

Unit 6

SOFTWARE PROCESS MODEL

Concept of System

- The word “System” is derived from the Greek word 'Systema' which means 'an organized relationship among components'.
- A collection of components or elements that work together to perform a specific task is called a system.
- A system is a set of inter-dependent components which collectively accomplish certain objectives.
- Hence, it is a combination of resources working together to transform inputs into meaningful and usable outputs.

Concept of System

- Computerized information system is an organized set of hardware, software and different people to transform given data resources into meaningful and useful information for end users
- The environment or periphery around the system is called system environment.
- It affects the system performance.
- The border line around the system that separates the system with its environment is called system boundary.

Concept of System

- Some components of a system are:
 - Entity: It is an object of interest which takes part in system.
 - Attribute: It is property or characteristics of an entity. It describes the entity.
 - Activity: It represents all activities occurring within the specified time duration.
 - Event: It is instantaneous occurrence of activity that may change the state of system.

Concept of Project

- A project is well-defined tasks, which is a collection of several operations done in order to achieve a goal.
- Every project creates a unique product or service
- Project can be characterized as:
 - Every project may have unique and distinct goal
 - Project is not day-to-day operations
 - Project comes with a start time and end time.
 - Project ends when its goal is achieved
 - Project needs sufficient resources in terms of time, manpower, finance, materials etc.

Software Development Process

- Software development process is the process of creating computer software product.
- It is systematic operation that includes designing, preparing the specifications, programming, testing, bug fixing and documentation.
- It offers defined international structure for developer team to follow in design, creation and maintenance high quality softwares

SDLC Life Cycle

- SDLC stands for Software Development Life Cycle which consists of a detailed plan describing development, maintenance, replace, alter and enhance specific software
- Life cycle defines a methodology for improving the quality of software and overall development process
- Every phase of SDLC life cycle has its own process and output that feed into the next phase

SDLC Life Cycle

Phases of SDLC

- System study
- System analysis
- System design
- System coding
- System testing
- System implementation
- System maintenance & review

Importance of SDLC

- SDLC is a set of steps that serves as the basis for most system analysis and design methodologies.
- SDLC provides guidelines to follow for completing every activity in the system development process.
- It is a systematic approach to solve business problems.
- The steps in SDLC enhance management control, providing a framework for scheduling, budgeting and project management
- The tools associated with SDLC make it easier to solve the problem and helps to find errors early

Importance of SDLC

Some of the importance of SDLC can be listed as below:

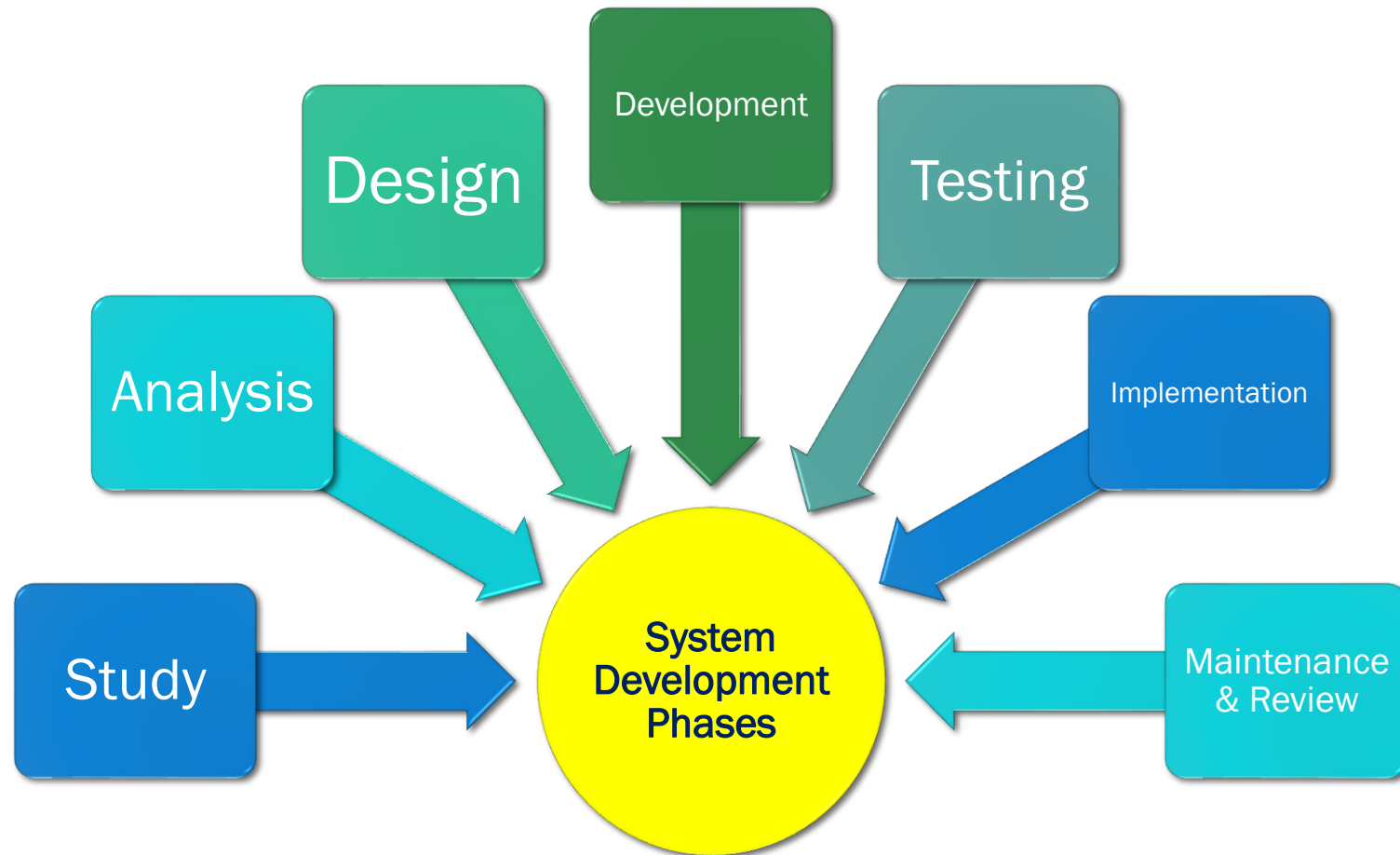
- SDLC is closely linked to structured system analysis and design.
- SDLC provides an explicit way of structuring i.e. it tells what to do, when to do, how to do, why to do in particular order.
- It encompasses some important aspects such as different phases, procedures, rules, techniques, tools, documentation, management etc.
- It maintains standards and management control.

Importance of SDLC

Some of the importance of SDLC can be listed as below:

- It breaks down entire software development process cycle which makes easier to evaluate and debug programs
- It provides guideline to complete each activity systematically for quality system development
- It gives clear idea to developer and save time and cost
- It provides proper documentation so designer can set and use functions, modules
- It allows to set primarily flexibility and contain a lot of innovation.

Phases of SDLC



Phases of SDLC

1. System study

It is the initial and one of the most important stages of SDLC.

The development team studies the present system and identifies the drawbacks.

They interact with customers and recognize the problems of existing system.

The development team proposes the new system based on this study.

Phases of SDLC

1.1 Feasibility Study

After the development team proposes the new system, the feasibility study is performed in order to determine whether the new system is feasible or not.

The testing is done on the basis of time, cost, technical and operational aspects.

A 'feasibility survey report' is made after completion of feasibility study.

Phases of SDLC

1.1 Feasibility study

- **Time feasibility/Schedule feasibility:** It is concerned with the time required for the development of new system.
- **Cost feasibility:** It is concerned with the total cost for the development of new system. It determines whether the organization can afford the total cost or not.
- **Technical feasibility:** It is concerned with specifying different devices and software for the new system. If all the technical requirements for the new system can be fulfilled, then the development of new system will be feasible.

Phases of SDLC

1.1 Feasibility study

- **Operational feasibility:** It is mainly related with human skills and administrative process. If the staffs need very long time and more cost to be trained in the new system, then the new system will not feasible.
- **Legal feasibility:** It is mainly focus to analyze if any violation of government laws has committed or not.
- **Economic feasibility:** This economic feasibility is done to check whether the system is economic during the operation time on the client side. If operation cost, manpower cost and other costs are within the limit then the new system is said to be economically feasible.

Phases of SDLC

2. System Analysis

- System analysis is a process of collecting accurate data, understand the process involved, identifying problems and recommending feasible suggestions for improving the system functioning.
- This involves studying the business processes, gathering operational data, understand the information flow, finding out bottlenecks and evolving solutions for overcoming the weakness of the system so as to achieve the goals
- Development team studies minutely to collect all the drawbacks and details of information from the users, management and data processing personnels

Phases of SDLC

2. System Analysis

System analyst performs following:

- Gather, analyze and validate information
- Define requirements and prototypes for new system
- Evaluate the alternatives and prioritize the requirements
- Examine information needs of end-user and enhances the system goal
- Software Requirement Specification (SRS) document, which specifies the software, hardware, functional and network requirements for the system

Phases of SDLC

3. System Design

- The system design involves designing of a new system that will meet the requirements identified during system analysis.
- It is the most creative and challenging phase of SDLC.
- System designing involves input design, output design, file system design, database design etc.
- System design can be of two types:
 - Logical design
 - Physical design

Phases of SDLC

3. System Design

Logical design:

- Designing the theoretical logics or business logics is called logic design.
- The logical requirements of system is defined for the further designing of the proposed system.
- It deals with the logical part of the system design.

Phases of SDLC

3. System Design

Physical design:

- The conversion of logical design into designing tools and techniques is called physical design.
- It is more detail and complex jobs describing the solution of problem.
- Physical design includes algorithms, flowchart, pseudocodes, decision table, decision tree, ER diagram, Data flow diagram (DFD) etc.

Phases of SDLC

4. System Development

- After designing the system, the actual system development process starts.
- Programmer has to choose suitable programming language to develop the program.
- It is programmer's responsibility to detect and fix syntactical and logical errors.
- The developed system is compiled and executed.

Phases of SDLC

5. System Testing

- After developing the system, each and every modules are tested individually and debugged.
- When the modules are bug free, they are integrated as a single system and is tested entirely.
- If the entire system is bug free and can fulfill the requirements, then it is ready to implement.
- There are two types of system testing methods:
 - While Box Testing/Glass Box Testing
 - Black Box Testing/Functional Testing

Phases of SDLC

5. System Testing

White Box Testing:

- It is a method of testing software that tests internal structures or workings of an application.
- It is a testing technique that examines the program structure and derives test data from the program logic/code.
- The other names of glass box testing are clear box testing, open box testing, logic driven testing or path driven testing or structural testing.

Phases of SDLC

5. System Testing

Black Box Testing:

- It is a method of software testing that examines the functionality of an application without peering into its internal structures or workings.
- Black-box testing is a method of software testing that examines the functionality of an application based on the specifications.
- It is also known as Specifications based testing.
- Independent Testing Team usually performs this type of testing during the software testing life cycle.
- This method of test can be applied to each and every level of software testing such as unit, integration, system and acceptance testing.

Phases of SDLC

5. System Testing

Types of Software Testing

1. Unit Testing
2. Integration Testing
3. System Testing
4. Acceptance Testing

Phases of SDLC

5. System Testing

A. Unit Testing

- Unit testing performed on each module or block of code during development.
- Unit testing is normally done by the programmer who writes the code

Phases of SDLC

5. System Testing

B. Integration Testing

- Integration testing is done before, during and after integration of a new module into the main software package.
- This involves testing of each individual code module
- One piece of software can contain several modules which are often created by several different programmers
- It is crucial to test each module's effect on the entire program

Phases of SDLC

5. System Testing

C. System Testing

- System testing is done by a professional testing agent on the completed software product before it is introduced to the market

D. Acceptance Testing

- Acceptance testing of the product is done by the actual end users.

Phases of SDLC

6. System Implementation

- After the new system is ready, then it is implemented in the organization.
- Application is installed or loaded on existing or new hardware and users are introduced to new system and trained.
- The software, application, website, portals are launched in this phase
- End user or customer will have chance to experience the new system

Phases of SDLC

6. System Implementation

- System implementation can be done in 3 ways.
 - **Direct:** Software is directly installed at user's site by replacing the old system. If the problem persists on new system, the user may face different problems
 - **Parallel:** Both new and old system are run in parallel for some time. After monitoring the new system for a reasonable period of time, if it is performing well, then the old system is replaced by new one.
 - **Phased:** System is installed module by module. If one module works efficiently then only another module is installed otherwise modification of first module is performed

Phases of SDLC

7. System Maintenance & Review

- When time changes, the requirements of the organization also gets changed and the system can no longer fulfill it.
- During maintenance, programmers make changes that users ask for and modify the system to reflect and support changing business condition
- These changes are necessary to keep the system running and useful.
- Maintenance is part of SDLC and it is repeated

Phases of SDLC

7. Maintenance & Review

Types of Maintenance

1. Corrective Maintenance
2. Adaptive Maintenance
3. Perfective Maintenance
4. Preventive Maintenance

Phases of SDLC

Types of Maintenance

A. Corrective Maintenance

- It corrects the source code of the system for omitting errors
- It is mainly used to remove errors as corrective measures
- The main aim of this maintenance is to remove bugs for system
- So, it is the process of diagnosing and correcting system after they occur

Phases of SDLC

Types of Maintenance

B. Adaptive Maintenance

- If the surrounding environment for the system is changed then certain changes should be made.
- These types of changes in adaptation from rules and regulations changed, policy of organization changed and from other factors is called adaptive maintenance

Phases of SDLC

Types of Maintenance

C. Perfective Maintenance

- It makes the system perfect, up-to-date and improve the life of the system
- The maintenance is performed to make the system perfect

Phases of SDLC

Types of Maintenance

D. Preventive Maintenance

- It makes the system prevent from failure in future
- The preventive measurement is applied to the system
- Its main concern activities are aimed on increasing system maintainability and prevent problems in future

Phases of SDLC

Review

- A software review is a process or meeting during which a software product is examined by project personal, managers, users, customers, user representatives or other interested parties for comment or approval
- As a general principle, a technical document is produced showing the progresses and activities involved in software development
- The review process is conducted according to market plan, contract signed and on the basis of requirement specifications

System Analyst

- System analyst is a chief person in system development team who analyzes and designs the new computerized information system.
- System analyst is the team leader and involves throughout all phases of the system development life cycle.
- System analyst is an IT professional who is involved in analyzing, designing, implementing and evaluating computer based information system.
- System analyst must have technical as well as organizational knowledge.

Roles of System Analyst

Change Agent

- System analyst may be viewed as agent of change.
- A system is designed to make changes on previous system.
- System analyst may use different approaches to introduce changes.

Investigator

- System analyst should investigate the existing system to find the reasons for its failure.
- He should extract the problems from existing system and monitor the program in relation to time, cost and quality.

Roles of System Analyst

Architect

- System analyst should play as architect in interface between user's logical design requirements and detailed physical system design.
- He must design detail physical design that fulfills the user's logical design requirement.
- The design becomes the blue print for the programmers.

Psychologist

- System analyst plays role of psychologist when he deals with users, interpret their thoughts and draw conclusion from these interactions.

Roles of System Analyst

Motivator

- System analyst plays the role of motivator in order to make the users accept the new system and make his team members work together for single goal.

Diplomat

- System analyst should deal people with diplomacy to improve acceptance of the system.

Responsibilities of System Analyst

Defining requirements:

- Being the main person in the system development team, a system analyst has to define the requirements of the system.
- The system analyst may use different fact finding techniques such as interview, questionnaire, field visit, observations etc. in order to define the requirements of the users or system.

Prioritizing requirements:

- After the requirements have been identified, system analyst need to set priority among the requirements.
- We can't fulfill or deal all the requirements with our limited manpower and resources.
- So it is system analyst's responsibility to prioritize the requirements.

Responsibilities of System Analyst

Analysis and Evaluation:

- The system analyst has to find out the drawbacks as well as strength of the current information system.
- He has to analyze and evaluate the current system to implement new system that can eliminate the drawbacks of current system.

Solving problems:

- System analyst has to solve all the problems that may occur during the software development process.
- The analyst must study the problems in depth and suggest the most appropriate solutions to it.

Responsibilities of System Analyst

Drawing functional specifications:

- System analyst is responsible for drawing the system's specification and requirements.
- The later developed system must meet the specifications of the system.

Design system:

- System analyst must design the new information system in an easy and understandable way for the implementer.
- He should design both logical and physical design. The design should be modular and flexible.

System Evaluation:

- System analyst must critically evaluate the new information system in order to find the drawbacks and errors.
- He must decide when to do evaluation and how to do it.

Characteristics of System Analyst

Knowledge of organization

- System analyst must have sound knowledge of working mechanism, management structure & functional relationship among departments, staffs of organization.
- He should understand daily operations, rules and regulations of the particular system

Technical knowledge

- System analyst must be familiar and well trained in recent relevant areas of computing technologies and updated systems
- He must be ready to advice development team which addresses user's need with higher level of efficiency

Characteristics of System Analyst

Good interpersonal relation

- System analyst must be good listener, diplomat and able to influence & resolve conflicts, understand needs and motivate team.

Interpersonal communication skill

- System analyst must be able to talk intelligently with high level management, technical and other staffs and programmers and influence them and change their mind and attitudes.

Analytical skill

- System analyst must be able to perceive the core problem & discard duplicate data.
- He must find the solution of problem using appropriate analytical tools

Characteristics of System Analyst

Breadth of knowledge

- System analyst should have knowledge about various types of peoples, their jobs, perception to handle the team

Character and ethics

- Ethics is personal character quality. Successful analyst must be professional, resourceful, inventive and creative.
- System analyst requires a strong character and sense of ethics.

Problem solving skill

- System analyst must have skills of defining and analyzing the problems, considering, evaluating and selecting the alternatives.

System Design Tools

- Data Flow Diagram (DFD)
- Entity Relationship Diagram (E-R Diagram)
- Flowchart
- Decision Table
- Decision Tree
- Unified Modeling Language (UML)
- Use Case Diagram

System Design Tools

Data Flow Diagram (DFD)

- DFD is a graphical tool that allows analysts to understand the flow of data in an information system.
- DFD is a picture of movement of data between external entities and the processes and data stores within the system.
- It is a traditional visual representation of the information flows within the system but does not show program logic or processing steps.
- It shows how information enters and leaves the system. DFD can be categorized into following.

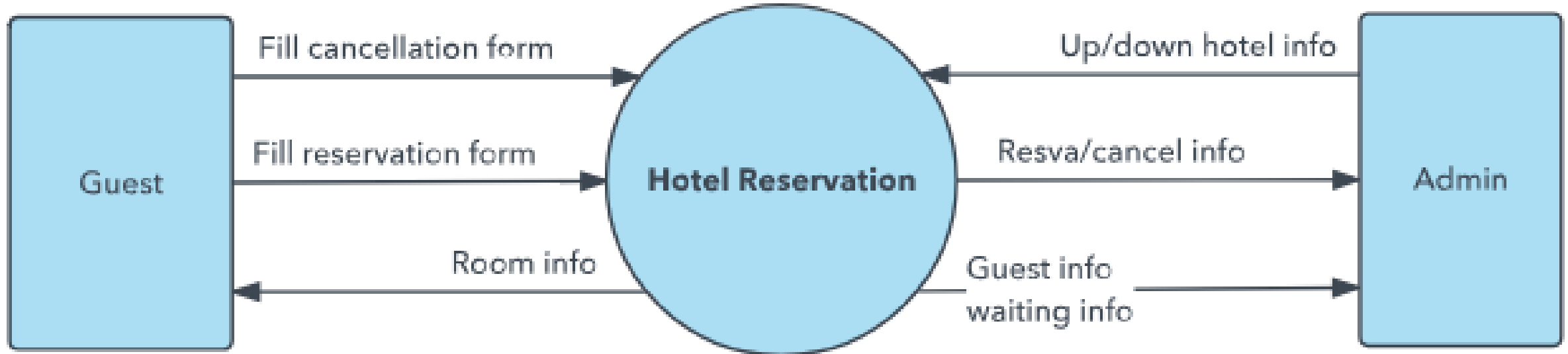
System Design Tools

Context Diagram (Level - 0 DFD)

- DFD Level 0 is also called a Context Diagram.
- It's a basic overview of the whole system or process being analyzed or modeled.
- It's designed to be an at-a-glance view, showing the system as a single high-level process, with its relationship to external entities.
- It should be easily understood by a wide audience, including stakeholders, business analysts, data analysts and developers.
- Context diagram has single process.

System Design Tools

Level - 0 DFD



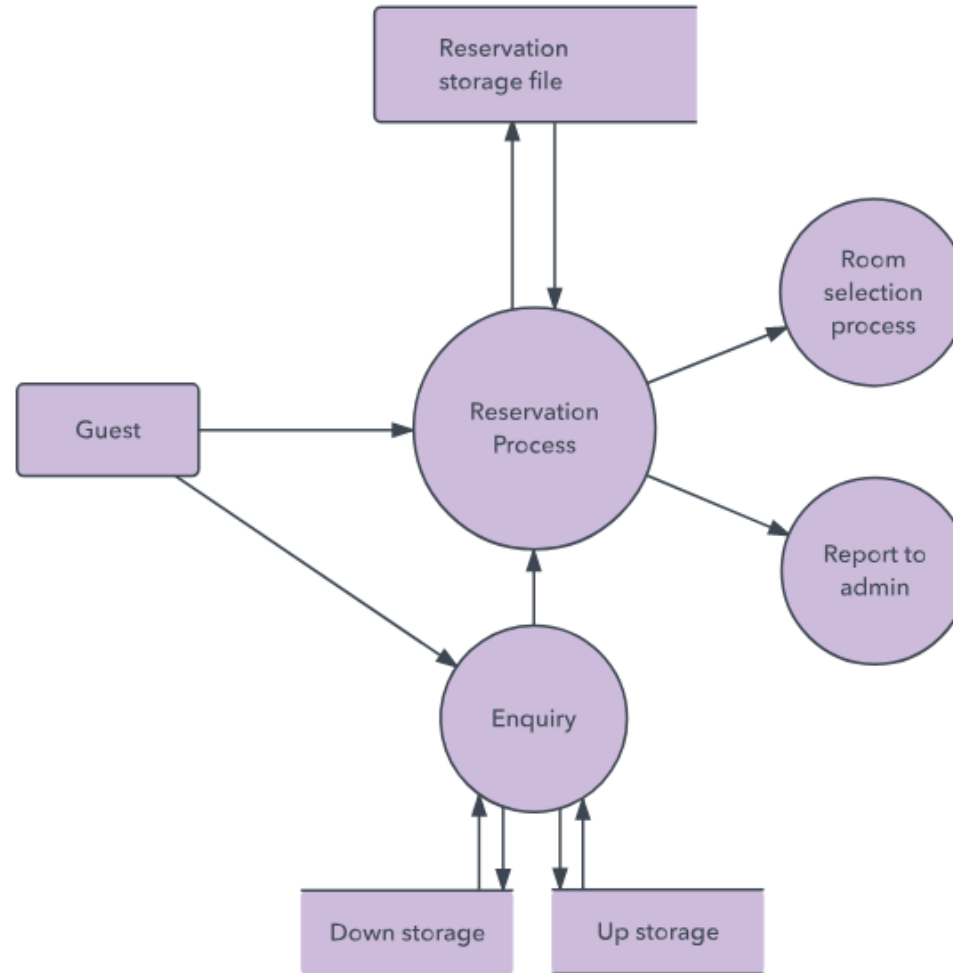
System Design Tools

Level - 1 DFD

- DFD Level 1 provides a more detailed breakout of pieces of the Context Level Diagram.
- You will highlight the main functions carried out by the system, as you break down the high-level process of the Context Diagram into its sub processes.

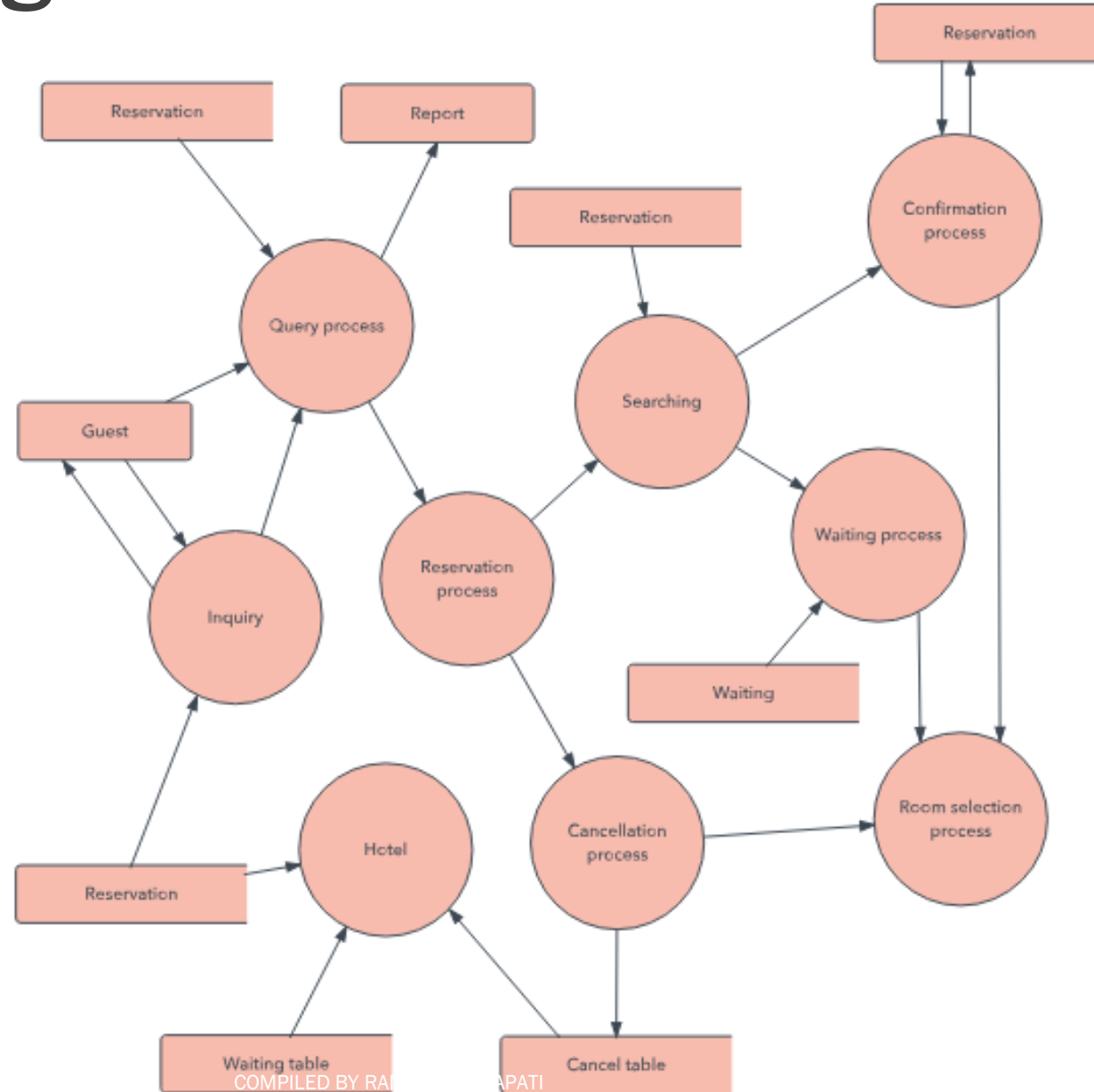
System Design Tools

Level - 1 DFD



System Design Tools

Level - 2 DFD



System Design Tools

Symbols and Notation used in DFD

- Two common systems of symbols are named after their creators:
 - Yourdon and Coad
 - Gane and Sarson
- One main difference in their symbols is that Yourdon-Coad and Yourdon-DeMarco use circles for processes, while Gane and Sarson use rectangles with rounded corners. There are other symbol variations in use as well.


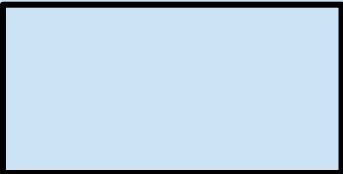
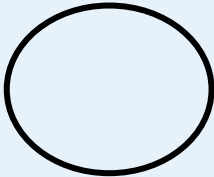
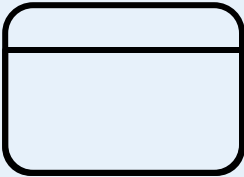


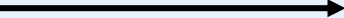

System Design Tools

- **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.
- **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.”

System Design Tools

- **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.”
- **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.”

System Design Tools

Notations	Yourdon and Coad	Gane and Sarson
External Entity		
Process		
Data Store		
Data Flow		

System Design Tools

DFD Rules

- Each process should have at least one input and an output.
- Each data store should have at least one data flow in and one data flow out.
- Data stored in a system must go through a process.
- All processes in a DFD go to another process or a data store.

System Design Tools

E-R Diagram

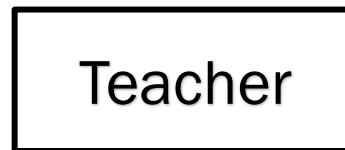
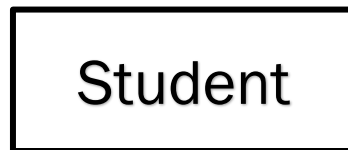
- An Entity Relationship (ER) Diagram is a type of chart that illustrates how “entities” such as people, objects or concepts relate to each other within a system.
- ER Diagrams are most often used to design or debug relational databases in the fields of software engineering, business information systems, education and research.
- Also known as ERDs or ER Models, they use a defined set of symbols such as rectangles, diamonds, ovals and connecting lines to depict the interconnectedness of entities, relationships and their attributes.
- They mirror grammatical structure, with entities as nouns and relationships as verbs.

System Design Tools

Components of E-R Diagram

Entity:

- Entity is a physical or conceptual object like person, place, event, job etc
- One entity is related to another entity
- Entities are represented by means of rectangles.
- Rectangles are named with the entity set they represent.

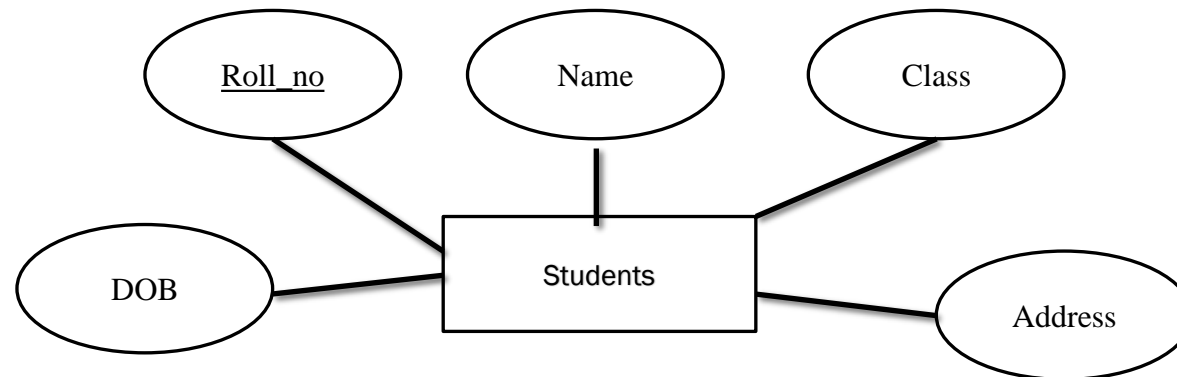


System Design Tools

Components of E-R Diagram

Attributes:

- Attribute is a descriptive property or characteristics of an entity
- Attributes are the element, properties of entities.
- Attributes are represented by means of ellipses.
- Every ellipse represents one attribute and is directly connected to its entity (rectangle).

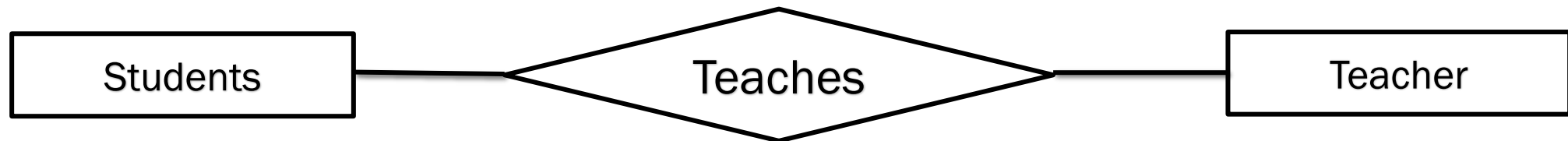


System Design Tools

Components of E-R Diagram

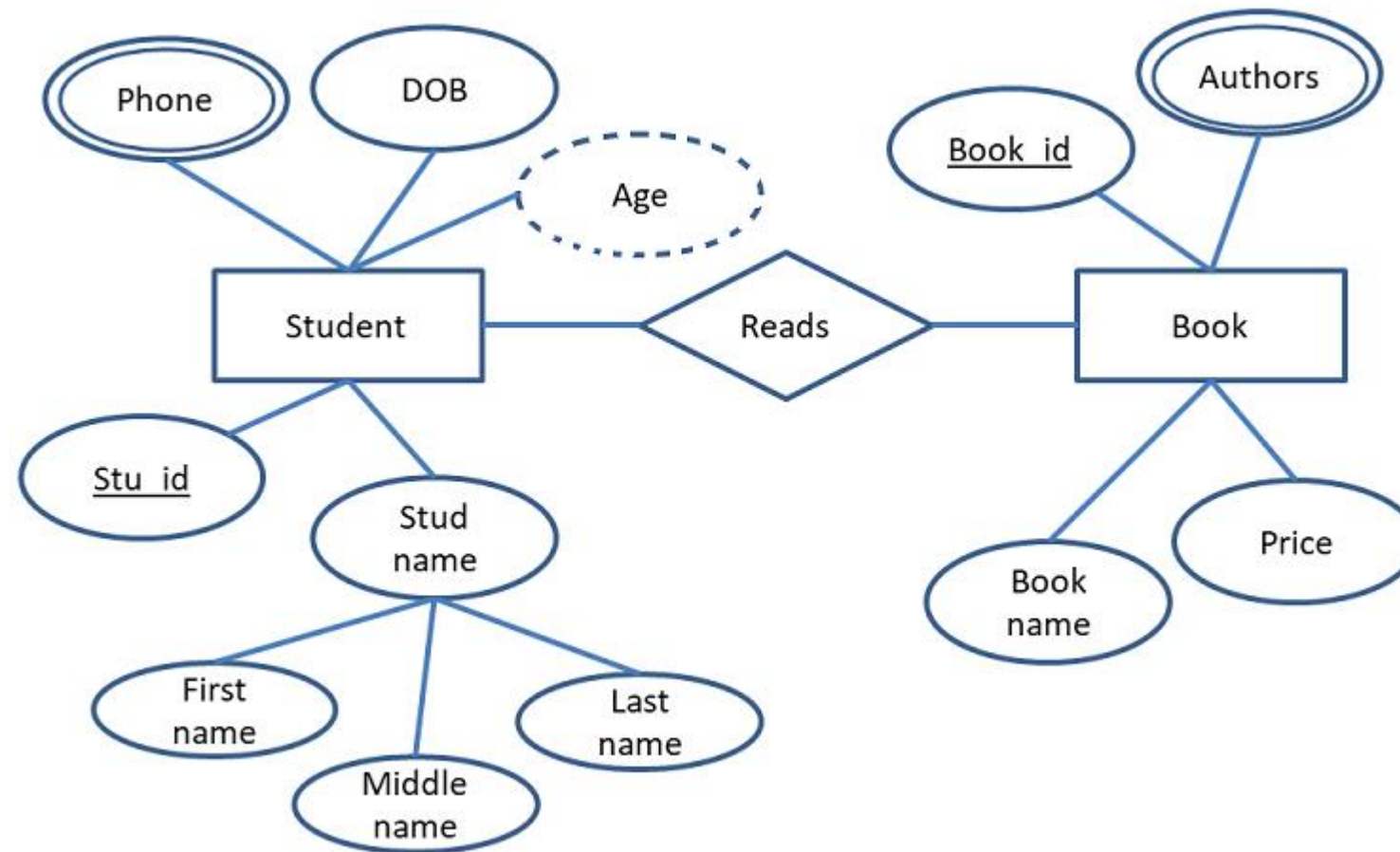
Relationship:

- Relationships are represented by diamond-shaped box.
- Name of the relationship is written inside the diamond-box.
- All the entities (rectangles) participating in a relationship, are connected to it by a line.

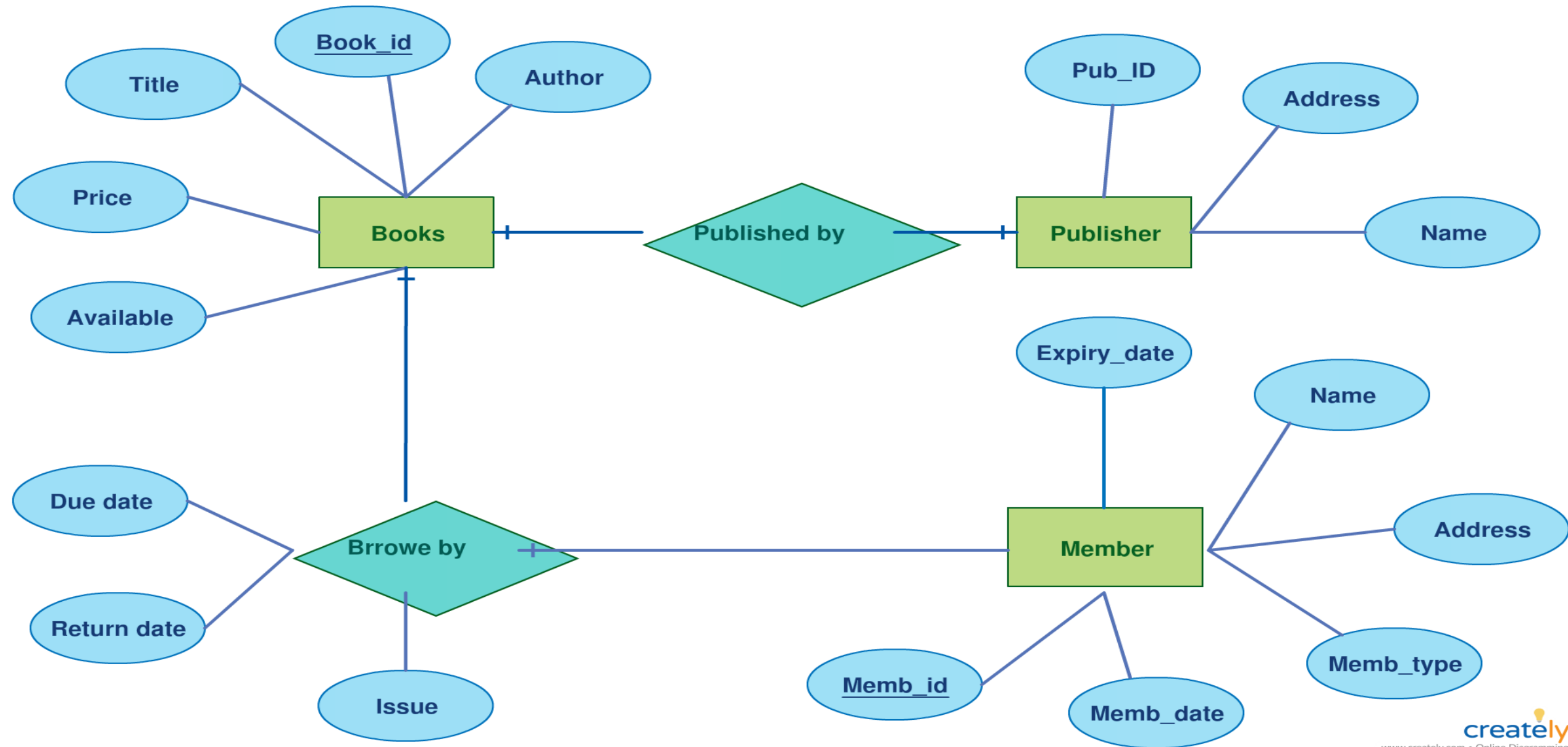


System Design Tools

ER Diagram



E-R Diagram for Library Management System




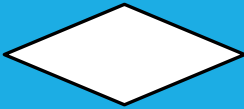




System Design Tools

Flowchart

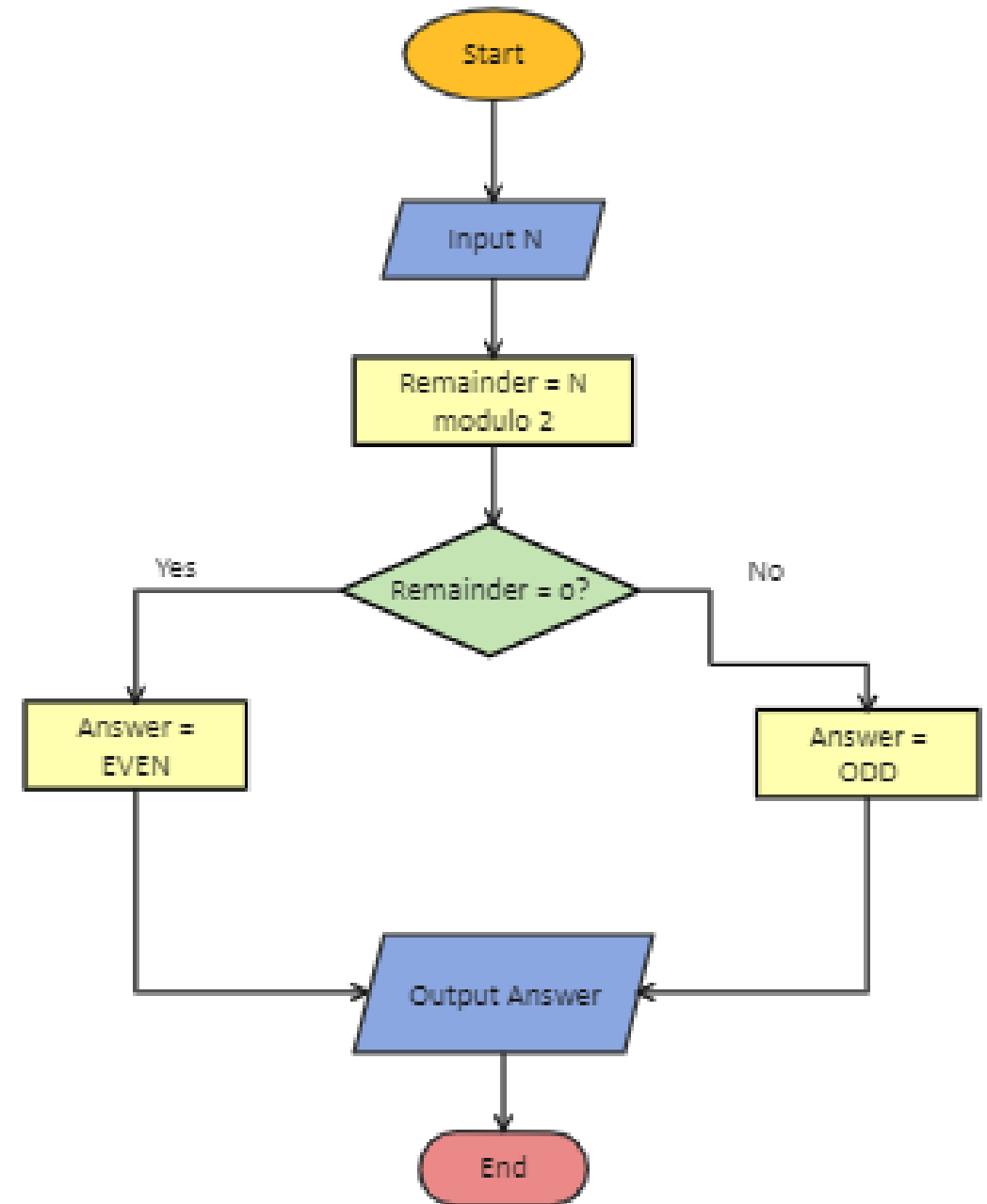
- A flowchart is a visual representation of the sequence of steps and decisions needed to perform a process.
- Each step in the sequence is noted within a diagram shape.
- Steps are linked by connecting lines and directional arrows.
- This allows anyone to view the flowchart and logically follow the process from beginning to end.
- A flowchart is a powerful business tool with proper design and construction, it communicates the steps in a process very effectively and efficiently.

System Design Tools

Symbols	Meaning
	Start/End
	Input/Output
	Process
	Condition
	Connector
	Control Flow

System Design Tools

Flowchart



System Design Tools

Decision Table

- Decision table testing is a testing technique used to test system behavior for different input combinations.
- This is a systematic approach where the different input combinations and their corresponding system behavior (Output) are captured in a tabular form.
- That is why it is also called as a **Cause-Effect table** where Cause and effects are captured for better test coverage.
- A Decision Table is a tabular representation of inputs versus rules/cases/test conditions. Let's learn with an example.

System Design Tools

Decision Table

- Decision table uses a standard format and handle combination of conditions in a very concise manner.
- There are 3 parts in decision table
 1. **Condition Stub:** This part of table contains the various conditions that apply in the situation the table is modeling
 2. **Action Stub:** This part of table lists the actions that result for a given set of conditions
 3. **Rules:** This part of table specifies which actions are to be followed for a given set of conditions.

System Design Tools

Decision Table

Example 1: Decision Base Table for Login Screen.

- The condition is simple if the user provides correct username and password the user will be redirected to the homepage. If any of the input is wrong, an error message will be displayed.

Process Name		Rule 1	Rule 2	Rule 3	Rule 4
Conditions	Is Username Correct?	F	T	F	T
	Is Password Correct?	F	F	T	T
Actions	Show Error Message.	T	T	T	F
	Show Home page.	F	F	F	T

System Design Tools

Decision Table

Interpretation:

- **Case 1** – Username and password both were wrong. The user is shown an error message.
- **Case 2** – Username was correct, but the password was wrong. The user is shown an error message.
- **Case 3** – Username was wrong, but the password was correct. The user is shown an error message.
- **Case 4** – Username and password both were correct, and the user navigated to homepage

System Design Tools

Decision Tree

- A decision tree is a graph that uses a branching method to illustrate every possible outcome of a decision.
- It is also a technique to represent condition and actions in a diagrammatic form in computer.
- The diagram resembles the branches of tree.
- The root of the tree is the starting point of the decision sequence and progression from the left to right along a particular branch is the result of making a series of decisions.

System Design Tools

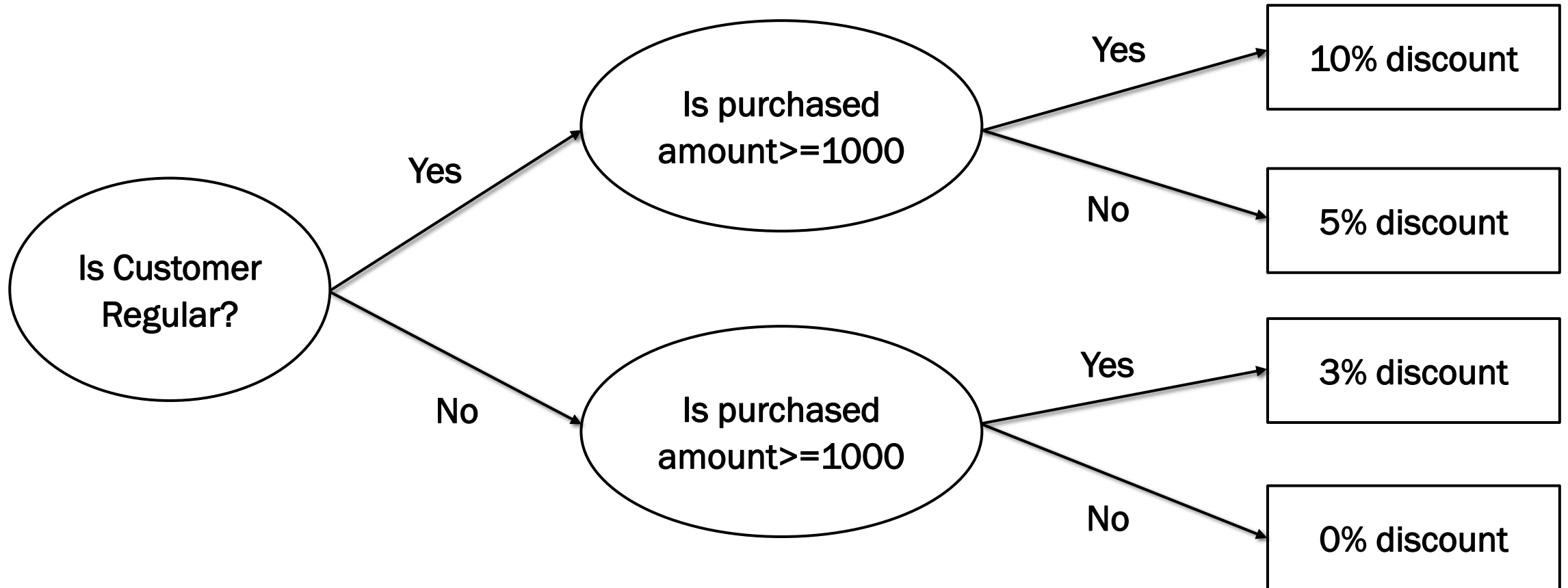
Decision Tree

An example of decision tree to show the calculations of discount policy

- if the customer is Regular and purchased amount ≥ 1000 then 10% discount
- if the customer is Regular and purchased amount < 1000 then 5% discount
- if the customer is not Regular and purchased amount ≥ 1000 then 3% discount
- if the customer is not Regular and purchased amount < 1000 then 0% discount

System Design Tools

Decision Tree



System Design Tools

Use Case Diagram

- A Use Case diagram is a graphic representation of the interactions among the elements of a system.
- It is a methodology used in system analysis to identify, clarify and organize system requirements.
- Use case diagram represents defines what can be done in system instead of defining how it is done.
- A use case diagram is usually simple and maintains following conditions.
- It only summarizes some of the relationships between use cases, actors and systems.

System Design Tools

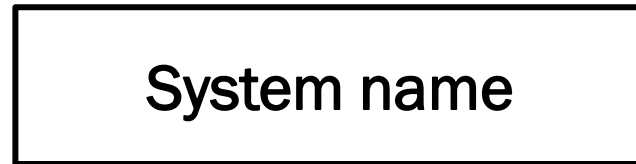
Use Case Diagram

- It does not show the order in which steps are performed to achieve the goals of each use case.
- Use case represents only functional requirements of a system.
- Use case diagram consists of **actors** and **actions**
- **actor:** It is an external entity that interacts with the system. It is someone or something that exchanges information with the system
- **Action:** It represents a sequence of related tasks initiated by an actor to accomplish a specific goal. It is specific way of using the system.

System Design Tools

System

- System boundaries are drawn using a rectangle that contains use cases. Actors are placed outside the system boundaries



Use Case

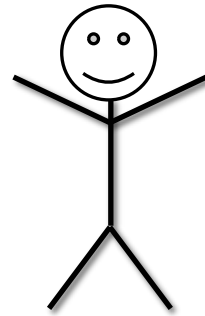
- Use case is drawn using ovals. System functions are labelled with verbs.



System Design Tools

Actors

- Actors are the users of a system. Actor or user interacts with the system

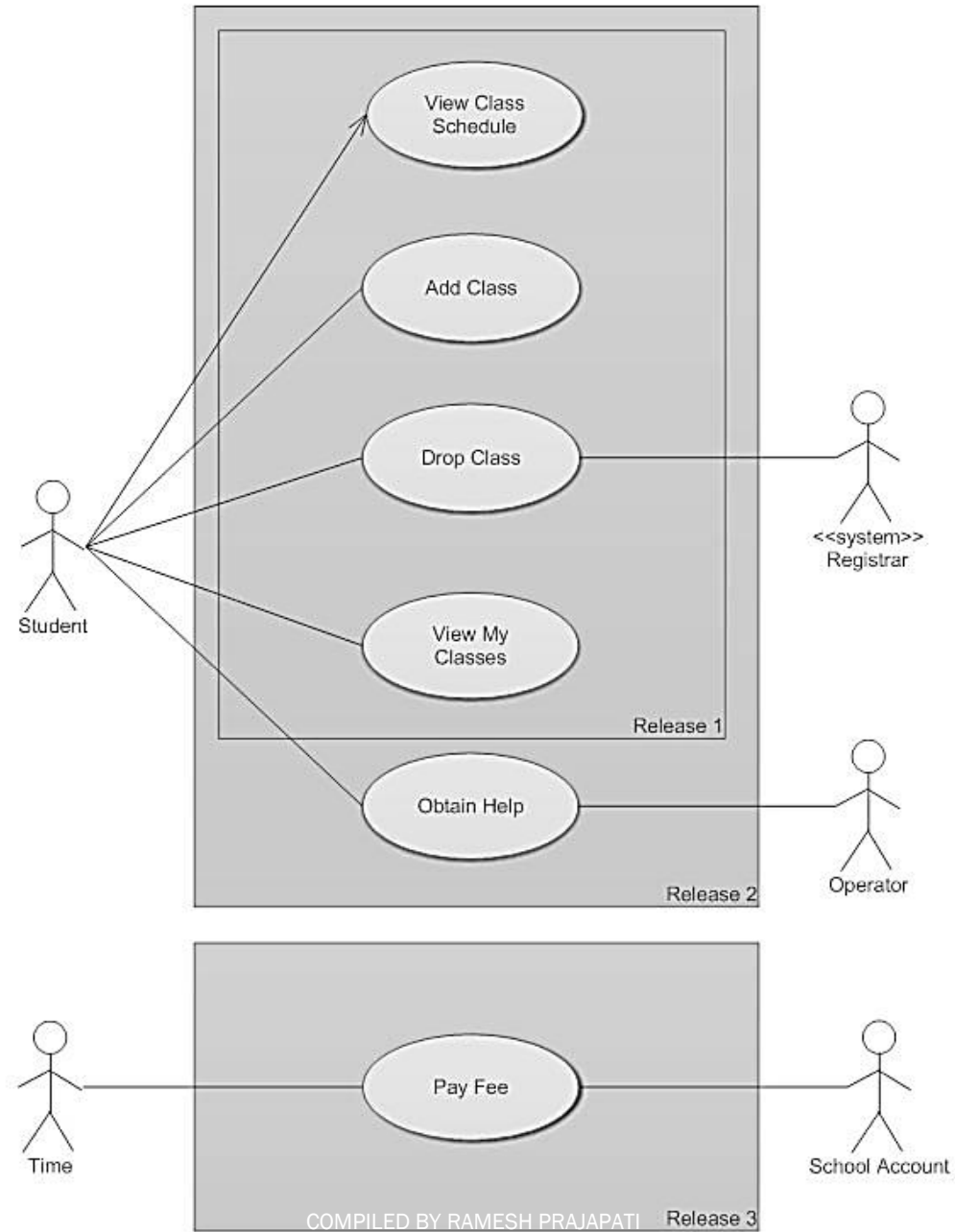


Relationship

- Relationships between an actor and a use case is illustrated with a simple line.



Use Case Diagram: Class Registration



System Design Tools

Unified Modeling Language (UML)

- Unified Modeling Language (UML) was developed by Grady Booch, Ivar Jacobson and James Rumbaugh at Rational Software in 1990's.
- It was adopted by Object Management Group (OMG) in 1997. It is a standardized, general purpose modeling language in the field of software engineering.
- It includes a set of graphic notation technique to create visual models of object oriented software intensive system.

System Design Tools

Unified Modeling Language (UML)

- UML diagram has ability to represent complex relationship as well as to represent data and data processing with consistent notation
- There are various techniques such as Use Case, Class diagram, State diagram, sequence diagram
- All these techniques and associated notations are incorporated into a standard object-oriented language called Unified Modeling Language (UML)

System Design Tools

Unified Modeling Language (UML)

- UML is language for specifying, visualizing and constructing the artifacts of software systems as well as for business modeling
- UML notation is useful for geographically depicting object oriented analysis and design models
- It specifies requirements of system and promotes communication among key persons involved in the development efforts.

Quality of Software

- Quality software refers to software which is reasonably bug or defect free, is delivered in time and within the specified budget, meets the requirements or expectation and is maintainable.
- It is defined as the ability of the software to function as per user requirement.
- **Good Design:** It is important to have a good and beautiful design to satisfy users
- **Reliability:** It should be able to perform the functionality perfectly without issues
- **Durability:** It is the ability to work without any issue for a long period of time

Quality of Software

- **Consistency:** It should perform consistently over platform and devices
- **Maintainability:** Bugs associated with software should be able to capture and fix quickly and new tasks and enhancement should be added without trouble.
- **Security:** Data must be protected and secured against unauthorized access. Poor coding and architectural weakness leads to vulnerabilities.
- **Performance:** It refers to an application's use of resources and how that affects its scalability, customer satisfaction, and response time
- **Portability:** Software must be possible to continue using the same basic functions in different situations.

System Development Models

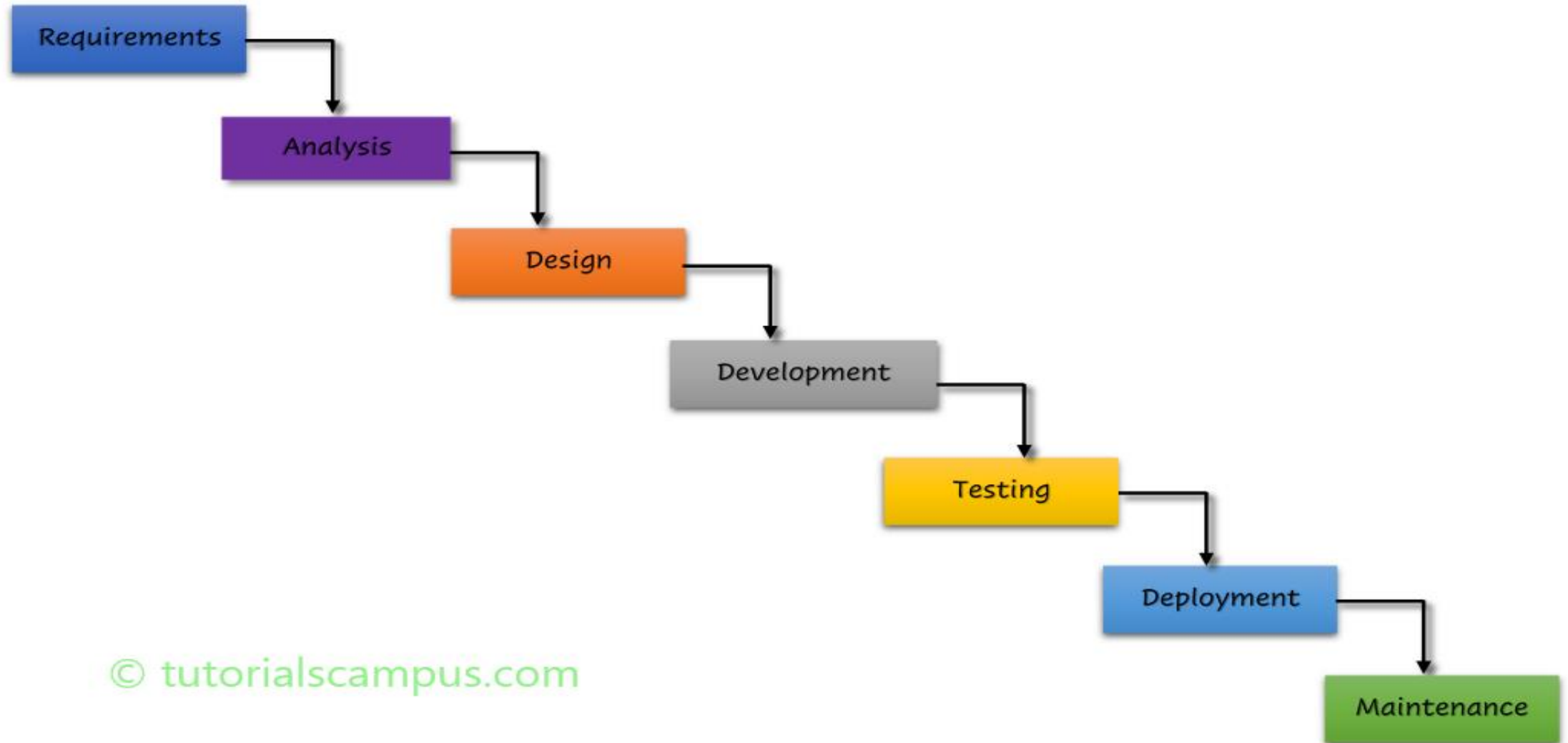
- A system development model is a framework that is used to structure, plan and control the process of developing an information system.
- Depending on the size and purpose of an organization, the system development model used may be different.
- Some of the most commonly used system development models are waterfall model, spiral model, prototype model, agile model etc.

System Development Models

Waterfall Model

- The waterfall model was the first system model.
- It is also referred as a linear-sequential life cycle model.
- It is very simplest and easiest model of SDLC.
- In waterfall model each phase must be completed before the next phase can begin.
- There is no overlapping in the phases.
- In this model, the outcome of one phase acts as the input for the next phase sequentially.

Water Fall Model



System Development Models

Problem definition: All possible requirements of the system to be developed are captured in this phase and documented in a requirement specification document.

Analysis: System analysis and study is completed in this phase.

Feasibility Study: The economic, social feasibility study is accomplished in this phase.

System design: The requirement specifications from first phase are studied in this phase and the system design is prepared. This system design helps in specifying hardware and system requirements and helps in defining the overall system architecture.

System Development Models

Development: System is developed using different development tools and programs.

Testing: The new system is tested for errors and drawbacks.

Maintenance: The system maintenance is accomplished according to the result given by testing.

Implementation: The system is implemented in organization.

System Development Models

.Advantages

- .Simple and easy to understand and use
- .Easy to manage due to the rigidity of the model.
- .Phases are processed and completed once at a time.
- .Works well for smaller projects where requirements are very well understood.
- .Clearly defined stages.
- .Well understood milestones.
- .Easy to arrange tasks.
- .Process and results are well documented.

System Development Models

.Disadvantages

- No working software is produced until late during the life cycle.
- High amounts of risk and uncertainty.
- Not a good model for complex and object-oriented projects.
- Poor model for long and ongoing projects.
- It is difficult to measure progress within stages.
- Cannot accommodate changing requirements.

System Development Models

Prototype Model

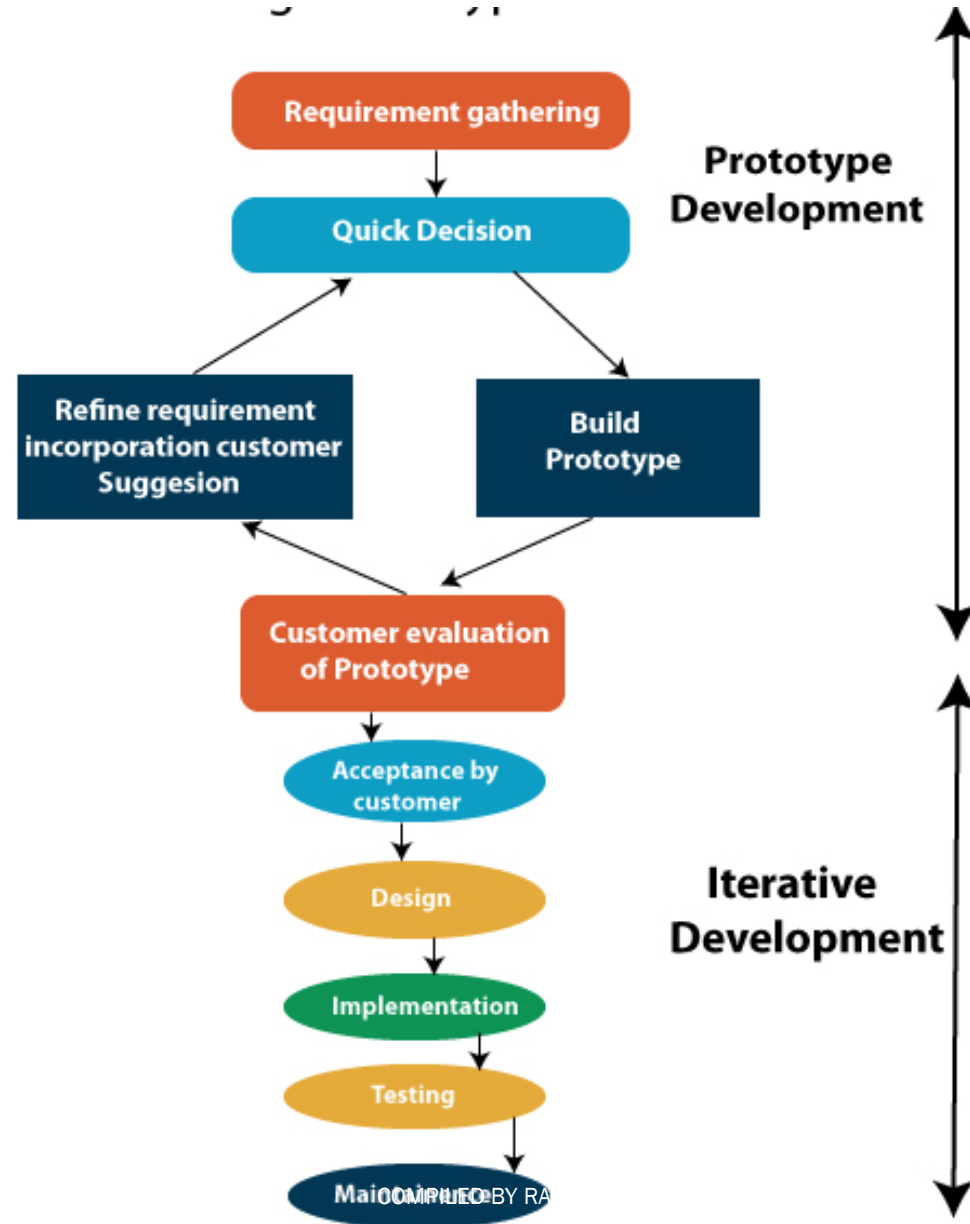
- Designing and developing small sized, similar but functional version of desired system is known as prototyping.
- In this model before designing phase, a prototype is developed, tested, reviewed and approved by the customer.
- After that design will be ready for coding, testing, installation and maintenance.
- This prototype is prepared based on the customer requirements.

System Development Models

Prototype Model

- By using this prototype, customer can understand the requirements of desired system and also the customer can get an “actual feel” of the system.
- It is an attractive idea for complex and bigger systems.
- This Prototype Model is same as waterfall model, but in this model we need to develop prototype and customer interaction will be there.
- Since there is customer interaction there will be less chance of rejection.

Prototype Model



System Development Models

Features of Prototype model

- Whenever the customer is not clear about the requirement in this situation we generally go for prototype model.
- If it is complex project then prototype model makes us clearly understand the requirement.
- Prototyping make sure that the customer constantly work with the system and provide a feedback about the system.

System Development Models

Advantages of Prototype model

- Customer satisfaction exists, because customer can feel the product at very early stage.
- If there is missing functionality, then it can be identified easily
- There will be less chance of software rejection.
- Requirement changes are allowed.
- Due to customer approval we can find the errors at early stage.
- Customer involvement will be there in the development where its leads to better solutions for any confusion / complexity / difficult functions
- The developed prototype can be re-used by developer and test engineer.

System Development Models

Disadvantages of Prototype model

- It is a time consuming if customer ask for changes in prototype.
- This methodology may expand the requirements beyond original plans.
- The invested effort in the preparation of prototypes may be too much if not properly monitored.
- Customer may get confused in between the prototypes and real systems.

System Development Models

Agile Model

- . Agile model refers to a software development approach based on iterative development which breaks tasks into smaller iterations, or parts do not directly involve long term planning.
- . Project scope and requirements are laid down at the beginning of the development process and plans regarding number of iterations, duration and the scope of each iteration are clearly defined in advance

System Development Models

Agile Model

- .Each iteration is considered as a short time "frame" which typically lasts from one to four weeks.
- .The division of projects into smaller parts reduce the overall project delivery time requirements
- .Each and every iteration involves a team working through a full software development life cycle.

System Development Models

Agile Methodology



System Development Models

Phases of Agile Model

Requirements gathering:

- Requirements are defined and business opportunities are explained to plan the time and effort needed to build the project.

Design the requirements:

- We can use high level user flow diagram or high level UML diagram to show the work of new features and show how it will apply to your existing system

System Development Models

Phases of Agile Model

Construction/iteration:

- Designer and developers start working on their project, which aims to deploy a working product. Product will go various stages of improvements

Testing:

- Quality Assurance (QA) team examines the product's performance and looks for the bug

System Development Models

Phases of Agile Model

Deployment:

- The team issues a product for the user's work environment

Feedback:

- After releasing the product, the last step is to receive feedback about the product and work through the feedback.

System Development Models

Advantages of Agile Model

- .Frequent delivery of product
- .Face-to-face communication with clients
- .Efficient design and fulfils the business requirement
- .Anytime changes are acceptable
- .It reduces total development time

System Development Models

Disadvantages of Agile Model

- .Not suitable for handling complex dependencies
- .Depends heavily on customer interaction, if customer is not clear, the project may go in the wrong direction
- .Transfer of technology to new team members may be quite challenging due to lack of documentation

Documentation & its importance

- ▣ Documentation plays very important roles in software development process and it provides the basic guide lines for the modification and enhancement of the software in future
- ▣ In each stage completion, the team members hand over the documentation to the next team member of another stages so that they can communicate well about the system
- ▣ Documentation is the process of communication about the system and it is one of the most important parts of software development.

Documentation & its importance

- ▣ It becomes easy to extend, re-design and debug the system through documentation
- ▣ It is process to help users of system and other people to use and interact with system
- ▣ It refers to keeping records of all project information in the system development process
- ▣ The different types of documentation techniques are printed manuals, user manuals, guides, reference manuals, technical reference guides, installation guides, configuration guides, administration guides, online documentation and help system

Documentation & its importance

▣ **Types of documentations**

▣ **Program manual:**

- ▣ It is written by the system programmer during development process.
- ▣ It is written in the line of source code within programs.
- ▣ It is very useful for the program modification and maintenance of the system.
- ▣ These types of documentations are not visible for the general users

Documentation & its importance

▣ **Types of documentations**

▣ **System manual**

- ▣ System manual is a physical description of a system, device or process.
- ▣ This technical description is used by expert users and designers as guidelines to maintain and modify various elements of the system.
- ▣ These descriptions are intended for experts who must make informed decisions about the installation, capabilities, modifications and applications of the software.

Documentation & its importance

- ▣ **Types of documentations**

- ▣ **User manual**

- ▣ User documentation includes the product guidelines addressed to the general user who needs to know basic requirements for getting the best use of software system.
- ▣ User documentation includes the manuals for software product use.

OOP Vs POP

SN	OOP	POP
6	Problems are viewed as real world entity	Problems are not viewed as real world entity
7	OOP is written by using High Level Language (HLL) such as C++, Java, ASP.NET etc	POP is written by using High & Middle Level Language (HLL & MLL) such as C, Pascal, FORTRAN etc
8	Concept of encapsulation provides high level of security	Security can not be maintained due to lack of such feature
9	Easy to reuse existing program because of inheritance feature	No proper mechanism for reuse of existing code
10	New data and functions can be easily added whenever required	Adding new data and function is difficult and time consuming

Thank You

END OF UNIT 6