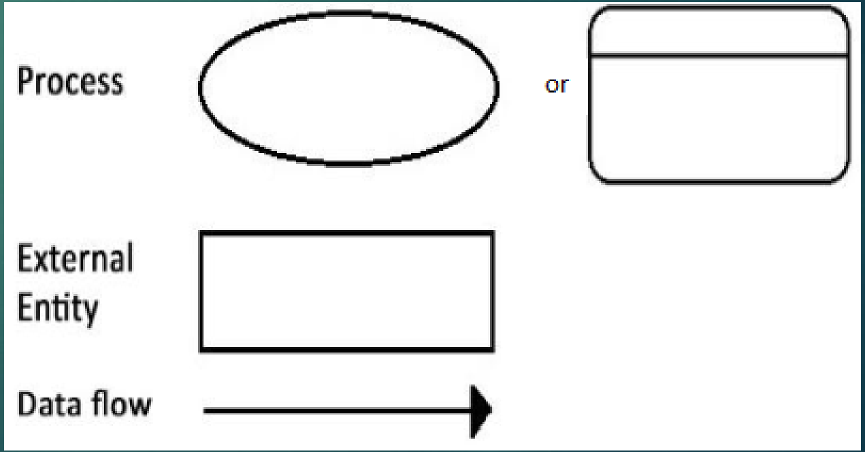
Write an Pseudo Code to add two numbers.

```
Step 1: Start
Step 2: Declare variables num1, num2 and sum
Step 3: Read values for num1, num2
Step 4: Calculate sum=num1+num2
Step 5: Display sum
Step6: Stop
```

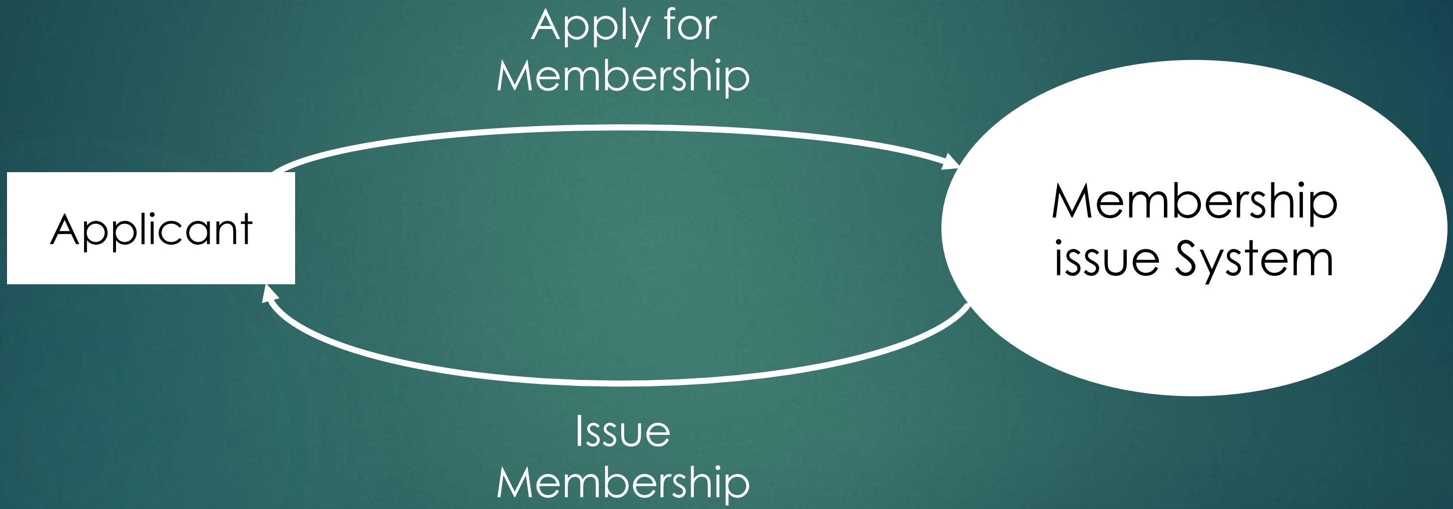
Context Diagram

Context diagrams are high-level diagrams, meaning they don't go into the detailed of the system. Instead, they map out an entire system in a way that's simple, clear, and easy to understand. For example, arrows are used to represent the flow of data between the system and each external element.

Symbol Used in Context Diagram:



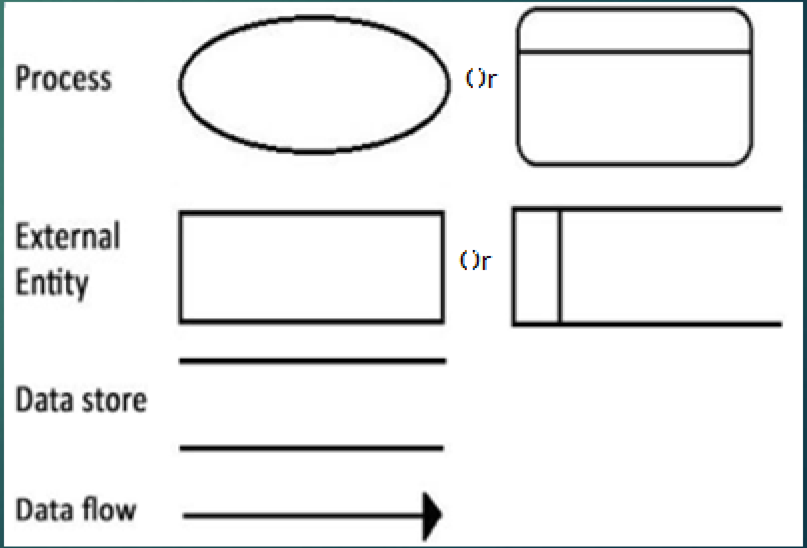
The context diagram of library membership is given below:



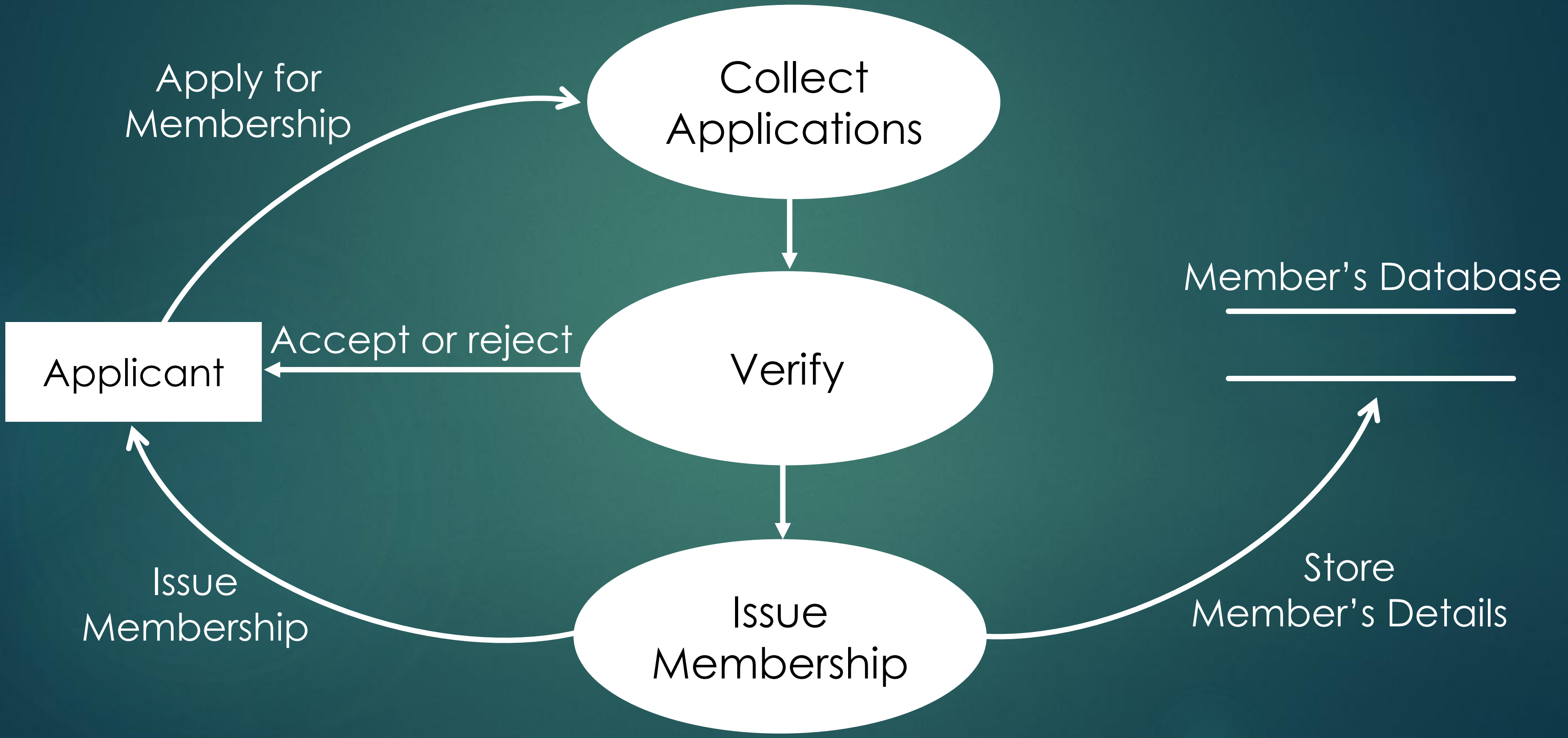
Data Flow Diagram (DFD)

Data Flow Diagram is a tool that describes the flow of data through a system and the work or processing performed by the system. The purpose of data flow diagram is to provide a bridge between users and system developers.

Symbol Used in Data Flow Diagram:



The Data Flow diagram of library membership is given below:



Decision Table

A decision table allows an analyst to identify the exact course of actions for given conditions. A decision table provides clear-cut decisions, leading to a good program design. A decision table generally consists of two parts. These ae

- ❑ Condition
- ❑ Action

Example of Decision Table

The discount policy of a departmental store is as follows:

i. If a customer is a member and purchase exceeds Rs. 1,000 then discount is 15%
ii. If a customer is a member and purchase is less than or equal to Rs. 1,000 then discount is 10%
iii. If a customer is not a member and purchase exceeds Rs. 1,000 then discount is 6%
iv. If a customer is not a member and purchase is less than or equal to Rs. 1,000 then discount is 0%

Conditions	Customer is a member	Y	Y	N	N
	Purchase > 1000	Y	N	Y	N
Actions	15% discount	X			
	10% discount		X		
	6% discount			X	
	No discount				X

Decision Tree

A decision tree is another way of presenting a potentially confusing situation in a clear manner and it can be used in decision making or analysis. It helps a system analyst to study the relationship between conditions and actions. It looks like a tree with branches.

Example of Decision Tree

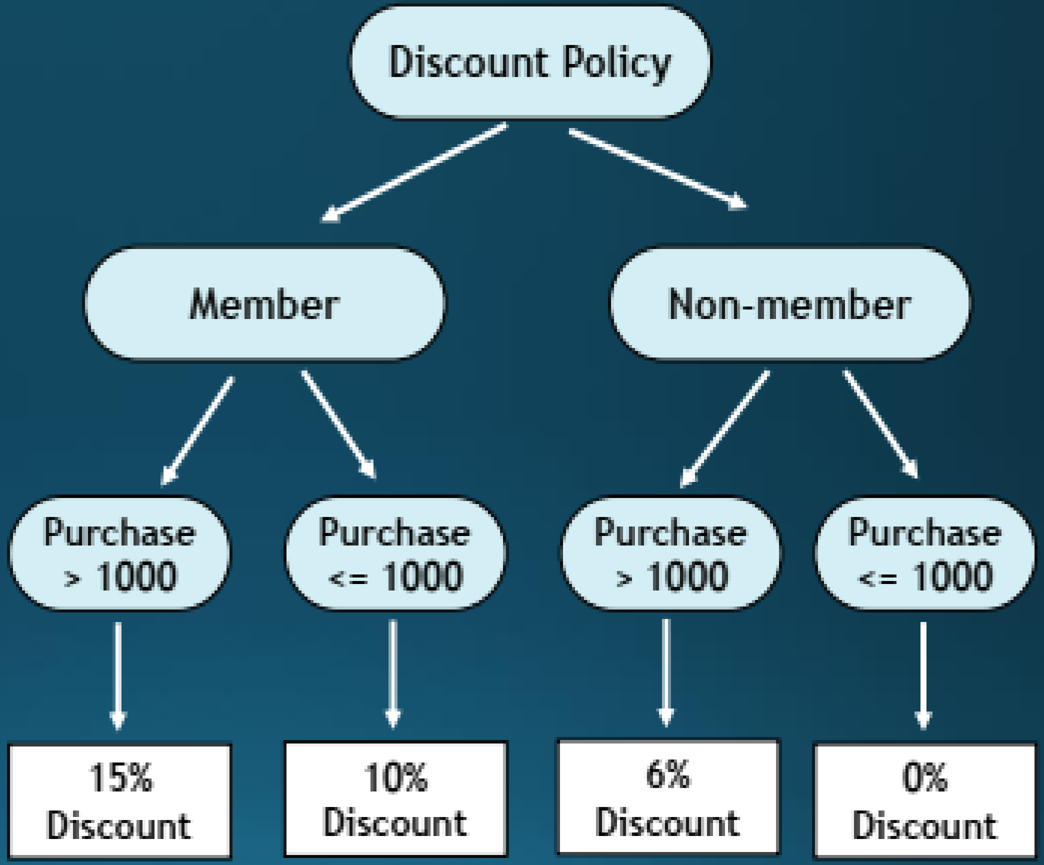
The discount policy of a departmental store is as follows:

i. If a customer is a member and purchase exceeds Rs. 1,000 then discount is 15%

ii. If a customer is a member and purchase is less than or equal to Rs. 1,000 then discount is 10%

iii. If a customer is not a member and purchase exceeds Rs. 1,000 then discount is 6%

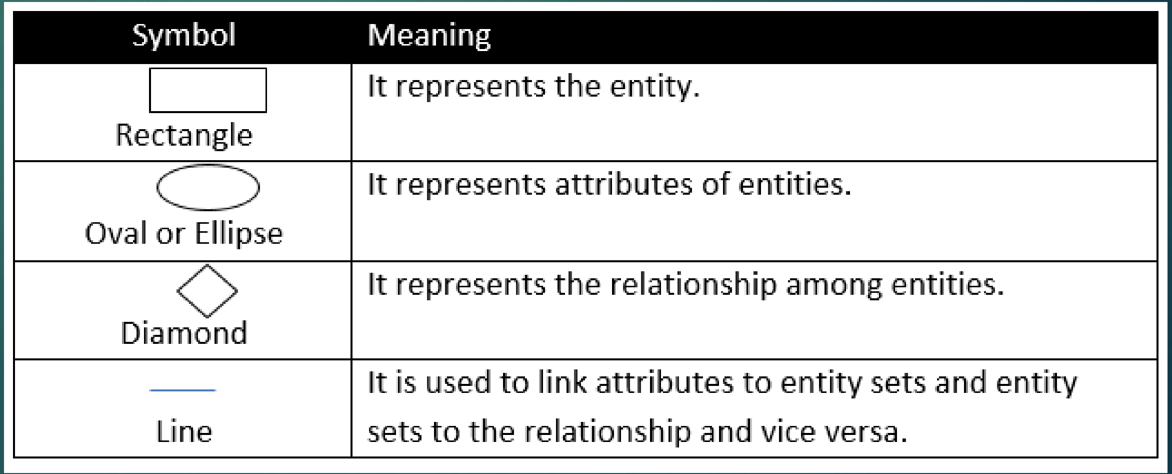
iv. If a customer is not a member and purchase is less than or equal to Rs. 1,000 then discount is 0%



E-R Diagram

An Entity–relationship model (ER model) describes the structure of a database with the help of a diagram, which is known as Entity Relationship Diagram (ER Diagram). An ER model is a design or blueprint of a database that can later be implemented as a database. ER diagram shows the complete logical structure of a database. The main components of the E-R model are:

- ❑ Entity Set
- ❑ Relationship Set.



Symbol	Meaning
▭ Rectangle	It represents the entity.
⬭ Oval or Ellipse	It represents attributes of entities.
◇ Diamond	It represents the relationship among entities.
— Line	It is used to link attributes to entity sets and entity sets to the relationship and vice versa.

Example of ER Diagram

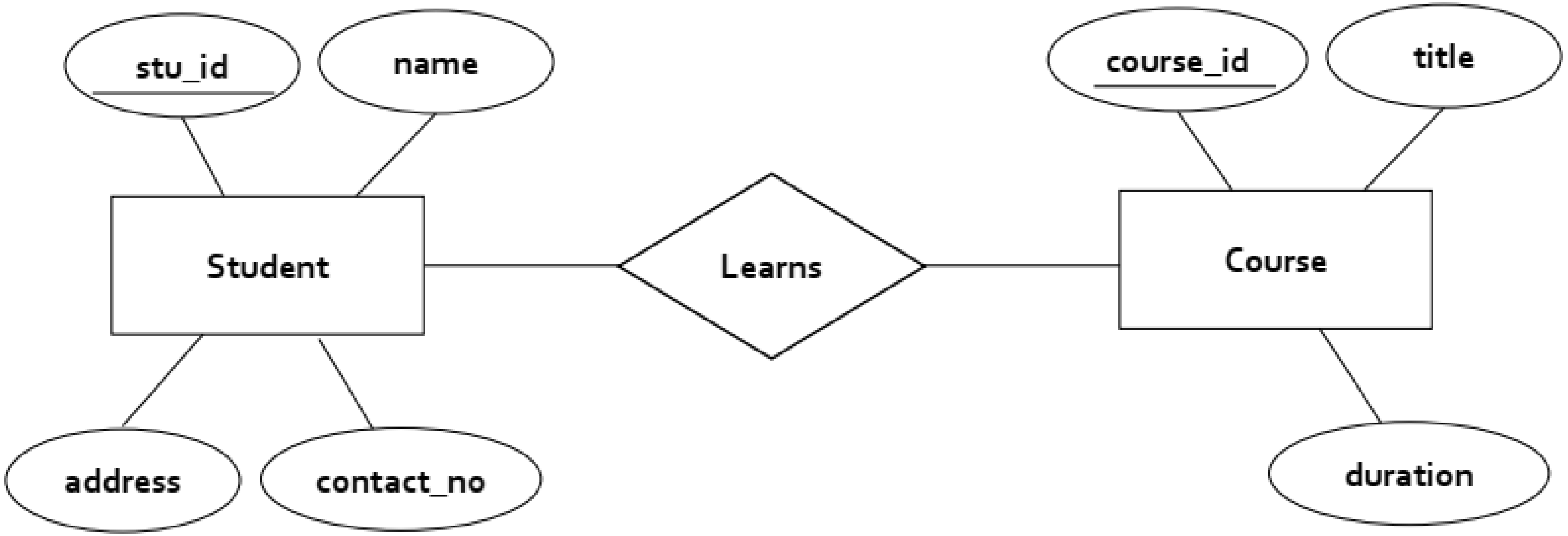
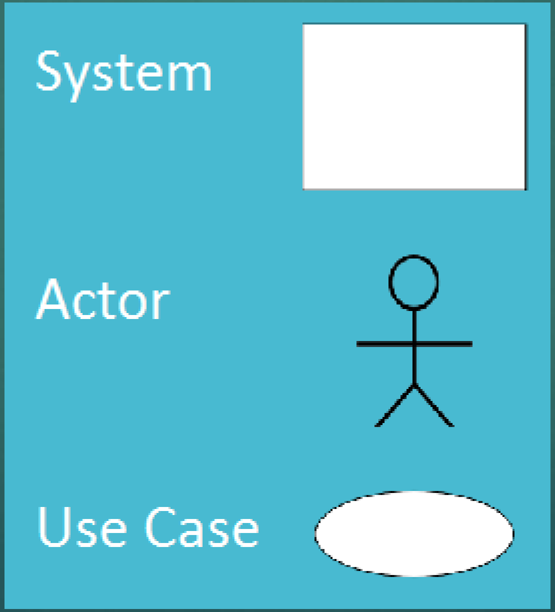


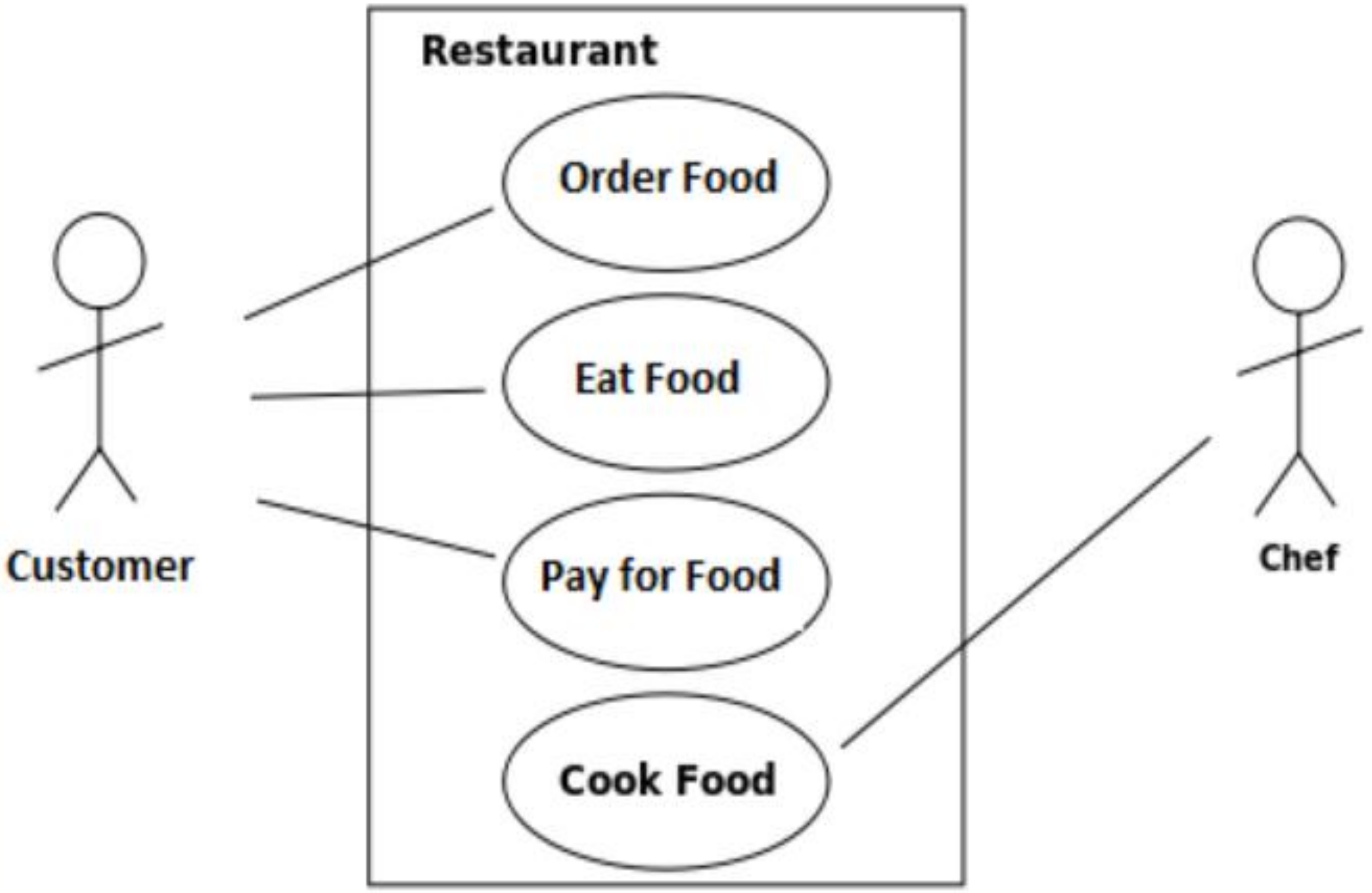
Fig: ER Diagram showing relationship between student and course

Case/ Use Case Tool

A Use Case is a set of scenarios that describe an interaction between a user and a system. It is a high level visualization of how the system works. Symbol Used in Data Flow Diagram:



Example of Use Case Diagram



Software and quality

The quality of software can be defined as the ability of the software to function as per user requirement. When it comes to sotware products it must satisfy all the functionalities written down in the document.

Key Aspects that conclude software quality include:

- ❑ Good Design
- ❑ Reliability
- ❑ Durability
- ❑ Consistency
- ❑ Maintainability
- ❑ Value of money

Software Development Model

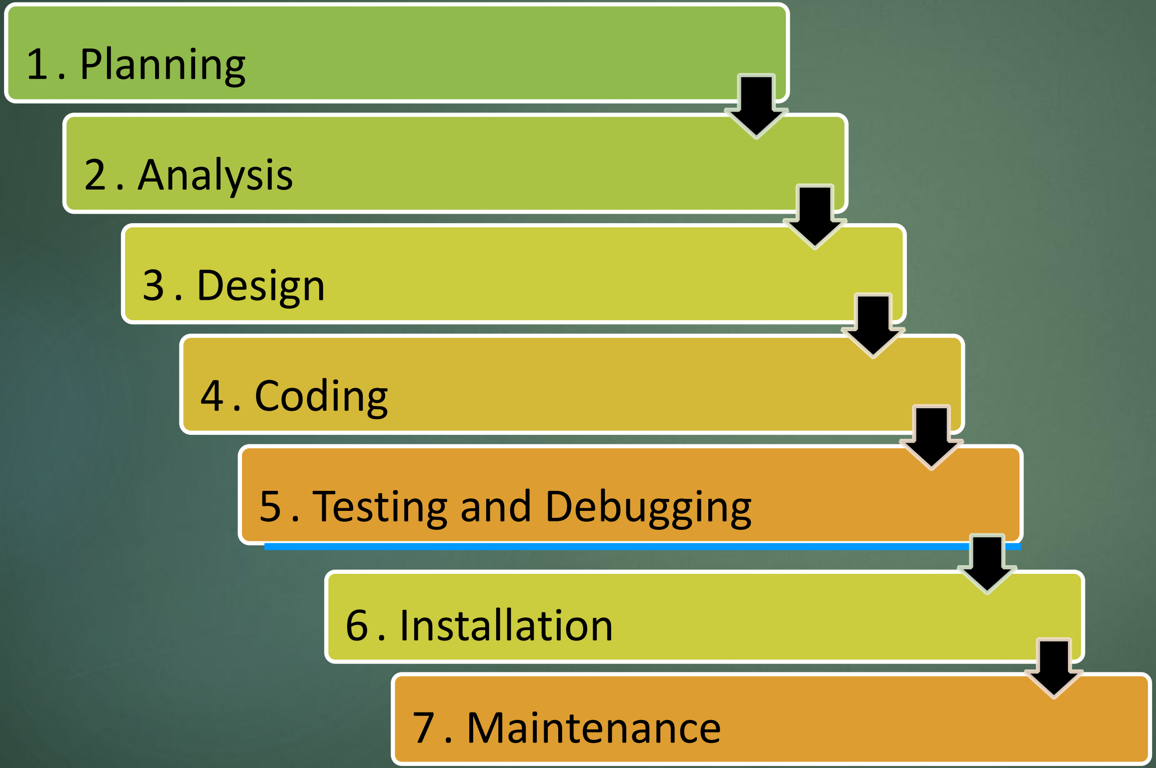
Software Development Model

- System development models specify how the activities of development process are organized in the total system development.
- Some Popular System Development Models are:
  - Waterfall Model
  - Prototype Model
  - Agile Model
- Every model has some advantages and limitations, so no model is perfect.

Waterfall Model

- The Waterfall Model was the first Process Model to be introduced.
- It illustrates the software development process in a linear sequential flow. Therefore, it is also referred to as a linear-sequential life cycle model.
- It is very simple to understand and use.
- In a waterfall model, each phase must be completed before the next phase can begin and there is no overlapping in the phases. This means that any phase in the development process begins only if the previous phase is complete.

Waterfall Model> Phases:



Advantages of Waterfall Model

- ❑ It is simple model suitable for small size project.
- ❑ It is less expensive.
- ❑ A Schedule can be set with deadlines for each state of production.
- ❑ Each phase of development processed in order, without overlapping the phase

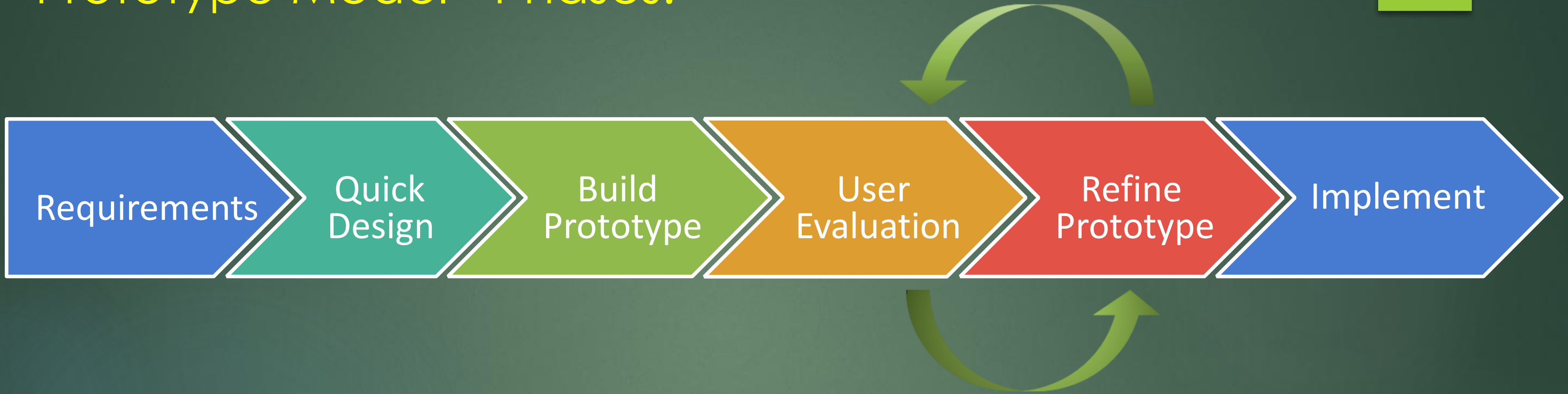
Disadvantages of Waterfall Model

- ❑ It has no back trach mechanism.
- ❑ It is not suitable for large size project.
- ❑ It has lack of proper documentation.

Prototype Model

- ❑ Prototyping refers to the activity of creating prototypes of software applications, for example, incomplete versions of the software program being developed.
- ❑ This model is useful to systems where requirements are not known in detail during the development.
- ❑ If developers have the authority to alter or change earlier decision, then this model can be implemented to develop the system or software.
- ❑ Users are actively involved in development. Therefore, errors can be detected in the initial stage of the software development process.

Prototype Model> Phases:



Advantages of Prototype Model

- ❑ Communication between the systems analyst and user are improved.
- ❑ The user plays a more active role in system development.
- ❑ Implementation is much easier because the user knowns what to expect.

Disadvantages of Prototype Model

- ❑ There is no completion deadline of system.
- ❑ It is expensive.
- ❑ This methodology may increase the complexity of the system as the scope of the system may expand beyond original plans.

Agile Model

- Agile SDLC model is a combination of iterative and incremental process models with focus on process adaptability and customer satisfaction by rapid delivery of working software product.
- Agile Methods break the product into small incremental builds. These builds are provided in iterations.
- Customer satisfaction with frequent delivery of working product.
- Suitable for systems where requirements change frequently
- Gives ‘power’ to users to change their mind and send new requirements
- Emphasize the use of latest design and technologies

Agile Model> Phases:

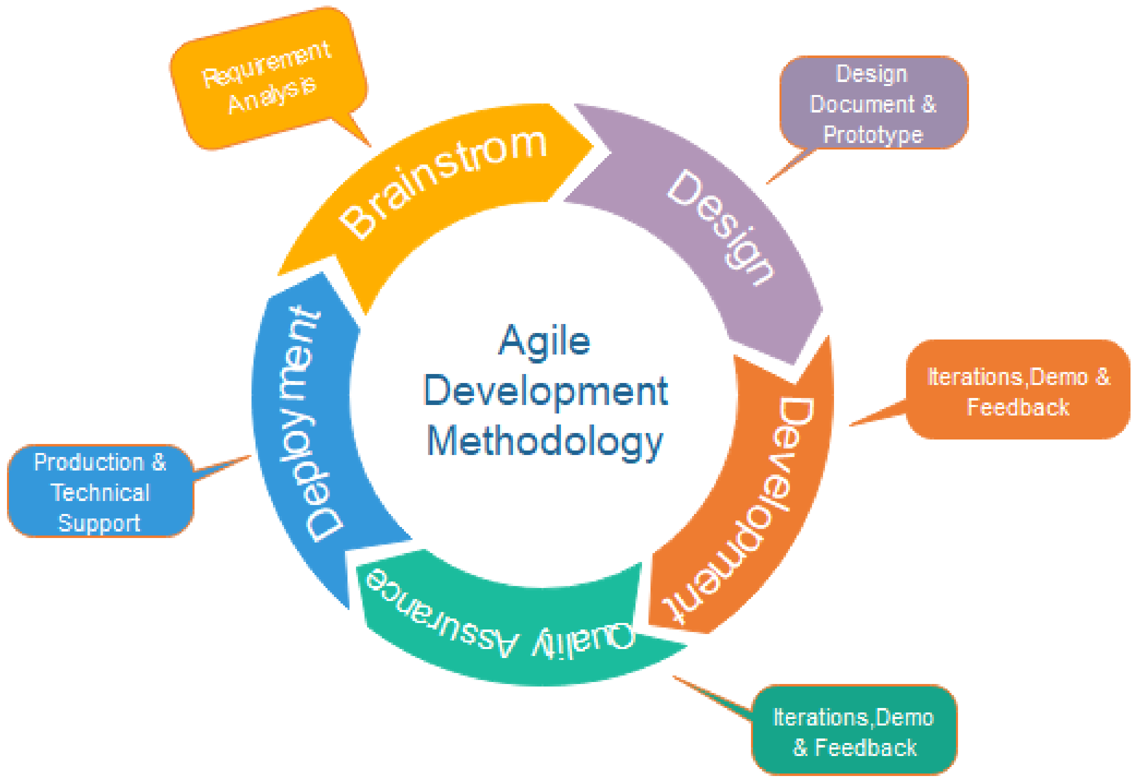

Requirement Analysis
Design Document & Prototype
Brainstrom
Design
Agile Development Methodology
Iterations,Demo & Feedback
Development
Production & Technical Support
Deployment
Quality Assurance
Iterations,Demo & Feedback

Fig. Agile Model

Advantages of Agile Model

- ❑ Promotes teamwork and cross training.
- ❑ Project can be developed rapidly.
- ❑ Suitable for fixed or changing requirements.
- ❑ Delivers partial working solutions.
- ❑ Gives flexibility to developers.

Disadvantages of Agile Model

- ❑ There is a very high individual dependency.
- ❑ Transfer of technology to new team members may be quite challenging.

Why Documentation is Important in Software Development?

Documentation is a critical component of software development for several reasons. It serves as a guide to ensure the software is understandable, maintainable, and scalable. Here are some of the key reasons why documentation is so important in the development process:

- ❑ Knowledge Transfer and Onboarding
- ❑ Improves Code Quality
- ❑ Facilitates Maintenance and Debugging
- ❑ Ensures Consistency and Standardization
- ❑ Supports Collaboration
- ❑ Regulatory and Compliance Requirements
- ❑ Helps with Testing and Quality Assurance
- ❑ Improves Communication with Stakeholders
- ❑ Supports Long-Term Success
- ❑ Promotes Best Practices