

What is System?

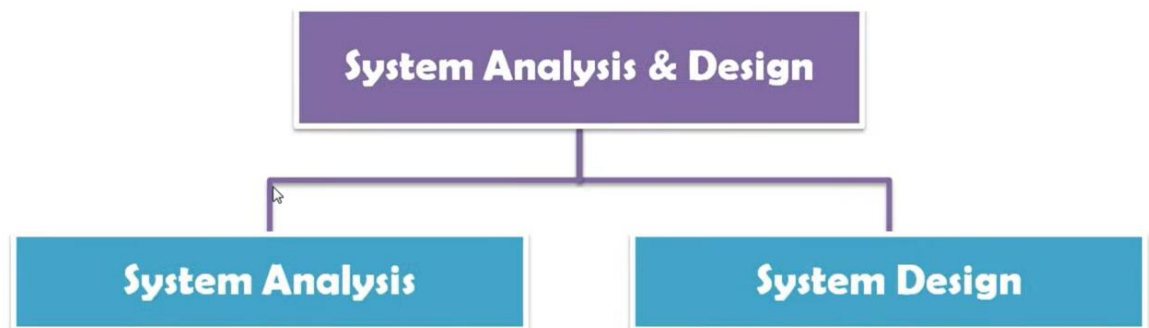
System is a group of elements or components which work together to accomplish a common task.

What is System Analysis & Design?

- In very simple words, system analysis and design is a study in which we learn how to analyse an existing system and create a better one.

Why we need it?

- For System Development
- Creating a new one
- Updating the existing one



System Analysis

- System analysis is a process of studying and observing a system to know how it works and to identify its goal and purposes.
- System analysis is conducted for the purpose of studying a system or its parts in order to identify its objectives
- It specifies “what the system should do”.

System Design

- It is a process of planning a new system or replacing an existing system.
- It is done by defining its components or modules to satisfy the specific requirements.
- It focuses on "how to accomplish the objective of the system".

Properties / Characteristics of a System

A system must have following properties / characteristics:

- Organization
- Integration
- Interaction
- Interdependence
- Central Objective

Organization

- It implies Structure and Order. Elements should be well arranged in order to achieve predetermined objective.

Integration

- It implies how components of a system are tied together.
- The parts of the system must work together within the system even though each part performs a unique function.

Interdependence

- It implies how components of a system depend on one another.
- The components must be well linked and they must coordinate with each other for proper functioning.
- Interdependence is important because the output of one subsystem may be required by other subsystem as input.

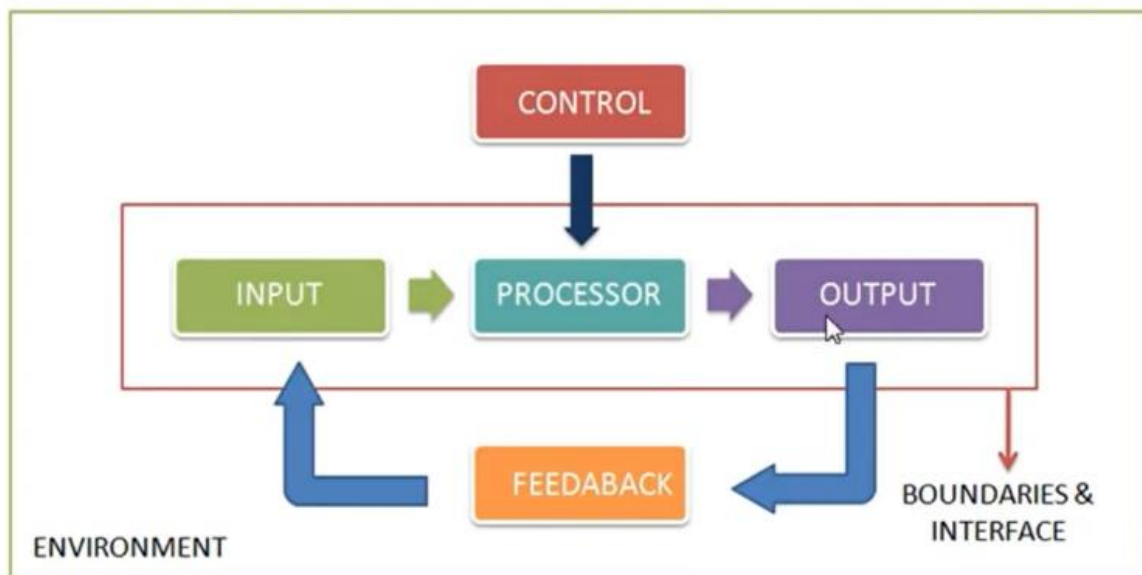
Central Objective

- The objective of the system must be central.
- It means, the user must know the main objective of the system in the early phase for successful design and conversion.

Elements of a System

There are 6 elements in a system:

- Inputs and Outputs
- Processor
- Control
- Feedback
- Boundaries Interface
- Environment



Inputs and Outputs

- The information that enters in a system is treated as Input.
- Output is the outcome of input after processing.
- The main aim of a system is to produce an output which is useful for its users.

Processor

- It transforms the input into useful output.
- It is an operational element.
- It may modify the input either totally or partially, depending on the output specification.

Control

- This element guides the system. It is a decision making element that controls the pattern of activities governing input, processing and output.

Feedback

- The Output is checked with the desired output set and then necessary steps are taken for achieving the output as per the standards.
- Positive feedback encourages the performance of the system.
- Negative feedback is informational in nature.

Boundaries and Interface

- The boundaries are the limits under which the components of a system interact with each other.
- Each system has boundaries that determine its sphere of influence and control.
- The interconnection and the interaction between the sub-systems is known as the Interface.

Environment

- The things outside the boundary of the system are known as environment.
- It is the source of external elements that strike on the system.
- Change in the environment affects the working of the system.

System analyst

- System analyst is a person who uses system analysis and design techniques to solve system/business problems.
- System analyst analyses, design and implements system to fulfil organizational needs.
- System analysts conduct system study, identify requirements and determines the procedures to achieve the system objectives.
- 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.
- In the IT industry, System Analyst figures out how to solve a problem by linking different computers and by specifying what platform, protocols, software, hardware and communications medium can be used to solve a problem.
- System Analyst is responsible for the system from its birth to death.

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 Up gradation
- System Implementation

- System Maintenance

System Analyst is also a...

- **Change Agent:** System Analyst evaluates the current system and as well as researches for new future updates for better System development.
- **Architect:** System Analyst is the architect of the system.
- **Motivator:** System Analyst is a motivator who motivates his team to work properly and do their best.
- **Monitor:** System Analyst monitors the complete system.

Qualities/Attributes of System Analyst

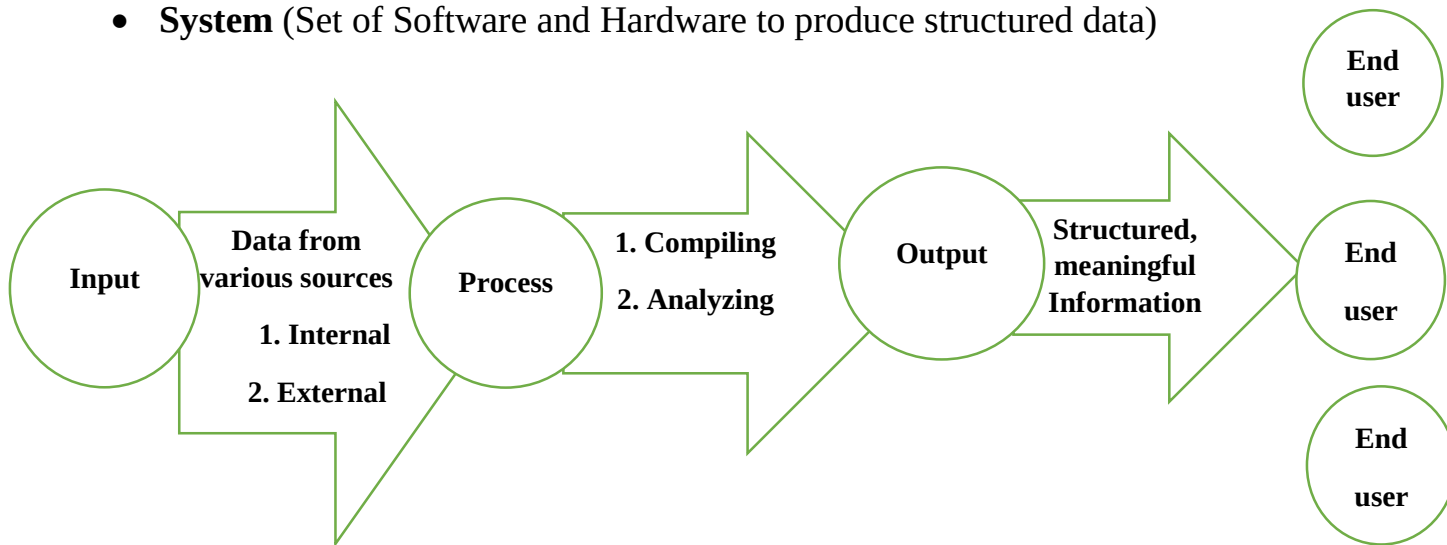
- Knowledge of the Organization
- Knowledge of computer hardware and software
- Problem-solving and critical thinking
- Analytical skills
- Adaptability
- Motivator Skill
- Good understanding, communication and teaching abilities.

Information System (I.S)

An information system is defined as the software that helps to organize and analyze data.

I.S

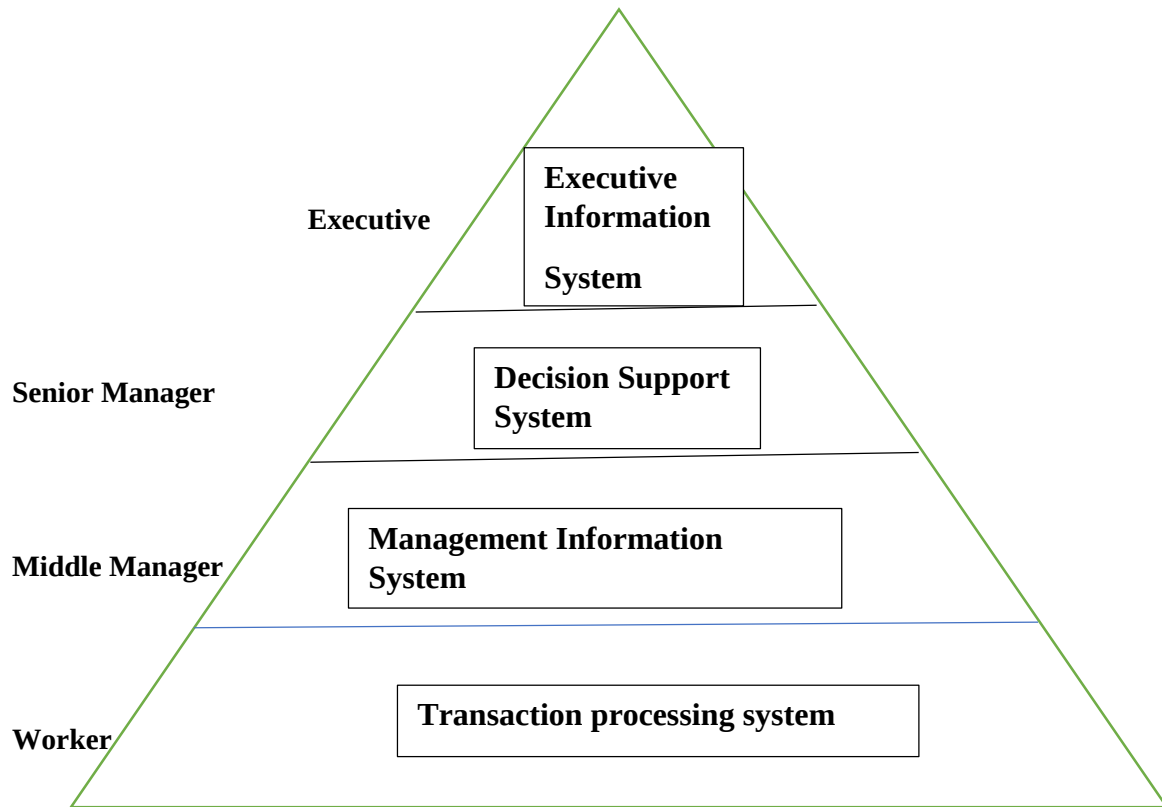
- **Information** (well-structured data)
- **System** (Set of Software and Hardware to produce structured data)



Information system components

System Resources	System Activity
People	Input
Hardware	Storage
Software	Process
Data	Output
Network	Central and maintenance

Types of Information System



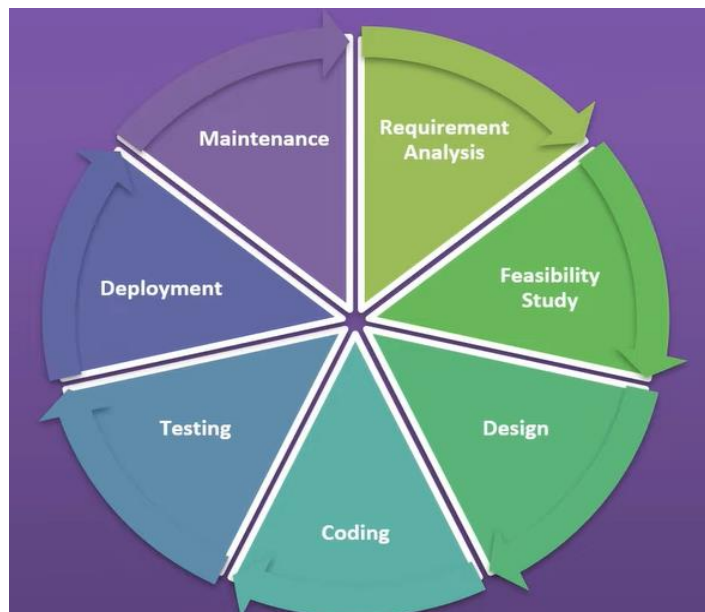
1. **Transaction processing system:** They are directly operated by workers, E.g.- Today sale, today inventory
2. **Management Information System:**
 - It insures the smooth running of organization. E.G- Summary report, Action report.
3. **Decision Support System:** It is knowledge base system used by senior manager. This system is usually interactive and are used to solve problems e.g.- Summary-report, graph, chart
4. **Executive Information System:** It is strategic- level information system that are top of the pyramid. The information at executive level come from both internal and external sources

SDLC

- SDLC stands for "Software Development Life Cycle".
- SDLC is a collection of processes which are followed to develop a software.
- SDLC is a methodology that defines some processes which are followed to develop a high-quality software.
- It covers the detailed plan for building, deploying and maintaining the software.
- The main aim of SDLC is to define all the tasks required for developing and maintaining software.
- It is followed for a software project within a software developing organization.

Phases of SDLC

- ☐ Phase 1: Requirement gathering and analysis
- ☐ Phase 2: Feasibility Study
- ☐ Phase 3: Design
- ☐ Phase 4: Coding
- ☐ Phase 5: Testing
- ☐ Phase 6: Deployment
- ☐ Phase 7: Maintenance



Phase 1: Requirement gathering and analysis

- It is the first phase of SDLC in which all the necessary information is collected from the customer to develop the software as per their expectation.
- Some important questions like: what is the need of software, who will be the end-user, what is the future scope of that software etc. are discussed.
- The main aim of this phase is to collect the details of each requirement of the customer so that the developers will clearly understand what they are developing and how to fulfil the customer's requirements.
- This phase gives a clear picture of what we are going to build.

Phase 2: Feasibility Study

- It is the second phase of SDLC in which an organization discusses about the cost and benefits of the software.
- It is an important phase because profit from the software plays an important role as if cost is very high then company may face loss.
- After the feasibility study, the project may be accepted, accepted with modifications or rejected.
- It measures how much beneficial the product is for the organization.

Phase 3: Design

- It is the third phase in which architects start working on logical designing of the software.
- In this phase a SRS (System Requirement Specification) document is created which contains all logical details

like how the software will look like, which language will be used, database design, modular designs etc.

- This phase provides a prototype of the final product.
- Basically, all it includes is design of everything which has to be coded.

Phase 4: Coding

- When the designing of the software is completed, then, a group of developers starts coding of the design using a programming language.
- The interface of the software and all its internal working according to design phase is implemented in coding phase.
- A number of developers code the modules and then all modules are arranged together to work efficiently.
- It is the longest phase of SDLC.

Phase 5: Testing

- Once the software development is completed, then it is sent to the testers. The testing team starts testing the functionality of the entire system.
- In this phase, the software is checked for bugs or errors.
- Whenever a bug is found, then the software is resent to the coders to fix it and then overall software is re-tested.
- This is done to verify that the entire application works according to the customer requirement.

Phase 6: Deployment

- After overall testing of the software and after checking that it is bug free, then the software is launched and available for the users to use it.
- Even after deployment of the software, if any bugs or errors are still found then the software is re-evaluated by the maintenance team and then it is re-deployed with a new version.

Phase 7: Maintenance

- The maintenance team look over the software usage and user's feedbacks.
- Maintenance is necessary to eliminate errors in the system during its working life and to tune the software.
- The bug fixing, upgrade and enhancement of the software is looked over by the maintenance team.
- Feasibility Study

SDLC Models

- Using the concept of basic SDLC some models were created to ensure success in the process of software development.
- Every developed software is different and it requires a suitable SDLC approach to be followed based on the internal and external factors.

Some most important and popular SDLC Models are:

- Waterfall model
- Iterative model
- Spiral model
- Prototype model
- V model
- Big Bang model

Waterfall Model

- The Waterfall Model was the first SDLC Model to be introduced.
- It is based on the basic concept of SDLC.
- In waterfall model, each phase must be completed before the next phase can begin and there is no overlapping in the phases.

Phases of Waterfall Model

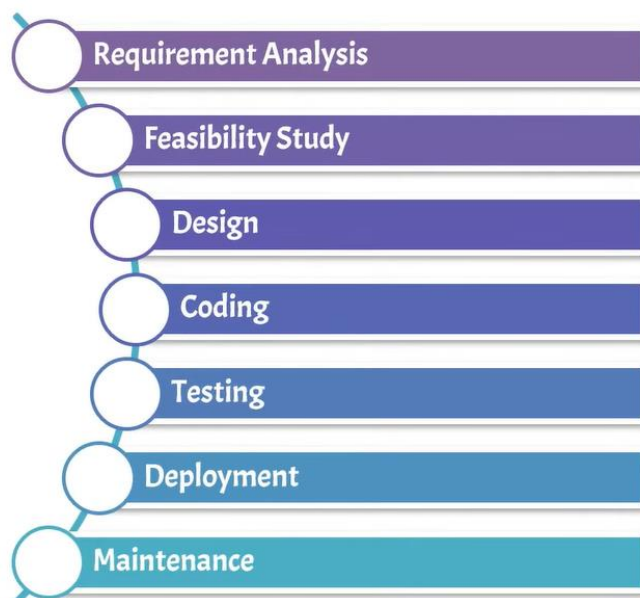
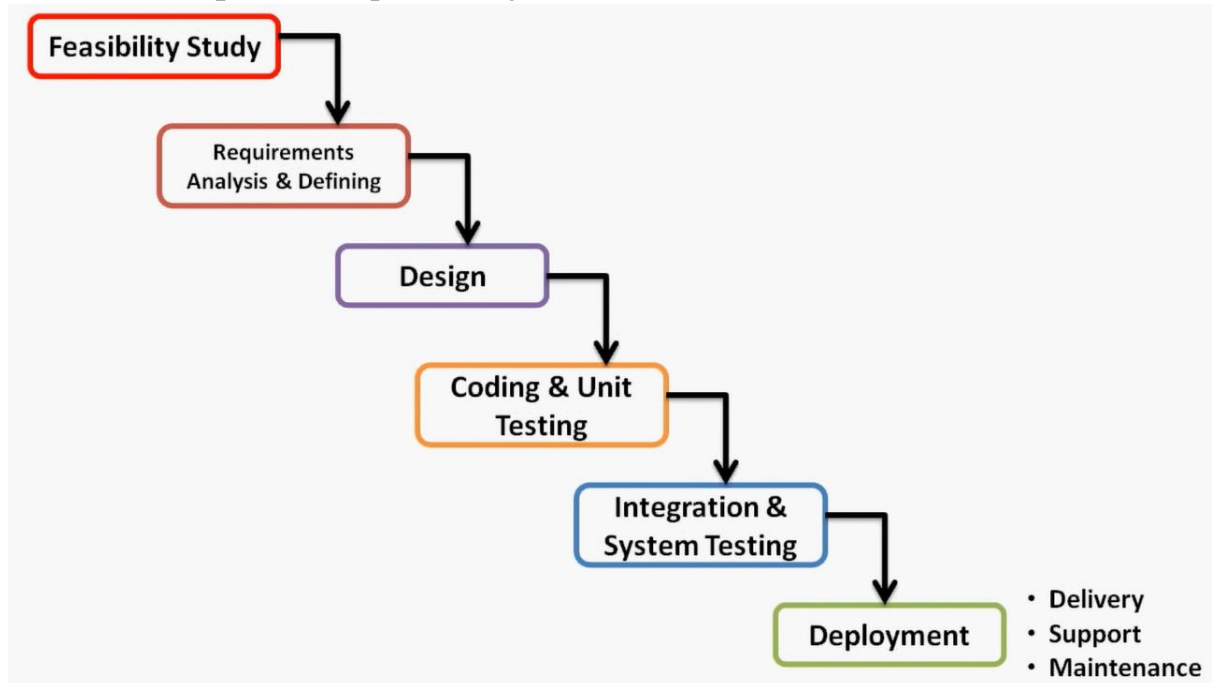
- Requirement Analysis
- Feasibility Study
- Design
- Coding
- Testing
- Deployment

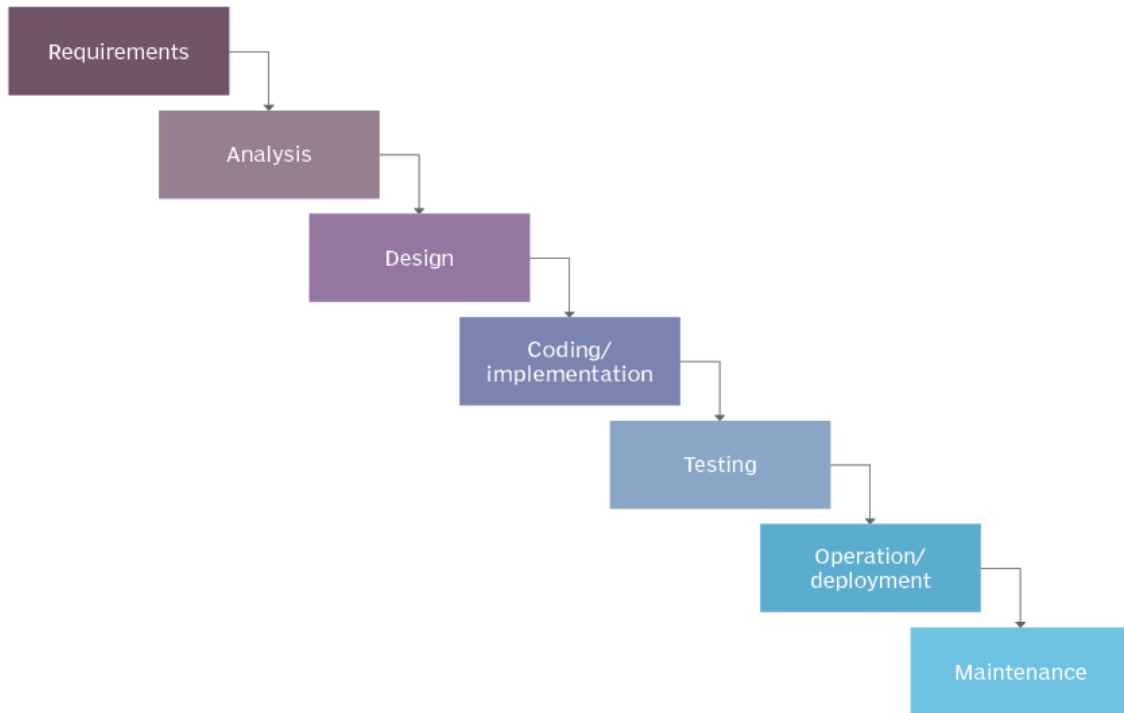
Waterfall Model

- It is earliest SDLC approach that was used for software development
- The waterfall Model illustrates the software development process in a linear sequential flow. It means that every phase in the

development process begins only if the previous phase is completed.

- Also called as classic life cycle model
- It is also known as linear-sequential life cycle model
- In this SDLC model, the outcome of one phase acts as the input for the next phase sequentially.





When to use Waterfall Model

- Requirements are very clear and fixed.
- No ambiguous requirements
- Tools and technology used is consistent and is not changing
- Resources are well prepared and are available to use.
- A project is short
- Risk is minimum

Advantages of Waterfall Model

- Simple and easy to understand and use
- Works well for smaller projects where requirements are well understood and fixed
- Easy to manage due to the rigidity of the model
- Phases are processed and completed one at a time
- Clearly defined stages

- Easy to classify and prioritize tasks
- Process and results are well documented

Disadvantages of Waterfall Model

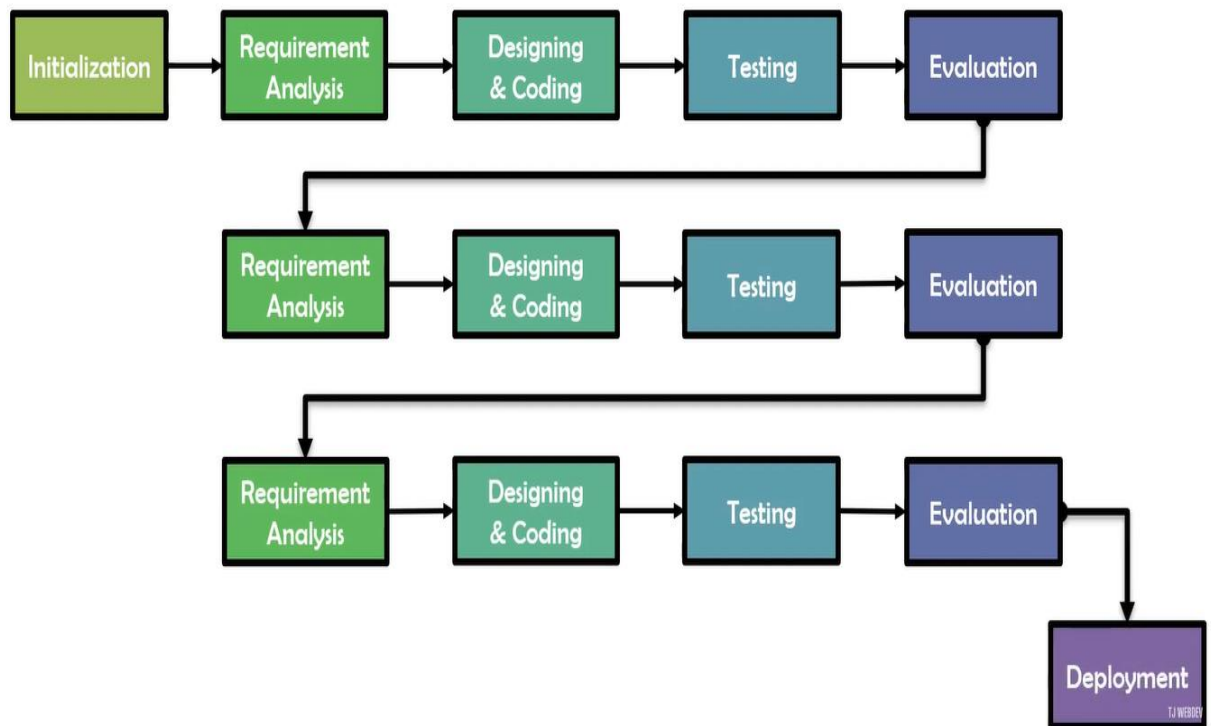
- Not the best choice for complex and large projects
- Not suitable for the projects where requirements are at a moderate to high risk of changing
- Adjusting scope during the life cycle can end a project
- No working software is produced until late during the life cycle.

Iterative Model

- In the Iterative model, the software development is initiated with small sets of requirements and then the software is iteratively enhanced, until the complete system is implemented and ready to be deployed.
- In this model, software development is not attempted to be started with a full specification of requirements.



- Instead, development begins by specifying just some part of the software, which is then reviewed to identify the software for bugs and evolution.
- Enhancing of software is repeated and reviewed by the developers and hence results in producing a new version of that software at the end of each iteration of the model.



When to use Iterative Model

- Major requirements of the project are clearly defined
- Some features or enhancements may be added with time
- The project is large
- Resources with needed skill sets are not available and are planned to be used on contract basis
- High-risk features and goals which may change in the future
- A new technology is being used and is being learnt by the development team while working on the project

Advantages of Iterative Model

- Bugs and errors in the software can be identified early.
- Software is produced early which facilitates customer evaluation and feedback
- It is easily adaptable to the ever changing needs of the client
- The end-user can give their feedback quickly
- Parallel development can be planned
- Risks are identified and resolved during iteration

- Less time is spent on documentation and more time is given for development
- Progress can be measured.

Disadvantages of Iterative Model

- Requirements changes, again and again, can cause over budget and over time
- Not suitable for smaller projects
- Project completion date prediction is a problem because the software is changing again and again in iterations
- More resources may be required
- More management attention is required
- Design can be changed again and again because of imperfect requirements

Prototype Model

- Prototype model is a software development model in which a prototype (i.e., an early sample of the final product) is built and evaluated. And then rework is done on it as per client requirements, until a final acceptable product is achieved.
- A prototype is a working model of software with some limited functionality.
- A prototype does not always hold the exact logic used in the actual software application.
- It helps to get valuable feedback from the customer and helps software designers and developers to understand about what exactly is expected from the product under development.

Types of Prototyping Models

1. Rapid Throwaway Prototype

- Rapid throwaway is based on the preliminary requirement. It is quickly developed to show how the requirement will

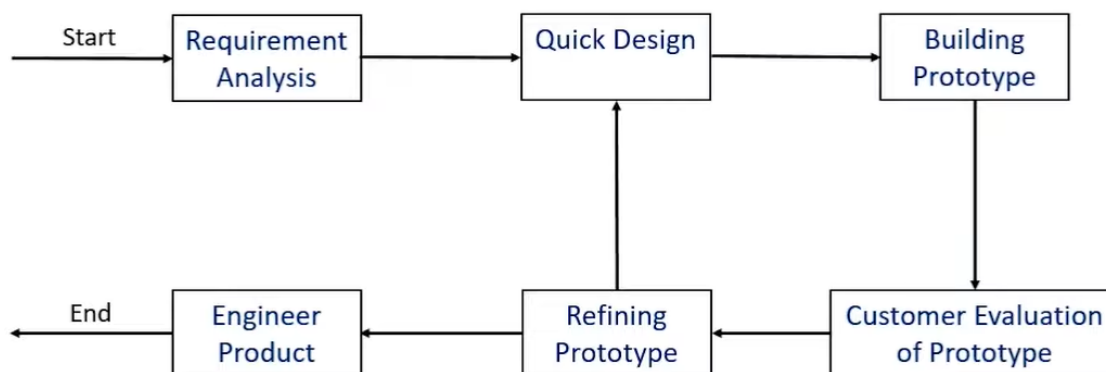
look visually. The customer's feedback helps drives changes to the requirement, and the prototype is again created until the requirement is satisfied.

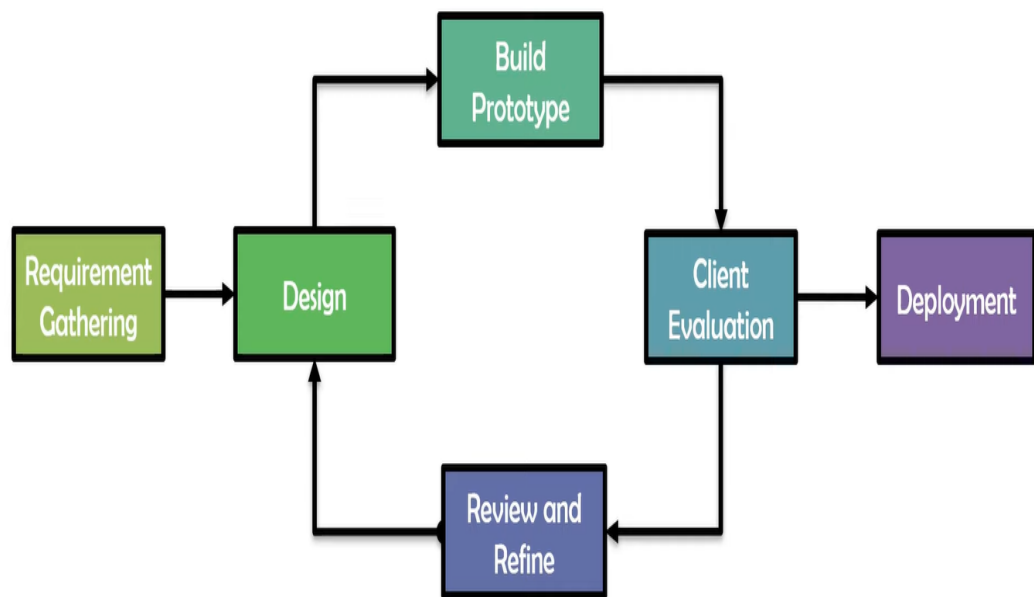
- In this method, a developed prototype will be discarded and will not be a part of the ultimately accepted prototype. **This technique is useful for exploring ideas and getting instant feedback for customer requirements.**

2. Evolutionary Prototyping

- Here, the prototype developed is incrementally refined based on customer's feedback until it is finally accepted. It helps you to save time as well as effort. That's because developing a prototype from scratch for every interaction of the process can sometimes be very frustrating.
- This model is helpful for a project which uses a new technology that is not well understood. It is also used for a complex project where every functionality must be checked once. It is helpful when the requirement is not stable or not understood clearly at the initial stage.

Steps of Prototype Model





When to use Prototype Model

- Prototype model should be used when the desired system needs to have a lot of interaction with the end users.
- When requirements are not clear
- Mid - Large project
- Client can communicate without delays

Advantages of Prototype Model

- Client feedback is received quickly which speeds up the development process
- Users are actively involved in the development
- More accurate user requirements are obtained.
- Quicker customer feedback provides a better idea of customer needs
- Developed prototypes can be used later for any similar projects
- Missing functionalities and errors can be detected early
- Software designers and developers understand about what exactly is expected from the product

Disadvantages of Prototype Model

- Prototyping may be a slower and time taking process
- This model is time consuming and expensive
- After seeing an early prototype the users may demand the actual system to be delivered in soon.
- Risky for fresh developers
- Poor documentation due to changes in the requirements
- Regular meetings are vital to keep the project on time
- If end-user not satisfied with initial prototype, he/she may lose interest in the project.

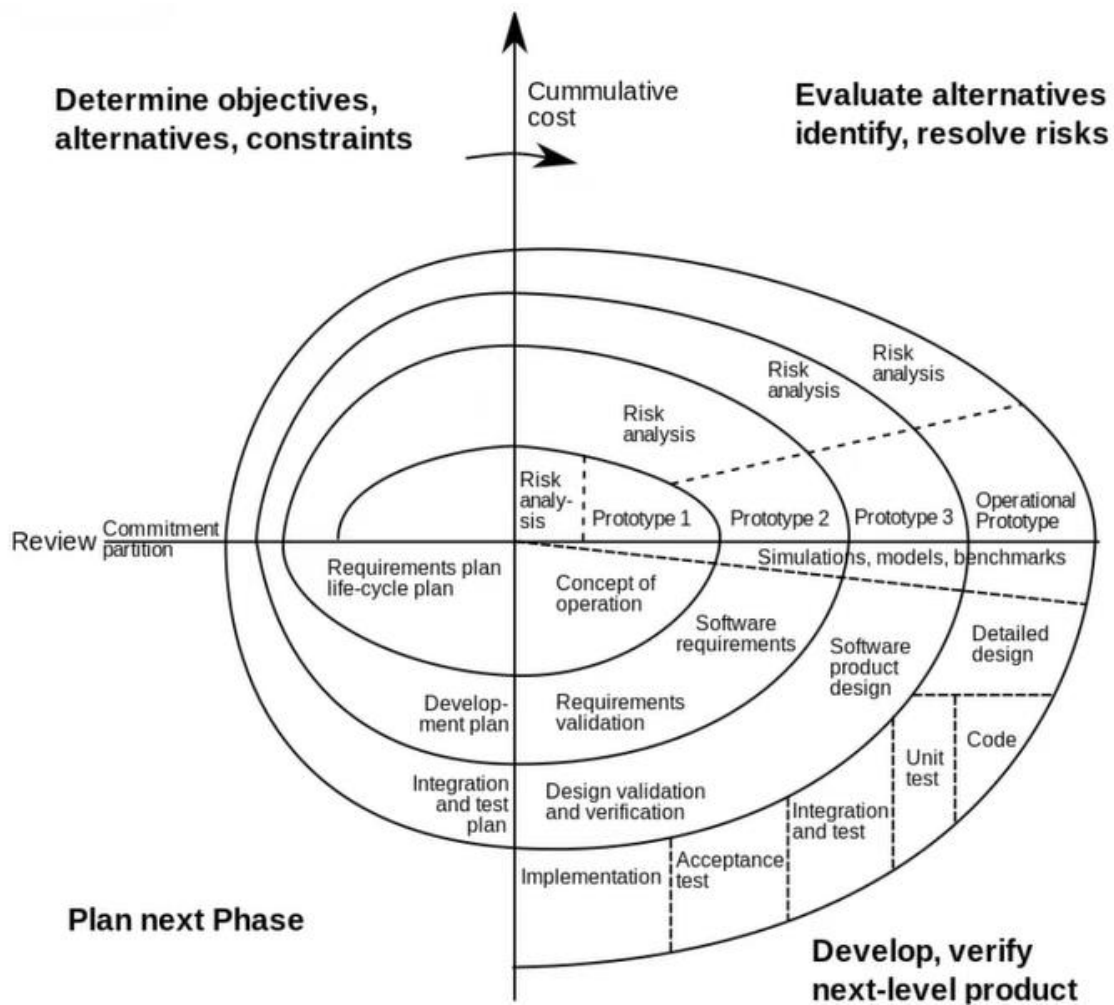
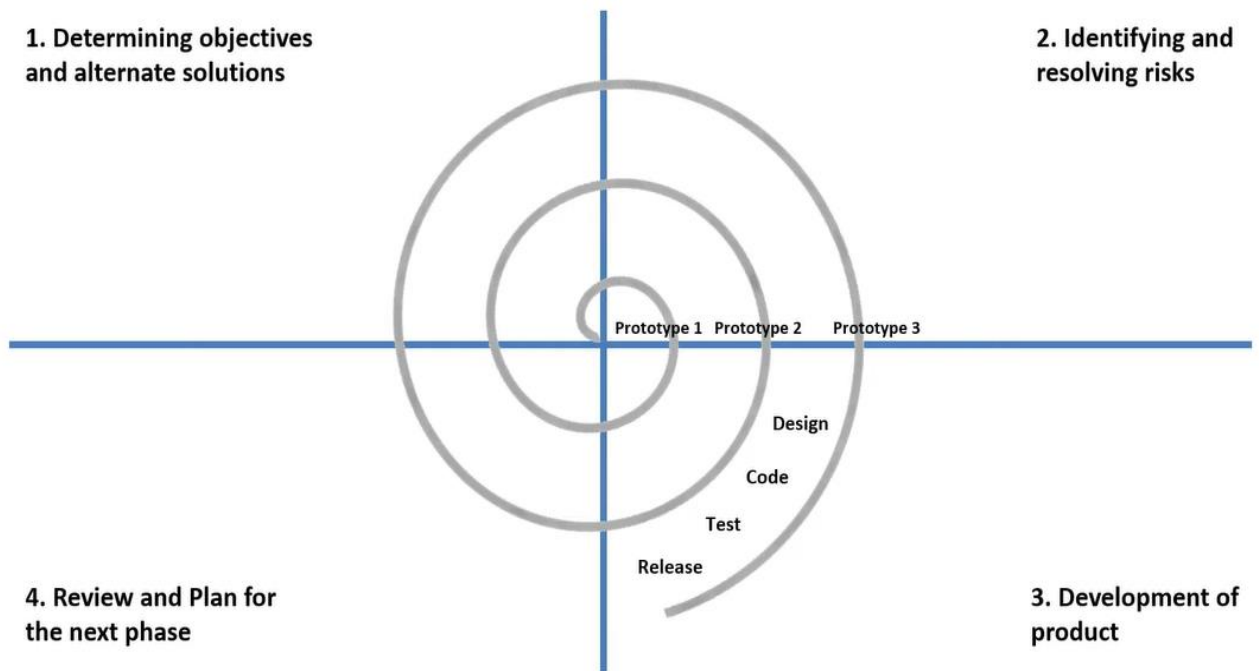
Spiral Model

- Spiral model is a risk handling software development model.
- It was first introduced by Barry Boehm in his 1988 article
- It is a combination of iterative nature of prototyping with the controlled and systematic aspects of waterfall model.
- There are some phases that are completed in one iteration (spiral) and the output is a small prototype of the large software. Then the same iteration is repeated until the entire software is built.

Phases of Spiral Model

The entire process of development is divided into four different stages which keep on repeating until the entire project is completed. These stages are

1. Determining objectives and alternate solutions and constraints.
2. Evaluates alternatives, Identifying and resolving risks
3. Development and verify next level - product.
4. Review and Plan for the next phase



- The Spiral model is also called "Meta-Model" as it includes all the other SDLC models.
- It is a risk-driven model, i.e., the overall success of a project highly depends on the risk analysis phase.

1. Determining objectives and alternate solutions and constraints:

Every software development starts with planning. The first quadrant starts with the feasibility study and requirement gathering and analysis.

In this phase, we determine the objective of the project means we determine the goal of the project. Requirements are gathered from the customers through interviews and discussions, analyses of the requirements, and all the requirements convert into documentation in the form of SRS. We also find alternative solutions means finding the possible ways to achieve our objective.

2. Evaluates alternatives, Identifying and resolving risks:

In the second quadrant, all the requirements gathered in the first quadrant are studied and all the alternative solutions are evaluated and the risks associated with the alternative solutions are identified, focus on high-risk features, analyze the risks and try to minimize the risks by using the best possible strategy. In the end, after finding the best strategy a prototype product is built which helps to minimize the risks before developing the actual product.

3. Development and verify next level – product:

In this third quadrant, all the required features are developed and verified through testing methodologies. At the end of this phase, a new software product or the next version of the product is ready to deliver to the customer.

This phase includes actual software development, testing the software, and deploying the software product to the customer so the customer can use the product and gives us feedback, which helps in reviewing and planning the next phase of the product.

4. Reviewing and planning for the next phase:

In the fourth quadrant, the customer evaluates the software product and gives us feedback. After the feedback, the software product is reviewed where the software engineers make a choice according to the feedback whether to continue with a further period of the spiral. If any further changes are required, the project continues to the next phase.

The next spiral depends on customer feedback, remaining risks, and project completion.

When to use the spiral model?

- This model is mostly used in larger projects. Which involve continuous enhancements.
- When the high risk is involved in developing the software product.
- When requirements are a bit complicated and users are unclear of their needs.
- When cost and continuous risk evaluation are important.
- When a product needs to be released frequently.
- When significant changes are expected.
- When the prototype is needed to avoid the risks.
- This model is used where the requirements are of developing a full-fledged, large, complex system with a lot of features and facilities from scratch.
- It may use when experimenting on technology, trying out new skills, or where the user is not able to offer requirements clearly.

Advantages of Spiral model

- This model is useful to avoid risks.
- Critical high-risk functions are developed first.
- Early and frequent customer feedback.
- Provides an early indication of risks, without much cost.
- This model is very useful for larger and high-risk projects.
- Additional features can be added at the next spiral (phase).
- This model is very flexible compared to other models
- Product is delivered on every spiral so the end-users get a chance to see the system early.
- Use prototype on every spiral. As a result users see the system early

- Bulky and complex system can be made easily because of the risk management factor.
- Changing requirements can be accommodated.

Disadvantages of Spiral Model

- Management is more complex
- End of the project may not be known early
- Not suitable for small or low risk projects and could be expensive for small projects
- The successful completion of the project is very much dependent on Risk Analysis
- Time estimation is very difficult
- This model is not budget-friendly.
- Required some expert people, who can analyze the risks.
- More time is required for planning, risk evaluation, etc.

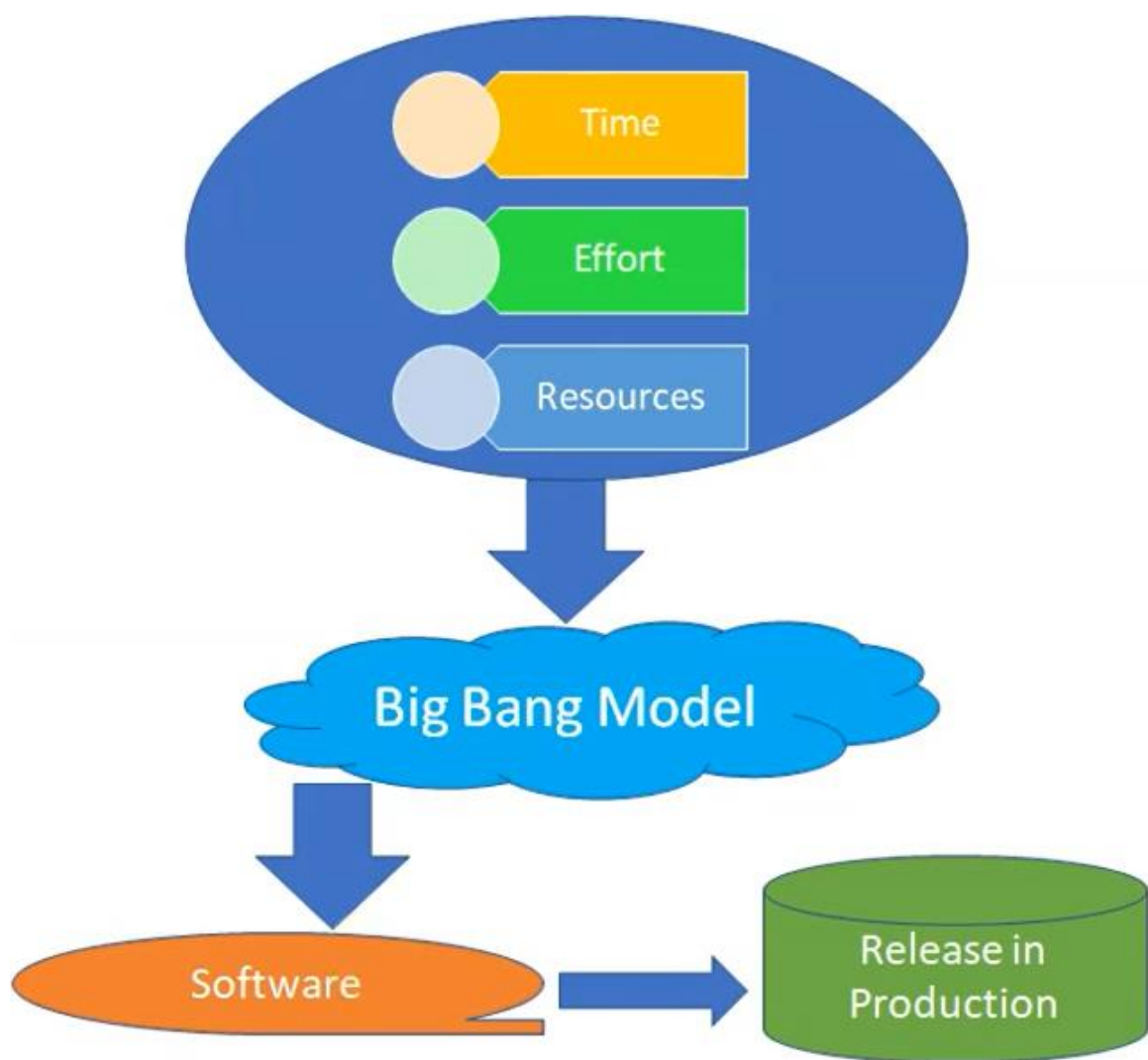
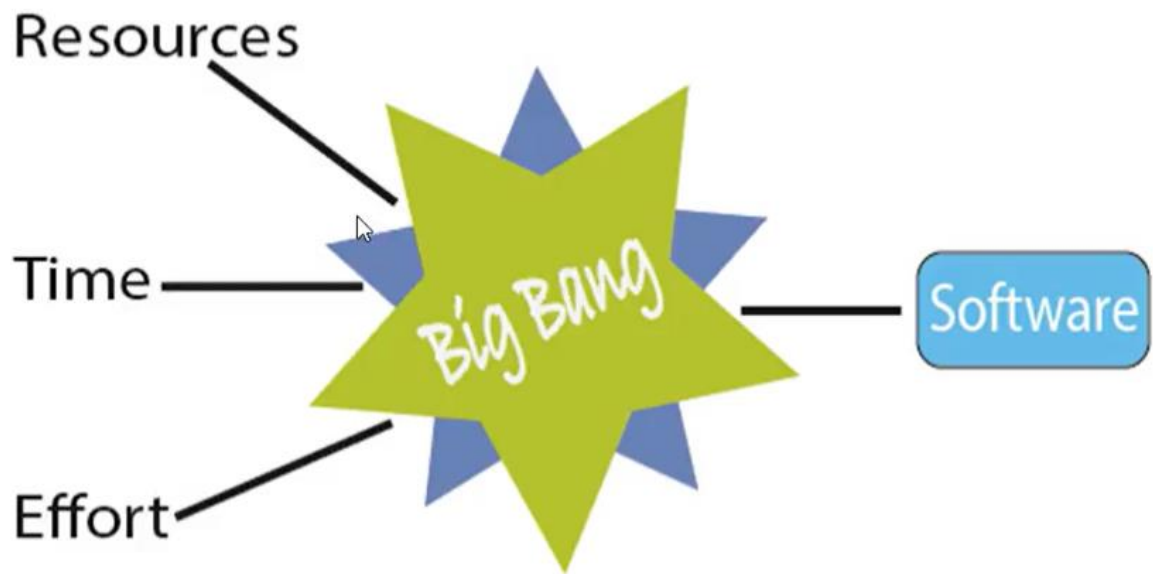
Big Bang Model

Big Bang Theory...

According to a theory known as big bang theory the existence of this world and other stars and all those things found in the space is the result of a big blast.

The name of **Big bang model** in software engineering is inspiration from the Big Bang theory.

- Ideal for small short project where only one or two programmers are working to create a small project with less requirements or some non-final kind of requirements.
- Best for the students who are practicing programming and developing the software so they can directly focus on the development instead of going through lots of Software Engineering.
- This model has no certain techniques to work on a software development project. This model may start with a small and basic amount of money and resources to start with.
- This model requires the least planning. It does not follow any formal method. This model is adopted when the customer is not sure about his needs and requirements. The requirements are not analyzed as well.
- In this model, the entire focus remains on writing the code and delivering the final project. The requirements are understood and implemented as they come. As no planning and analysis take place that is why this model is riskier than others.



Advantages of Big Bang Model

- There is no planning required.
- Simple Model.
- Few resources required.
- Easy to manage.
- Flexible for developers.

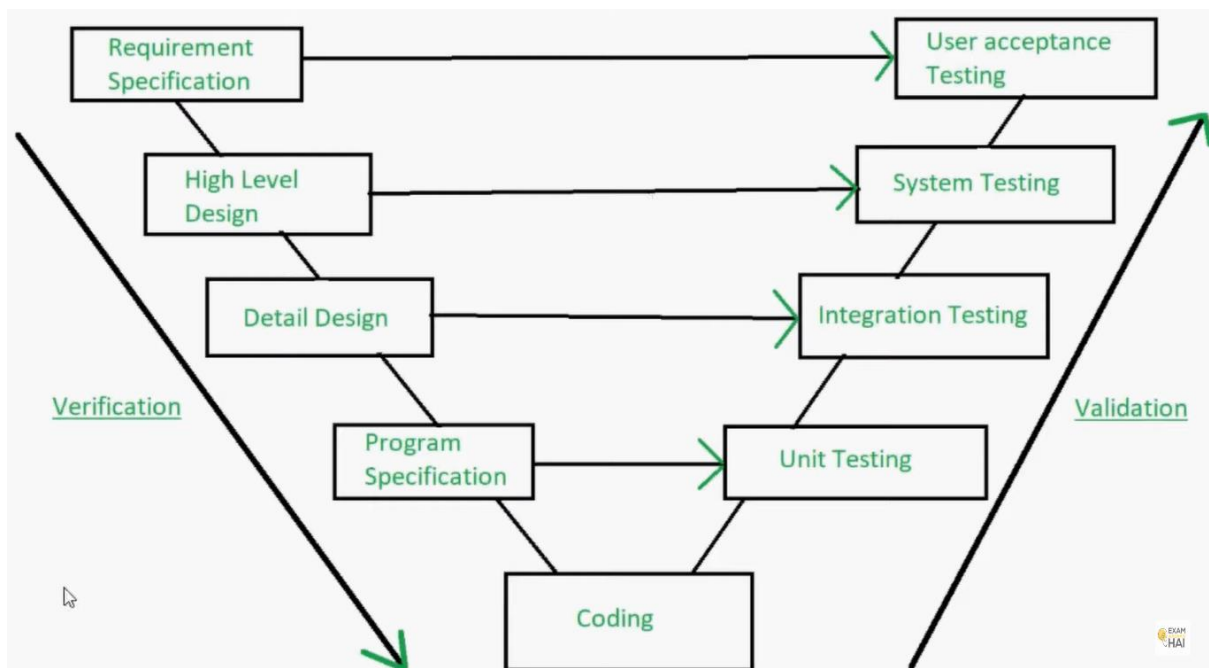
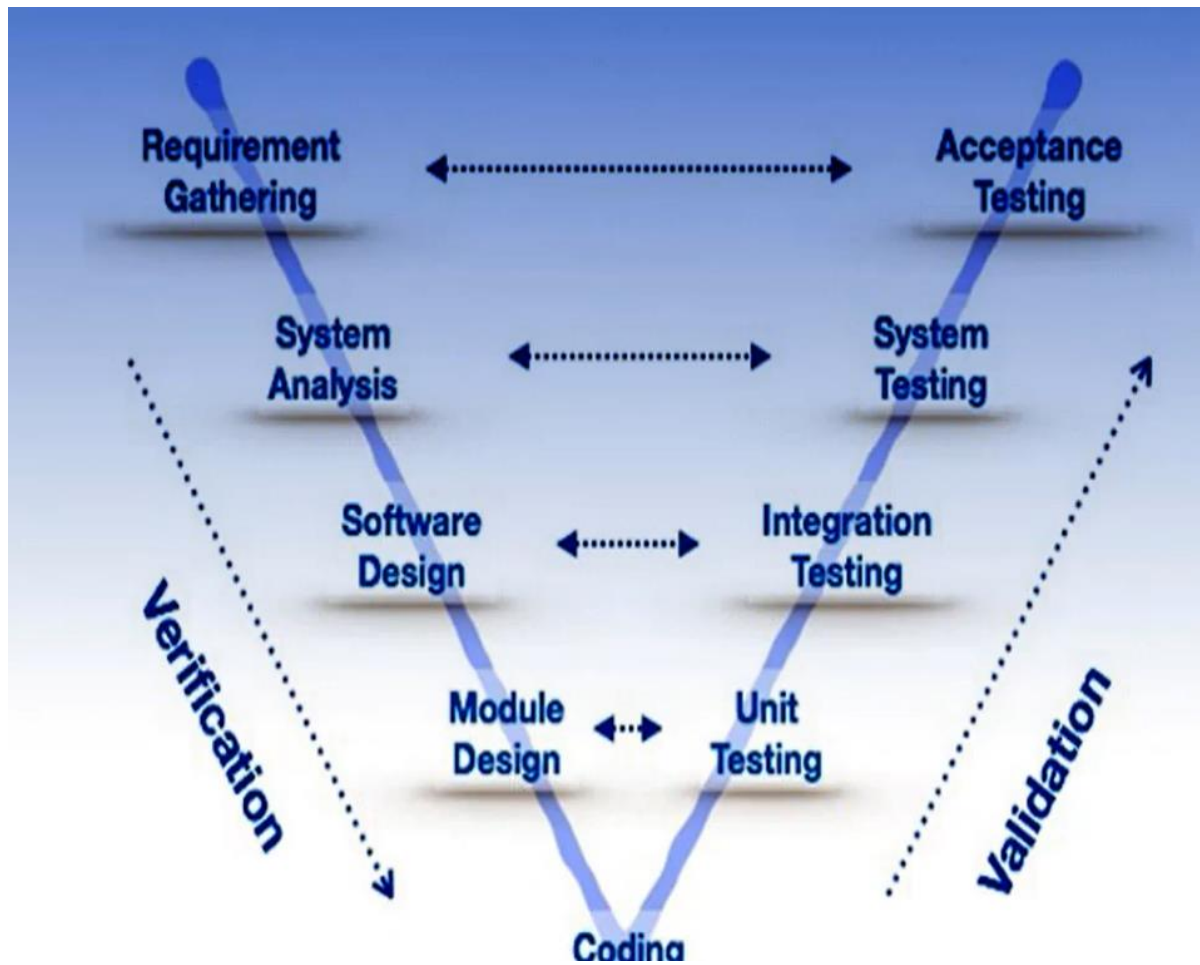
Disadvantages of Big Bang Model

- It is not suitable for larger and complex projects
- Very poor model for long and ongoing projects
- Risk is very high and uncertainty.

V- Model

- The V- Model of software engineering is also known as Verification and validation model.
- IT is a type of SDLC model where its phases executes in V Shape.
- The major plus point of this V- Model is that it over comes a drawback of waterfall model in which you cannot go back to the previous stage for correction.

- IT also allow you to test the software at each stage in reverse manner



When to use V- Model

- Where requirements are clearly defined and fixed.
- The V-Model is used when ample technical resources are available with technical expertise.

Advantages of V- Model:

- This is a highly disciplined model and Phases are completed one at a time.
- V-Model is used for small projects where project requirements are clear.
- Simple and easy to understand and use.
- This model focuses on verification and validation activities early in the life cycle thereby enhancing the probability of building an error-free and good quality product.
- It enables project management to track progress accurately.

Disadvantages of V- Model:

- High risk and uncertainty.
- It is not a good for complex and object-oriented projects.
- It is not suitable for projects where requirements are not clear and contains high risk of changing.
- This model does not support iteration of phases.
- It does not easily handle concurrent events.

What is verification and Validation?

Verification:

It is the process of checking whether the software meets the specifications.

Validation:

The Process of checking whether the specifications meet the customer needs.

- V-Model allow you to verify and validate in parallel

Confusing Sentence.....

Verification: Are we building the system right?

Validation: Are we building the right system?

Verification:

- The Process of evaluating the product during the development is called verification.

Validation:

- The Process of checking the product after the development is called validation.

Verification	Validation
It includes checking documents, design, codes and programs.	It includes testing and validating the actual product.
Methods used in verification are reviews, walkthroughs, inspections and desk-checking.	Methods used in validation are Black Box Testing, White Box Testing and non-functional testing.
It checks whether the software confirms to specifications or not.	It checks whether the software meets the requirements and expectations of a customer or not.
It can find the bugs in the early stage of the development.	It can only find the bugs that could not be found by the verification process.
The goal of verification is application and software architecture and specification.	The goal of validation is an actual product.
Quality assurance team does verification.	Validation is executed on software code with the help of testing team.
It comes before validation.	It comes after verification.
It consists of checking of documents/files and is performed by human.	It consists of execution of program and is performed by computer.

Verification	Validation
Verification is for prevention of errors.	Validation is for detection of errors.
Verification finds about 50 to 60% of the defects.	Validation finds about 20 to 30% of the defects.
Verification is about process, standard and guideline.	Validation is about the product.

How to perform verification

- This is all human based job.
- Inspection & review kind of things are used to perform the verification.
- It is performed by the quality assurance team most of the time.

How to perform validation

- This is human and computer based work
- Testing i.e. Black Box, White Box fall into this category
- Usually performed by the testers
- Customers can also validate the product

Which one is better

Both are good at their own places.

If we catch the bug in verification phase it will be easy to correct and will not cost too much as compared to validation.

CASE Tool

- Stands for Computer-Aided Software Engineering.
- These are special kind of software which provide support, ease in different phases of Software Development life cycle.

Categories of CASE Tools

Upper case

- Used for planning analysis & designing.

Lower case

- Used for implementation testing and maintenance..

Integrated case

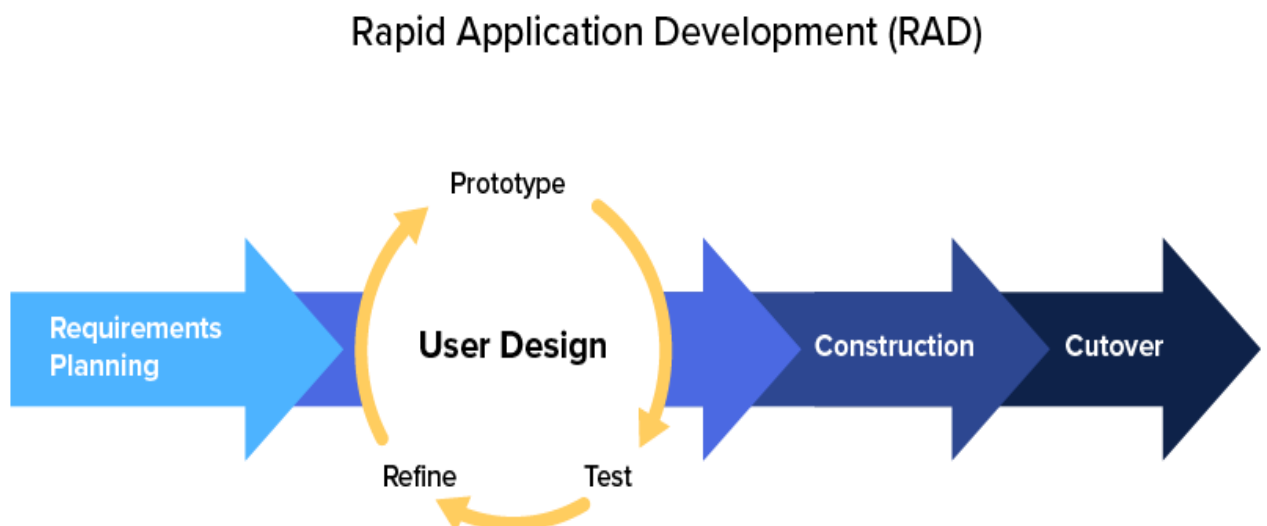
- Used to combine the elements of upper case and lower case tools into a project

Some examples of CASE tools

- Tools which are used to draw Engineering diagrams
- Editor of high level languages which provide support to design User interface.
- In web pages development we have the ability to watch the live preview in different devices
- Test case generation tools.

RAD - Rapid application development

- RAD model heavily based on prototyping and iterative development and no detail preplanning is involve in this model.
- This model was first proposed by IBM in 1980's
- The software project which we can brake into modules can use this RAD model.
- Development of each module requires basics SDLC steps like waterfall steps.



Steps in RAD

Define the Requirements: At this step you don't need to collect the requirements like the other development model the broader nature of requirement is enough that will help you to get the specific kind of requirements.

User Description: This is where prototype development take place on the basis of user feedback, and this step validate the data collected in the 1st step.

Constructions: At this stage we refine the prototype and make it ready to deliver. All kind of required modifications and enhancements are made in this phase.

Cutover: The last stage where all the independent developed modules are move to the live production, here any full scale testing and team training is performed if required.

When to use RAD model:

- RAD should be used when there is a need to create a system that can be modularized in 2-3 months of time.
- It should be used if there's high availability of designers for modelling and the budget is high enough to afford their cost along with the cost of automated code generating tools.
- RAD SDLC model should be chosen only if resources with high business knowledge are available and there is a need to produce the system in a short span of time (2-3 months).

Advantages

- Reduce development Time.
- Encourages customer feedback.
- Quick initial reviews.
- Easy to accommodate changing the requirements.

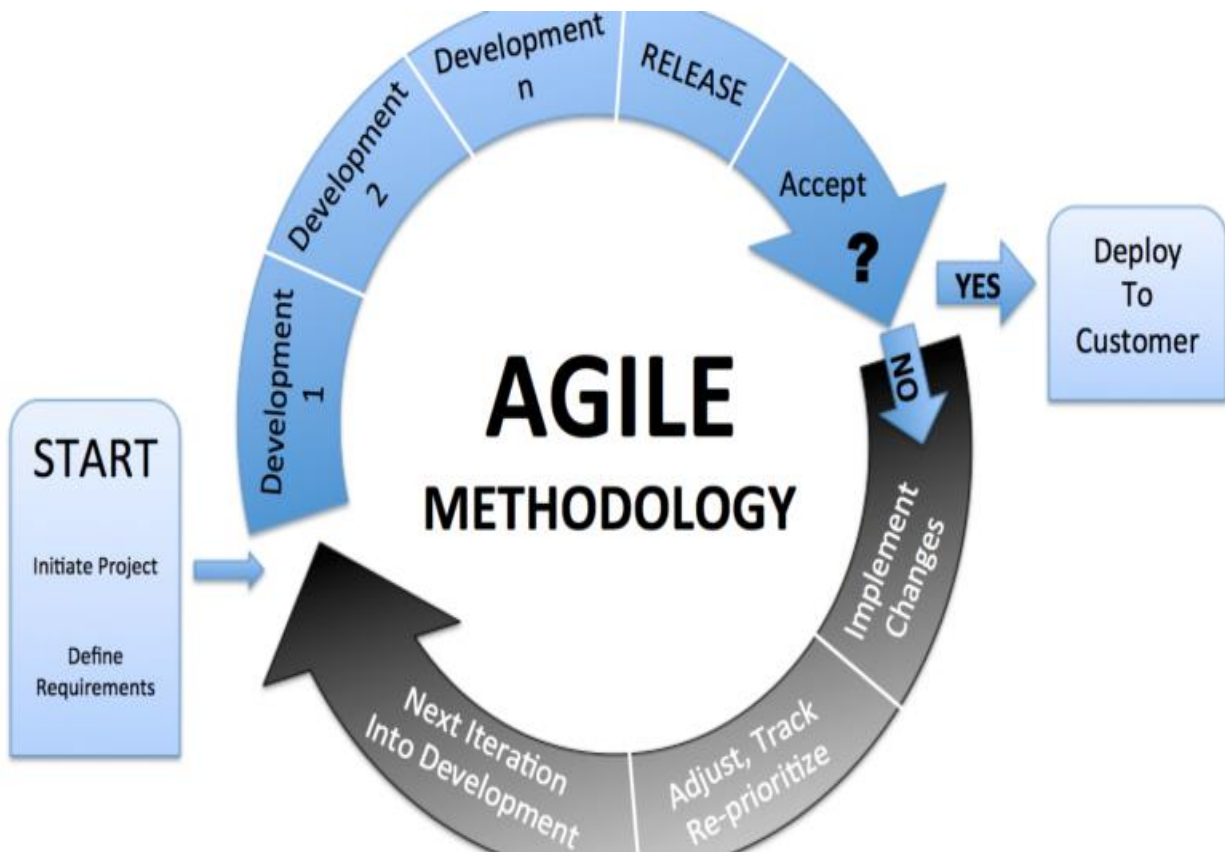
Disadvantages

- Need of highly skilled team
- Customer involvement is required throughout the RAD model
- Implementable to only those Projects which can be modularized.
- Depends on strong team and individual performances for identifying business requirements.
- Only system that can be modularized can be built using RAD
- Requires highly skilled developers/designers.
- High dependency on modelling skills
- Inapplicable to cheaper projects as cost of modelling and automated code generation is very high.

What is Agile?

- Agile is an umbrella, and lots of different types fall under this umbrella
- Different type like Scrum, XP, lean, kanban and many more.
- Some people think that Agile itself is a development approach.

- But No, agile is not development approach agile is a bag which contains the lots development approaches that follow the same core theme.
- The core theme is iterative development, Faster Delivery, Increased quality.
- Agile methodologies means the different approaches of agile.
- Agile is a bag in which we have different methods of development.
- So those methods or those things which are in bag (agile) are methodologies.
- Agile means "Able to move your body".
- In Software Engineering the term Agile means able to Adopt the changes from requirements or technology based .(Ref: Cambridge dictionary online)
- Agile development model is also a type of Incremental model. Software is developed in incremental, rapid cycles.
- This results in small incremental releases with each release building on previous functionality.
- Each release is thoroughly tested to ensure software quality is maintained. It is used for time critical applications.
- Extreme Programming (XP) is currently one of the most well-known agile development life cycle model



When to use agile model:

- When new changes are needed to be implemented. The freedom agile gives to change is very important. New changes can be implemented at very little cost because of the frequency of new increments that are produced.
- To implement a new feature the developers need to lose only the work of a few days, or even only hours, to roll back and implement it.

Advantages of Agile model:

- Customer satisfaction by rapid, continuous delivery of useful software.
- People and interactions are emphasized rather than process and tools. Customers, developers and testers constantly interact with each other.
- Working software is delivered frequently (weeks rather than months).
- Face-to-face conversation is the best form of communication.
- Close, daily cooperation between business people and developers.
- Continuous attention to technical excellence and good design.
- Regular adaptation to changing circumstances.
- Even late changes in requirements are welcomed

Disadvantages of Agile model:

- In case of some software deliverables, especially the large ones, it is difficult to assess the effort required at the beginning of the software development life cycle.
- There is lack of emphasis on necessary designing and documentation.
- The project can easily get taken off track if the customer representative is not clear what final outcome that they want.
- Only senior programmers are capable of taking the kind of decisions required during the development process. Hence it has no place for newbie programmers, unless combined with experienced resources.

XP

(Extreme Programming)

- It is the type of Agile Development we develop software with small team where the software requirements changes rapidly.

Values

The five values of XP are communication, simplicity, feedback, courage, and respect and are described in more detail below.

- **Communication**

Software development is inherently a team sport that relies on communication to transfer knowledge from one team member to everyone else on the team. XP stresses the importance of the appropriate kind of communication – face to face discussion with the aid of a white board or other drawing mechanism.

- **Simplicity**

Simplicity means “what is the simplest thing that will work?” The purpose of this is to avoid waste and do only absolutely necessary things such as keep the design of the system as simple as possible so that it is easier to maintain, support, and revise. Simplicity also means address only the requirements that you know about; don’t try to predict the future.

- **Feedback**

Through constant feedback about their previous efforts, teams can identify areas for improvement and revise their practices. Feedback also supports simple design. Your team builds something, gathers feedback on your design and

implementation, and then adjust your product going forward.

➤ **Courage**

This shows a preference for action based on other principles so that the results aren't harmful to the team. You need courage to raise organizational issues that reduce your team's effectiveness. You need courage to stop doing something that doesn't work and try something else. You need courage to accept and act on feedback, even when it's difficult to accept.

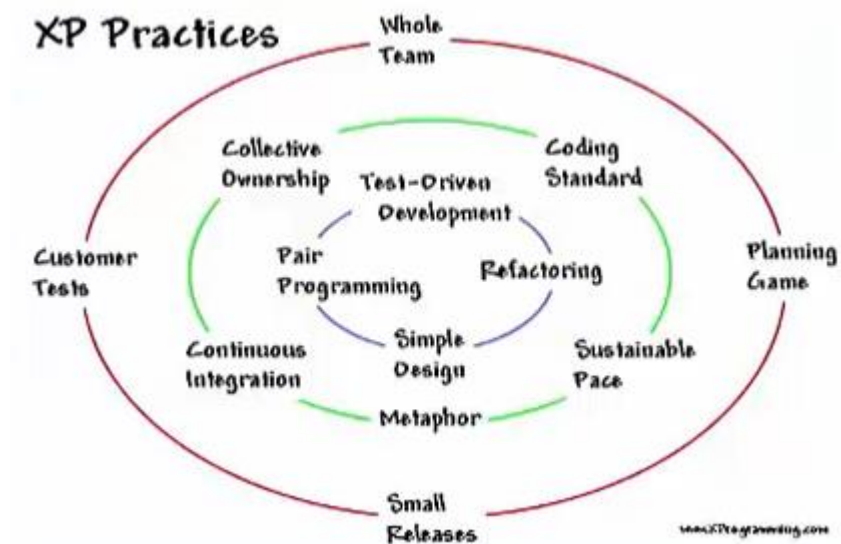
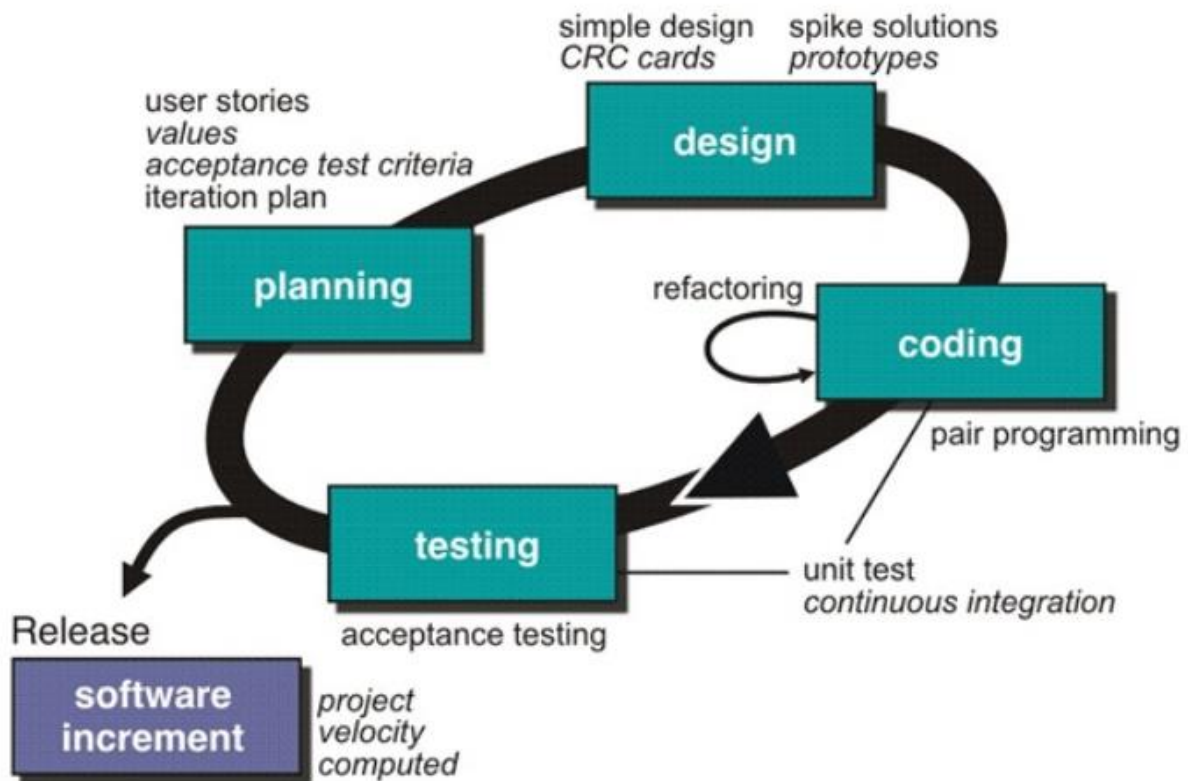
➤ **Respect**

The members of your team need to respect each other in order to communicate with each other, provide and accept feedback that honors your relationship, and to work together to identify simple designs and solutions.

➤ XP is summed up or based on twelve Practices.

1. The Planning Process
2. Small releases
3. Metaphor(resembles)
4. Simple design
5. Testing
6. Refactoring(change in design according to requirement change)
7. Pair programming (two programmer work on one code) like giving comment.
8. Collective ownership(everyone is owner of every mistake)
9. Continuous integration
10. 40-hour week
11. On-site customer

12. Coding standard



Advantages

- Close contact with the customer
- No unnecessary programming work
- Error avoidance through pair programming
- No overtime, teams work at their own pace
- Changes can be made at short notice
- Code is clear and comprehensible at all times
- Stable software through continuous testing

Disadvantages

- Additional work
- Customer must participate in the process
- Relatively large time investment
- Requires self-discipline to practice
- Relatively high costs

System requirements

System requirements in system analysis and design refer to the detailed specifications and capabilities that a system must possess to fulfil the needs and objectives of its stakeholders. These requirements serve as a foundation for the development and implementation of a successful system.

System requirements provide a clear understanding of what the system should do and how it should perform.

When determining system requirements, analysts consider various factors such as functional, non-functional, and technical requirements.

Here's a more detailed description of system requirements in system analysis and design:

1. Functional Requirements: Functional requirements describe the specific functions, features, and capabilities that the system should possess. They define what the system should do and how it should behave in different situations. These requirements are typically documented in use cases or user stories and focus on the interactions between the system and its users. Examples of functional requirements include:

- In one sentence "What the system should do."
- This is the list of the actual services which a system will provide.
- This is the list of the services /functions which a user wants from the software.
- Features of software which the client demands.
- Business rules of the particular organization for which developing software.
 - User authentication: The system should provide a secure login mechanism for users to access their accounts.
 - Data entry: The system should allow users to enter and modify data in a user-friendly and efficient manner.
 - Reporting: The system should generate accurate and customizable reports based on user-defined criteria.
 - Integration: The system should be able to integrate with external systems or APIs to exchange data or perform specific tasks.

- For example, functional requirements for an e-commerce system may include creating an account, adding items to the shopping cart, processing payments, and generating order confirmations.

2. Non-Functional Requirements: Non-functional requirements specify the quality attributes, constraints, and performance expectations of the system. They focus on aspects such as performance, reliability, security, usability, and scalability. These requirements define how well the system should perform its functions. Examples of non-functional requirements include:

- How a system should behave while performing the operations.
- These are the constraints on the services which system is offering e.g. Timing of operations, way to response in particular condition.
- Sometime known as quality attributes.
- For Example: • Recoverability • Response Time • Explain example: Database get ready after update within 2 second.
- **Performance:** The system should respond to user actions within two seconds for 90% of the time.
- **Security:** The system should implement encryption protocols to protect sensitive user data.
- **Usability:** The system should have user-friendly interface to facilitate ease of use.
- **Availability:** The system should have a 99.99% uptime to ensure it is accessible to users at all times.
- **Scalability:** The system should be able to handle a growing number of users and increasing data volumes without significant performance degradation.

Importance of Non-Functional Requirements

- These are not ignore able requirements you can't say as these are non-functional so we should not worry about it.
- These requirements are something which user do not explain most of the time but want in the system.
- These requirements are directly related to the using experience of user and end user.

- 3. User Requirements:** User requirements represent the needs and expectations of the system's end-users. They are gathered through direct interaction with stakeholders, including interviews, surveys, and user workshops.

Understanding user requirements is critical for creating a system that aligns with the users' workflows and preferences

- 4. Business Requirements:** Business requirements focus on the organizational objectives and goals that the system should help achieve. They link the information system's purpose to the broader business strategy.

- 5. System Constraints:** Constraints refer to limitations or boundaries within which the system must operate.

These can include budgetary constraints, time limitations, technology limitations, and legal or regulatory requirements. Constraints help project teams understand the boundaries and make decisions accordingly. For instance, a constraint could be that the system must be developed within a budget of \$100,000 and be compliant with data protection regulations.

- 6. Use Cases and Scenarios:** Use cases and scenarios provide detailed narratives of how users will interact with the system to accomplish specific tasks or processes. They describe the step-by-step interactions and workflows, which are crucial for system design and testing. Use cases help ensure that all user interactions are accounted for and that the system functions as expected in real-world scenarios.

Requirements determination

Requirements determination is a crucial phase in the system analysis and design. It involves the identification, gathering, and analysis of system requirements to understand what the new or modified system should accomplish. This phase lays the foundation for the entire system development process, and its success greatly impacts the final product's effectiveness.

Here are the key steps involved in performing requirements determination:

- 1. Identify Stakeholders:** The first step is to identify the stakeholders who will be affected by the system. Stakeholders can include users, managers, executives, customers, and other individuals or groups with a vested interest in the system. It is important to involve all relevant stakeholders to

ensure their needs and perspectives are considered during requirements determination.

2. **Gather Initial Requirements:** The next step is to gather the initial requirements from the stakeholders. This can be done through various techniques such as interviews, surveys, workshops, and observations. The goal is to elicit information about the stakeholders' needs, objectives, and expectations for the system.
3. **Analyze Requirements:** Once the initial requirements are gathered, they need to be analyzed and refined. This involves reviewing and clarifying the requirements, identifying any inconsistencies or conflicts, and ensuring they are complete and feasible. Requirements analysis may also involve prioritizing requirements based on their importance and impact on the system.
4. **Document Requirements:** The requirements need to be documented in a clear and structured manner to ensure they are well-understood by all stakeholders. This typically involves creating a requirements document or a requirements specification. The document should include detailed descriptions of the functional and non-functional requirements, user requirements, business requirements, and any other relevant information.
5. **Validate Requirements:** After the requirements are documented, they should be validated to ensure their accuracy, completeness, and suitability. This can be done through reviews, discussions, and validations with stakeholders. Validation helps identify any potential gaps or issues in the requirements and allows for necessary adjustments before proceeding to the next stages of system development.
6. **Manage Requirements Changes:** Throughout the system development lifecycle, requirements may change due to various factors such as evolving business needs, technological advancements, or new insights gained during the development process. It is important to establish a process for managing requirements changes, including evaluating the impact of changes, documenting them, and obtaining appropriate approvals from stakeholders.

By performing these steps, system analysts and designers can effectively determine the requirements of the system, ensuring that the final solution meets the needs and expectations of all stakeholders.

Requirements determination is an iterative process, and it's important to gather feedback and refine the requirements continuously as the project progresses.

Traditional Methods for Determining Requirements

Traditional methods for determining requirements in system analysis and design are well-established approaches that have been used for many years. These methods are based on structured and systematic techniques for gathering, documenting, and analyzing requirements. Some of the traditional methods include:

1. **Interviews:** Conducting one-on-one or group interviews with stakeholders to gather information about their needs, expectations, and concerns regarding the system. Interviews allow for direct interaction and provide an opportunity to ask follow-up questions for better understanding.

Tips for a Successful Interview

- Be on time.
- Know the interviewer's name, its spelling, and pronunciation.
- Have some questions of your own prepared in advance.
- Tell the truth.
- Listen carefully to the interviewer.
- No interview is complete until you follow up with a thank-you note.

2. **Questionnaires and Surveys:** Using written questionnaires or online surveys to collect information from a large number of stakeholders simultaneously. Surveys are useful for gathering opinions, preferences, and feedback from a diverse group of users.

Questionnaires have the advantage of gathering information from many people in a relatively short time and of being less biased in the interpretation of their results. Choosing right questionnaires respondents and designing effective questionnaires are the critical issues in this information collection method.

In most situations, one copy of questionnaires obviously cannot fit to all the users.

To conduct an effective survey, the analyst should group the users properly and design different questionnaires for different group.

Moreover, the ability to build good questionnaires is a skill that improves with practice and experience.

When designing questionnaires, the analyst should concern the following issues at least:

- The ambiguity of questions.
- What kind of question should be applied, open-ended or close-ended?
- What is the proper length of the questionnaires?

Interview Outline	
Interviewee: Name of person being interviewed	Interviewer: Name of person leading interview
Location/Medium: Office, conference room, or phone number	Appointment Date: Start Time: End Time:
Agenda: Introduction Background Project Overview of Interview Topics to be covered Permission to Tape Record Topic 1 Questions Topic 2 Questions Summary of Major Points Questions from Interviewee Closing	Approximate Time:
General Observations: Interviewee seemed busy - probably need to call in a few days for follow-up questions since he gave only short answers. PC was turned off- probably not a regular PC user.	

Unresolved Issues, Topics not covered: He needs to look up figures of previous years. He raised the issue of how to handle the problem, but he did not have to discuss.	
When to ask question, if conditional Question number 1 Have you used current system? If so, how often? If yes, go to Question number 2.	Answer Yes, I ask for report weekly Observation Seemed anxious - may be over-estimating usage frequency
Question 2 What do you like least about this system?	Answer Using wrong units Observations: Options determine the right units, but user chose the wrong one.

3. **Observation:** Observing users and system processes in their actual working environment. This technique helps analysts understand existing workflows, identify pain points, and observe how users interact with the current system or perform their tasks.
4. **Document Analysis:** Reviewing existing documentation, such as business plans, reports, manuals, and procedures, to extract relevant requirements and gain insights into the existing system.
5. **Data Collection Techniques:** Using data collection techniques such as sampling, data mining, and data analysis to understand patterns, trends, and correlations in the data, which can contribute to the identification of requirements.
6. **Use Cases and Scenarios:** Developing use cases and scenarios that describe specific interactions between users and the system. Use cases help in understanding the functional requirements and system behavior in real-world situations.

These traditional methods provide a structured approach to gather, analyze, and document system requirements. However, it's important to note that these methods can be time-consuming and may not always capture all the needs of stakeholders.

Contemporary Methods for Determining System Requirements

Even though we called interviews, questionnaires, observation, and document analysis traditional methods for determining a system's requirements, all of these methods are still used by analysts to collect important information. Today, however, additional techniques are available to collect information about the current system, the organizational area requesting the new system, and what the new system should be like. In this section, you learn about two modern information-gathering techniques for analysis: joint application design (JAD) and prototyping. These techniques can support effective information collection and structuring while reducing the amount of time required for analysis.

1. JAD Sessions (Joint Application Development):

JAD was developed by Chuck Morris and Tony Crawford of IBM in 1977 and has since been proven successful on thousands of software projects

across industry sectors and application boundaries. Joint Application Development (JAD) is a user requirements elicitation process that involves the system owner and end users in the design and development of an application through a succession of collaborative workshops called JAD sessions. The JAD approach leads to shorter development lifecycles and greater client satisfaction because it draws users and information systems analysts together to jointly design systems in facilitated group sessions.

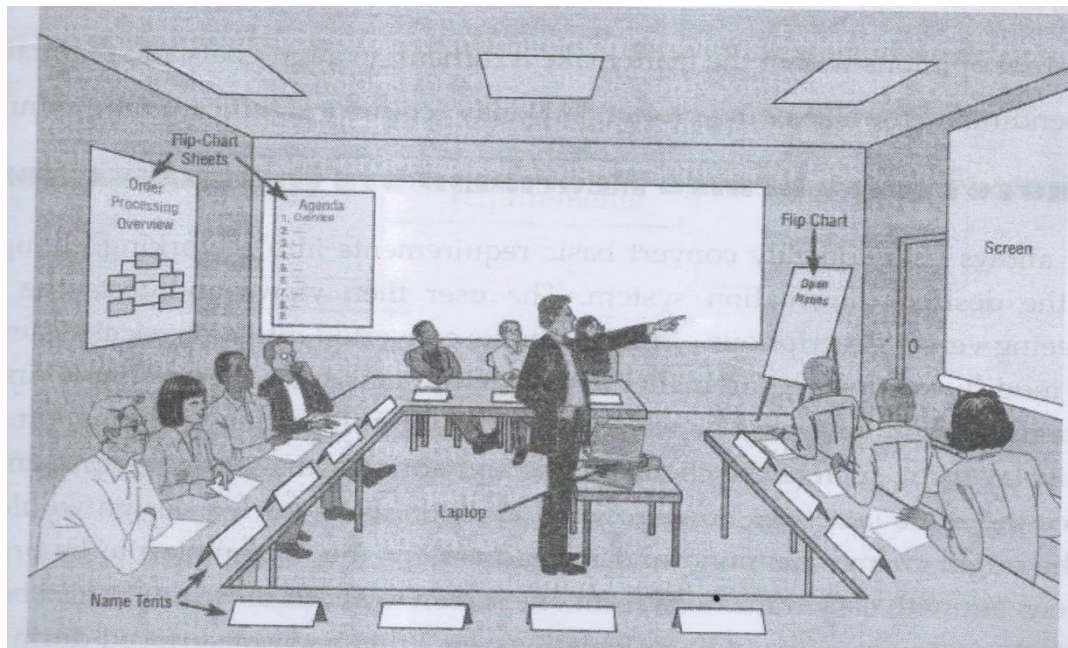
The primary purpose of using JAD in the analysis phase is to collect systems requirements simultaneously from the key people involved with the system.

JAD sessions are usually conducted in a location away from where the people involved normally work, in order to limit distractions and help participants' better concentrate on systems analysis.

The following is a list of typical JAD participants:

- **JAD session leader:** The JAD leader organizes and runs the JAD. This person has been trained in group management and facilitation as well as in systems analysis. The JAD leader sets the agenda and sees that it is met. He or she remains neutral on issues and does not contribute ideas or opinions, but rather concentrates on keeping the group on the agenda, resolving conflicts and disagreements, and soliciting all ideas.
- **Users:** The key users of the system under consideration are vital participants in a JAD. They are the only ones who clearly understand what it means to use the system on a daily basis.
- **Managers:** Managers of the work groups who use the system in question provide insight into new organizational directions, motivations for and organizational impacts of systems, and support for requirements determined during the JAD.
- **Sponsor:** As a major undertaking, because of its expense, a JAD must be sponsored by someone at a relatively high level in the company such as a vice president or chief executive officer. If the sponsor attends any sessions, it is usually only at the beginning or the end.
- **Systems analysts:** Members of the systems analysis team attend the JAD, although their actual participation may be limited. Analysts are there to learn from users and managers, not to run or dominate the process.
- **Scribe:** The scribe takes notes during the JAD sessions, usually on a personal computer or laptop.

- **IS staff:** Besides systems analysts, other IS staff, such as programmers, database analysts, IS planners and data-center personnel may attend the session. Their purpose is to learn from the discussion and possibly to contribute their ideas on the technical feasibility of proposed ideas or on the technical limitations of current systems.



2. **Prototyping:** Creating a basic working model of the system to gather early feedback from stakeholders and refine requirements based on user interactions with the prototype.

System acquisition

- System acquisition refers to the process of acquiring a new software system or information technology (IT) system for an organization.
- It involves identifying the organization's requirements, evaluating potential solutions, selecting the most suitable system, and acquiring it through various means such as purchasing, development, or outsourcing.

It is a complex process that involves many steps, including:

- 1. Identifying the need for a new or modified system.** This involves understanding the current system's capabilities and limitations, as well as the organization's future needs.
- 2. Developing a requirements document.** This document specifies the functional and non-functional requirements for the new or modified system.
- 3. Soliciting (ask for) proposals from vendors.** This involves identifying potential vendors and requesting proposals for the new or modified system.
- 4. Evaluating proposals.** This involves evaluating the proposals from vendors and selecting the best one.
- 5. Negotiating a contract with the vendor.** This contract specifies the terms and conditions of the purchase of the new or modified system.
- 6. Implementing the new or modified system.** This involves installing the system, configuring it, and training users.
- 7. Testing the new or modified system.** This involves testing the system to ensure that it meets the requirements.
- 8. Deploying the new or modified system.** This involves making the system available to users.

System acquisition is a critical process that can have a significant impact on an organization. It is important to follow a systematic approach to ensure that the new or modified system meets the organization's needs and is implemented successfully.

Here are some of the key benefits of system acquisition:

- **Improved efficiency and productivity.** A new or modified system can help to improve efficiency and productivity by automating tasks, reducing errors.

- **Increased customer satisfaction.** A new or modified system can help to improve customer satisfaction by providing better service, reducing wait times, and making it easier for customers to do business with the organization.
- **Enhanced decision-making.** A new or modified system can help to enhance decision-making by providing better information and insights.

However, there are also some risks associated with system acquisition, such as:

- **Cost overruns.** The cost of a new or modified system can sometimes exceed the original budget.
- **Implementation delays.** The implementation of a new or modified system can sometimes take longer than expected.
- **System failure.** A new or modified system can sometimes fail to meet the organization's needs.

It is important to carefully consider these risks before embarking (Starting) on a system acquisition project. By doing so, organizations can help to ensure that the project is successful and that the benefits outweigh the risks.

Outsourcing

- Outsourcing is the practice of contracting out business functions or processes to third-party providers.
- It can be a great way to save money, improve efficiency, and gain access to specialized skills. However, there are also some potential risks associated with outsourcing, such as loss of control, security concerns, and cultural clashes.

Advantages of outsourcing:

- **Cost savings:** Outsourcing can help to save money on labour costs, benefits, and overhead.
- **Improved efficiency:** Outsourcing can help to improve efficiency by freeing up internal resources to focus on core competencies.
- **Access to specialized skills:** Outsourcing can help organizations gain access to specialized skills that they may not have in-house.
- **Increased flexibility:** Outsourcing can help organizations to be more flexible in their operations. For example, they can scale up or down their operations more easily as needed.

Disadvantages of outsourcing:

- **Loss of control:** When you outsource, you are giving up some control over your business processes. This can be a problem if you are not careful to choose a reputable outsourcing provider.
- **Security concerns:** There are always security concerns when you outsource sensitive data or processes. You need to make sure that your outsourcing provider has strong security measures in place.
- **Cultural clashes:** If you outsource to a company in a different country, you may experience cultural clashes. This can lead to misunderstandings and problems.

Overall, outsourcing can be a great way to save money and improve efficiency. However, it is important to consider the pros and cons carefully before making a decision.

Here are some additional factors to consider when making an outsourcing decision:

- **The nature of the work to be outsourced:** Some work is more easily outsourced than others. For example, it is easier to outsource IT-related work than customer service work.
- **The size and complexity of the organization:** Larger and more complex organizations are more likely to benefit from outsourcing.
- **The organization's culture:** Some organizations are more comfortable with outsourcing than others.

If you are considering outsourcing, it is important to do your research and choose a reputable outsourcing provider. You should also carefully define the scope of work to be outsourced and develop a clear contract.

Sources of software

There are many different sources of software, each with its own advantages and disadvantages. Here are some of the most common sources of software:

- **Packaged software providers:** These companies sell software that is ready to use. Packaged software is often a good choice for organizations that need a specific software solution and do not have the time or resources to develop their own software.
- **Open-source software:** This software is developed and maintained by a community of users. Open-source software is often free to use and modify, which can be a great advantage for organizations that want to save money or customize their software.
- **In-house development:** This is the process of developing software within an organization. In-house development can be a good choice for organizations that have specific needs that cannot be met by packaged software or open-source software.
- **Cloud computing:** This is a model of computing where services are delivered over the internet. Cloud computing can be a good choice for organizations that need to access software on demand or that want to avoid the cost of maintaining their own software infrastructure.
- **Information technology services firms:** These companies provide a variety of services, including software development, software maintenance, and software consulting. Information technology services firms can be a good choice for organizations that do not have the internal resources to develop or maintain their own software.
- **Enterprise solution software:** It is a type of software that is designed to meet the needs of large organizations. These software solutions are typically complex and feature-rich, and they can be used to automate a wide range of business processes.

The best source of software for an organization will depend on its specific needs and requirements. It is important to carefully consider the advantages and disadvantages of each source of software before making a decision.

Here are some additional factors to consider when choosing a source of software:

- **The cost of the software:** Packaged software is often more expensive than open-source software, but it may be a better choice for organizations that need a specific software solution.
- **The level of customization required:** If the organization needs to customize the software, open-source software may be a better choice.
- **The level of support required:** Some sources of software, such as packaged software providers, offer more support than others.
- **The organization's technical expertise:** If the organization does not have the technical expertise to develop or maintain its own software, an information technology services firm may be a good choice.

By carefully considering these factors, organizations can choose the best source of software for their needs.

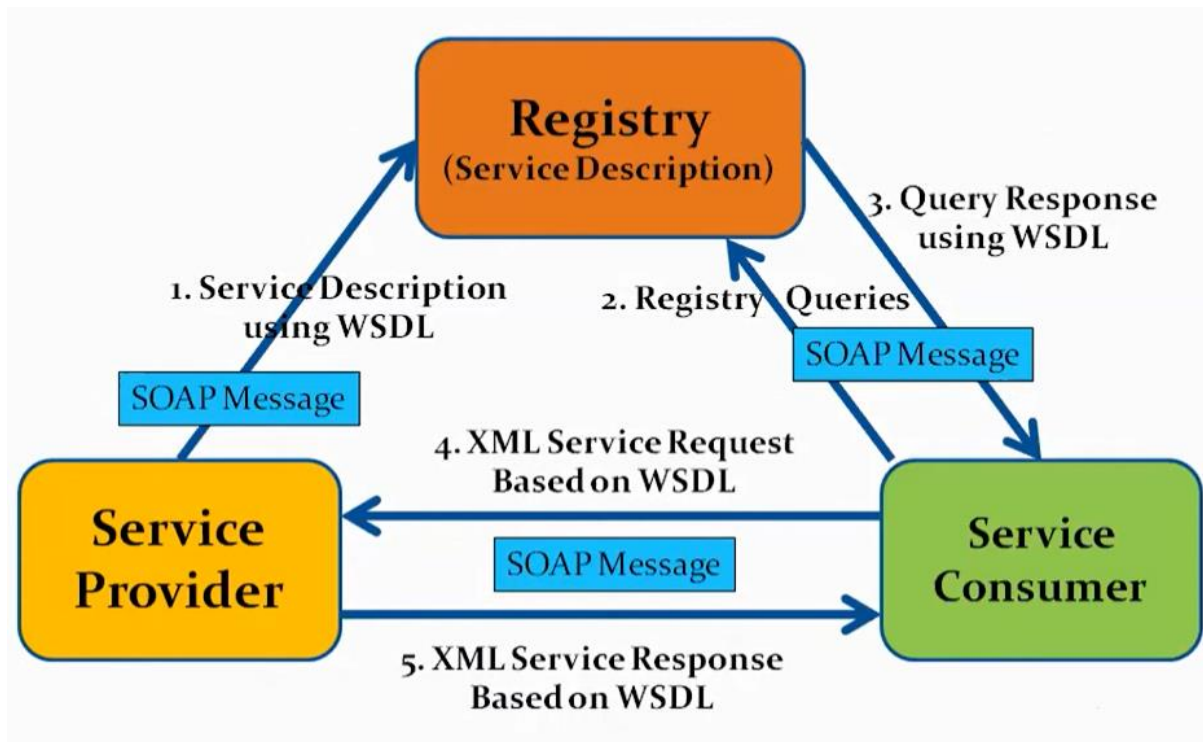
Service-oriented architecture (SOA)

- SOA is an architectural style that supports service orientation.
- Service orientation is design model of computer software which are using as service now a days.
- SOA is an architectural approach in which applications make use of services available in the network.
- In this architecture, services are provided to form applications, through a network call over the internet.
- Each service in SOA is a complete business function in itself. The services are published in such a way that it makes it easy for the developers to assemble their apps using those services.
- SOA allows users to combine a large number of facilities from existing services to form applications.
- Service-oriented architecture is an implementation of the "service concept" or "service model" of computing.
- In this architectural style, business processes are implemented as software services, accessed through a set of strictly defined application program interfaces (APIs).
- It is only a concept and not limited to any programming language or platform.

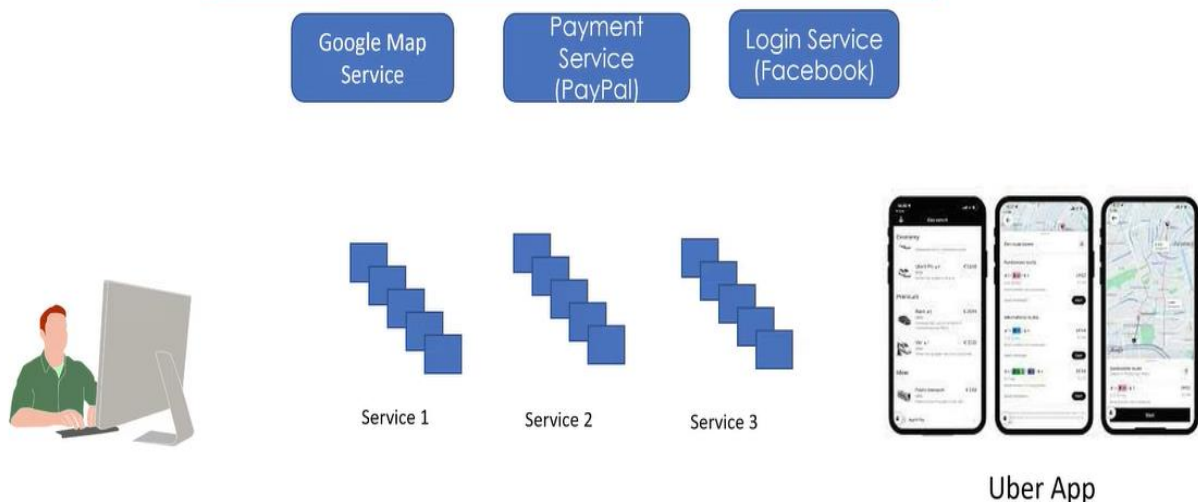
There are three major roles and operations within Service-oriented Architecture:

To implement the SOA developer need web services and web services is created with the help of web services frame work.

SO web service frame work is heart of SOA



Service Oriented Architecture (SOA)



Main concept or component of SOA

SOAP (Simple Object Access Protocol)

- SOAP, is a protocol specification for exchanging structured information in the implementation of web services in computer networks.
- It uses XML Information Set for its message format, and relies on other application layer protocols, most notably Hypertext Transfer Protocol (HTTP) or Simple Mail Transfer Protocol (SMTP), for message negotiation and transmission.

WSDL (Web Service Definition Language)

- WSDL is an XML-based interface definition language that is used for describing the functionality offered by a web service.
- The acronym is also used for any specific WSDL description of a web service (also referred to as a WSDL file), which provides a machine-readable description of how the service can be called, what parameters it expects, and what data structures it returns.

UDDI (Universal Description Discovery and Integration)

- UDDI is a platform-independent, Extensible Markup Language protocol includes a (XML)-based registry by which businesses worldwide can list themselves on the Internet, and a mechanism to register and locate web service applications.
- UDDI is an open industry initiative, sponsored by the Organization for the Advancement of Structured Information Standards (**OASIS**), for enabling businesses to publish service listings and discover each other, and to define how the services or software applications interact over the Internet.

Advantages of SOA:

- **Easy maintenance:** As services are independent of each other they can be updated and modified easily without affecting other services.
- **Platform independent:** SOA allows making a complex application by combining services picked from different sources, independent of the platform.
- **Availability:** SOA facilities are easily available to anyone on request.
- **Service reusability:** In SOA, applications are made from existing services. Thus, services can be reused to make many applications.
- **Reliability:** SOA applications are more reliable because it is easy to debug small services rather than huge codes
- **Scalability:** Services can run on different servers within an environment, this increases scalability

Disadvantages of SOA

- Complex design and implementation
- More network traffic

SRS- Software Requirement Specification

Def. A documents use to describe the behaviour of the software System, Functional, Non-Functional requirements of the software system.

Users of the SRS:

- Development Team
- Maintenance Team
- Clients

Contents of the SRS

1. **Category:** What kind of your software is
e.g.: Desktop application, Web application, Android app
2. **Purpose:** Describe what is the purpose of making this system
(as nothing is without any reason)
3. **Scope:** What is the area it covering, what is its range, to what limits it will help you.
4. **Introduction:** Define the existing system and Proposed System. *(In the perspective how the proposed one is better than existing.)*
5. **Advantages:** Define the advantages of the system. *(In every perspective).*
6. **Functional Requirements of the System**
7. **Non-Functional Requirements of the System**
8. **Software Tools:** Mention the software tools which will involve in all this development process.
9. **Deployment:** What kind of Environment will be needed to deploy the Software?
E.g. OS, RAM, Processor and etc.
10. **Hardware Specifications:** The hardware required to develop this system.
11. **Gantt chart:** Chart use to show the schedule of project.
(optional)

This is the Demo SRS about the project “*Virtual class room system*”

Project Title: Virtual Class Room System

Category: Web Application

PURPOSE

The purpose of developing this “Virtual class room system” is to keep the students and teachers connected through a virtual platform even when they cannot reach to the physical platform. (*E.g. University, Colleges*)

SCOPE

It will integrate the benefits of a physical classroom with the convenience of a ‘no physical-bar’ virtual learning environment, minus the change hazards and expenses. This will provide more flexibility in the existing learning platform structures. It provides a means of collaborative learning for the students and also promotes the distance learning.

INTRODUCTION

Introduction contain the following sub categories

Existing System

The present system is a manual system or a semi-automated system. Manual system involves paper work in the form of maintaining various files and manuals. Maintaining critical information in the files and manuals is full of risk and a tedious process. A manual system has following disadvantages

- ✓ It’s a limited system and fewer users friendly.
- ✓ Searching of particular information is critical it takes lot of time.
- ✓ In the existing system students need to learn course in the class room only.
- ✓ In the manual system student need bare the lot of time and cost.
- ✓ The existing system need to travel a location in which one have to participate in the course.
- ✓ Conduct of examination and compilation of result is a tedious process.
- ✓ The existing system needs to save the information in the form of files.

- ✓ Limited sharing is possible if the data is in the form of paper or disk drives.
- ✓ The manual system gives us less security for saving data, some data may be lost due to mismanagement.

Proposed System

As the virtual classroom is one that aims to give the student an experience equal to or better than sort they would find in a traditional classroom. There are obviously many advantages of the virtual classroom to the student, as well as the teacher and the associated educational institution.

The system after careful analysis has been identified to be presented with the following modules:

- ✓ **Student Registration:** Admin can register a student for a specific program.
- ✓ **Video Lectures:** Student can watch and listen the class room session repeatedly which will be uploaded by the teacher.
- ✓ **Result:** Students can get the result via this web application
- ✓ **Attendance:** It includes attendance of students for a specific class.
- ✓ **Record Storage:** The user information files should be stored in centralized database which can be maintained by the system.
- ✓ **Authentication:** Authentication of this application will be provided for only registered members.
- ✓ **Activities for Students:** The teacher can upload assignments to the students, multimedia which a student can download and listen when required.
- ✓ **Administration Access:** Administration would be able to keep an eye on the records of students.
- ✓ **Library:** Student helping material would be available.

Advantages for Students

Advantages for the students are explained below

- ✓ **Time and Cost Saving:** For students of virtual classrooms, advantages primarily concern access, time, and cost factors compared to those incurred from attending the traditional, physical classroom.

- ✓ **Remove Traveling:** Virtual classrooms remove the need of travel to attend the lectures.
- ✓ **Education at Doorstep:** Our system is a platform to provide education at doorstep.

Advantages for Faculty

Advantages for the teachers are explained below

- ✓ **Flexibility for Teacher:** For teachers of virtual classrooms, working part-time is a possibility because of the flexible nature of the classroom.
- ✓ **Effective Communication:** Another advantage to the teacher is that they have time to respond to student communications in the virtual classroom, which gives the teacher a chance to construct the best possible response to the student.
- ✓ **Independent of Location:** The teacher is not required to work from an office or classroom to instruct lessons or organize material and lessons this can be done from anywhere where they have access to a computer with an Internet connection.

FUNCTIONAL REQUIREMENTS

"Functional requirement describe what a system should do." [4] Functional requirements of our system are explained below.

- ✓ Users must have valid User ID and password to login thus creating their individual profiles.
- ✓ Administration can register new teachers.
- ✓ Students can choose courses.
- ✓ Attend lectures either at the scheduled time or view lecture later.
- ✓ Video lectures facility can also be available for the students.
- ✓ Students can check attendance, result and assignments.
- ✓ Faculties can take lectures, upload assignments, announcements, compile results and also can upload lectures and other discussions in various formats as in videos, power point presentation etc.
- ✓ Notice board facility can also be available for the students from administration and teachers.
- ✓ Notice board can also be available for the teachers from administration.
- ✓ Forms and guidance can also be available for the students regarding any query.

- ✓ There can be platform to discuss various queries and to put up suggestions posted both by students and teachers.
- ✓ Documents and media library that can help in active learning of a student.
- ✓ Student can send any application to the administration for any query.
- ✓ Administration can post upcoming events banner.

NON FUNCTIONAL REQUIREMENTS

Requirement that specifies criteria that can be used to judge the operation of a system are called non-functional requirements. [5] Non-functional requirements of our system are mentioned below

- ✓ Secure access of confidential data (user's details).
- ✓ Maximum time availability
- ✓ Better component design to get efficiency at peak time.
- ✓ Flexible service based architecture will be highly desirable for future extension.
- ✓ Teacher registration facility is accessible by administration only.

SOFTWARE TOOLS

Database Server: Microsoft SQL Server-2012

Client: Microsoft Internet Explorer or any web browser

Development Tools: Microsoft Visual Studio 2013

Programming Language: C#.Net

DEPLOYMENT

Operating System Server: Window 8, Linux, UNIX

HARDWARE SPECIFICATION

Processor: Intel Core i5

RAM: 4GB

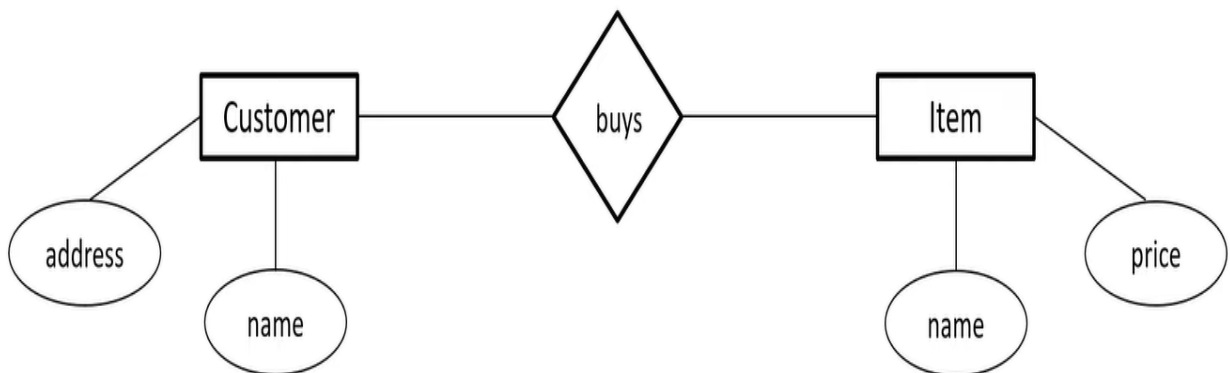
Hard Disk: 640 TB

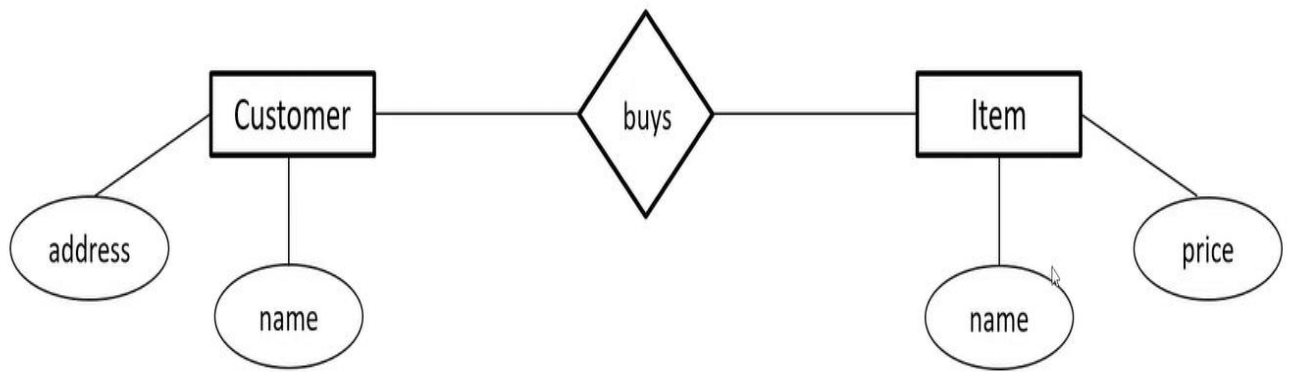
What is ER – Model?

- ER-Model stands for "Entity Relationship Model".
- It is a graphical approach for designing a database.
- It is a database modelling technique that generates an abstract diagram of system's data and that diagram is used in designing a relational database.
- It was developed by Peter Chen in 1976.

What is ERD?

- ERD stands for "Entity Relationship Diagram".
- The diagram generated by ER-Model is known as ERD.
- ERD shows the relationship of different entities stored in the database.





Customer

Name	Address
Rohan	12/ABC
Sohan	15/XYZ

Item

Name	Price
mouse	125
keyboard	150

- ER diagram explains the logical structure of database.
- At last, an ERD is implemented as a database.
- It helps us to analyze data requirements systematically to produce a well-designed database.
- It is considered as a best practice to complete ER modelling before implementing your database.
- ER diagrams are based on three basic concepts: entities, attributes and relationships.

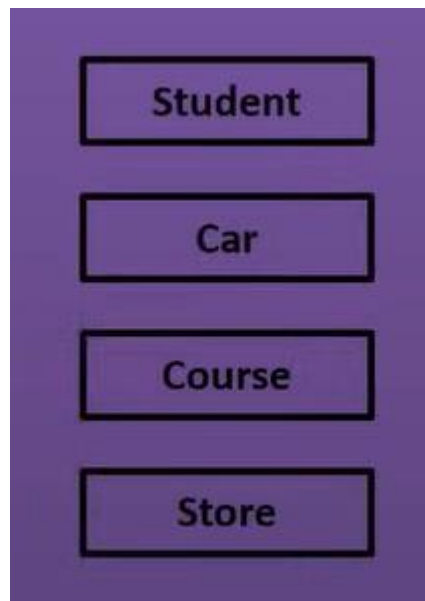
Entity:

An entity is a real world object or concept about which you want to store information. An entity can be place, person, object or a concept etc.

❑ Example

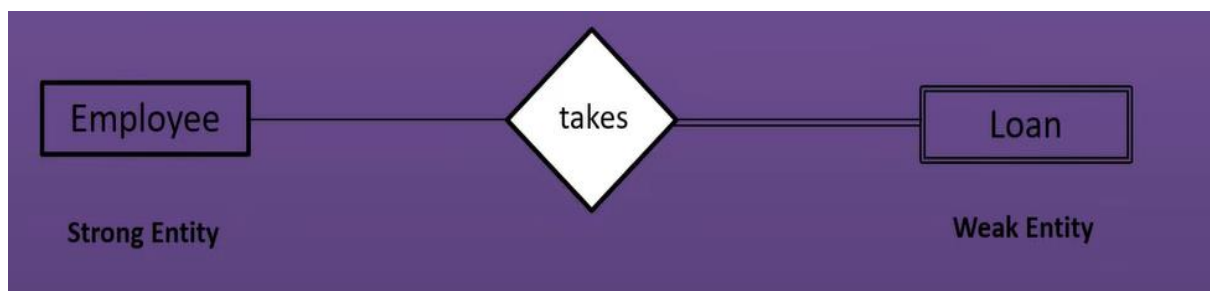
- Person: student, employee, patient
- Place: building, store
- Object: car, bike, product
- Concept: course

It is represented using rectangle shape.



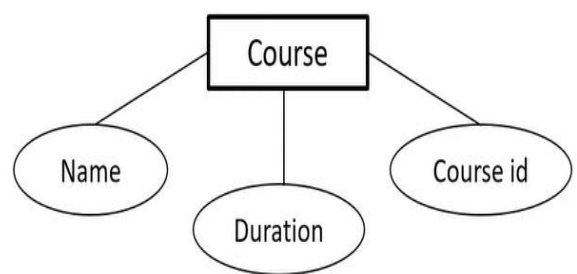
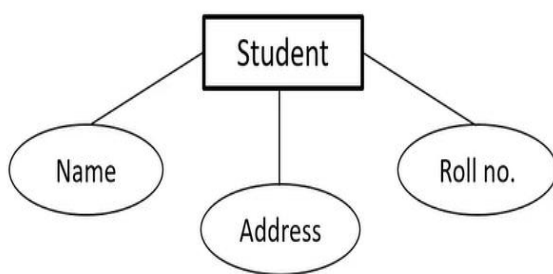
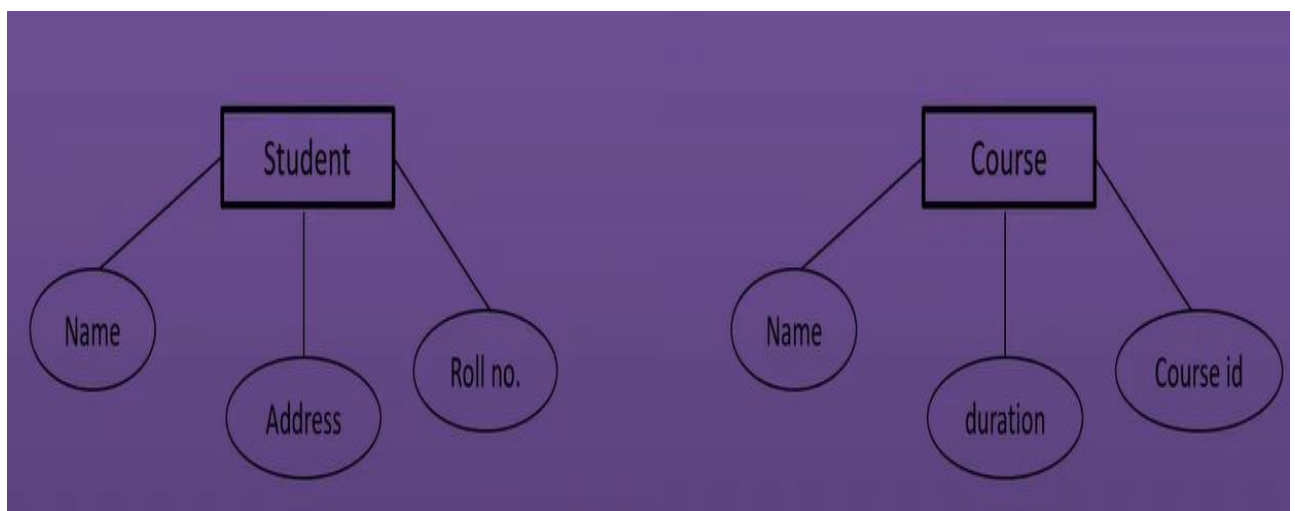
There are two types of Entities:

- **Strong Entity:** An entity which does not depend on any other entity to exist is called strong entity. Example: employee
- **Weak Entity:** An entity which depends on other entity to exist is called weak entity. Example: loan



Attributes:

- An attribute are the properties or characteristics of an entity.
- It describes an entity.
- It is represented using oval shape.



Student

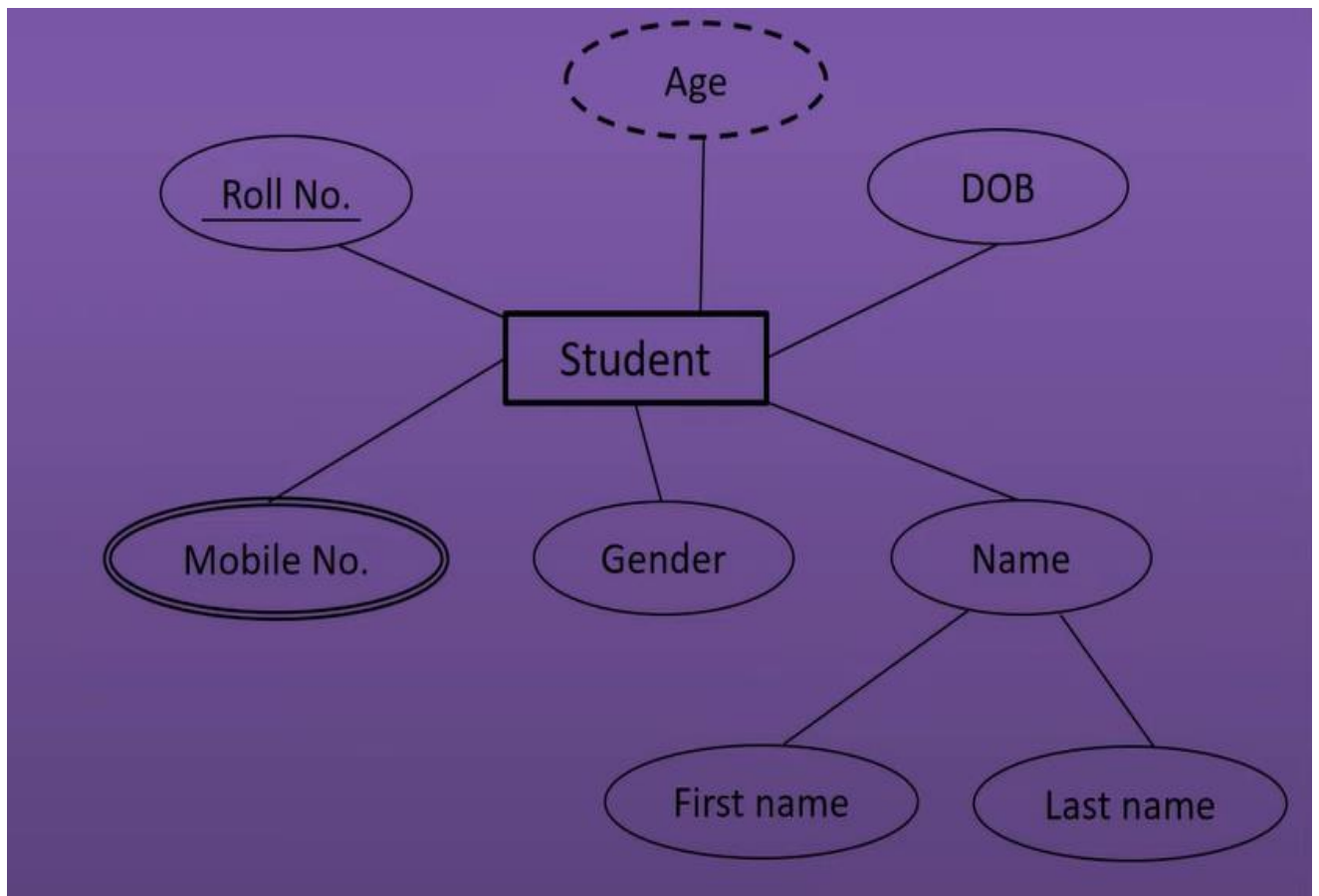
Name	Address	Roll No.
Rohan	12/ABC	001
Sohan	15/XYZ	002

Course

Name	Duration	Course id
Core java	125	101
Advance java	150	102

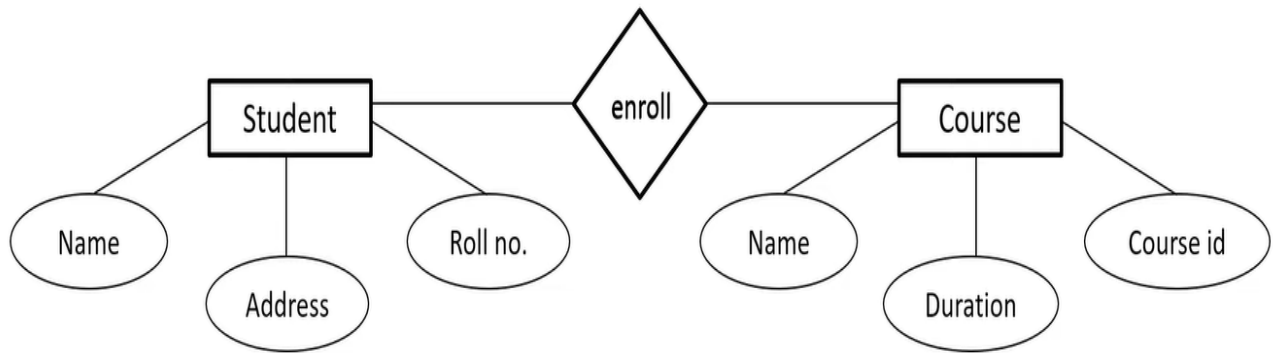
Attribute Types

- Single-value attribute
- Multi-value attribute
- Simple attribute
- Composite attribute
- Derived attribute
- Key attribute



Relationship:

- It specifies how entities are associated with each other.
- It is represented using diamond shape.



Student		
Name	Address	Roll No.
Rohan	12/ABC	001
Sohan	15/XYZ	002

enroll	
Course id	Roll No.
101	001
102	002

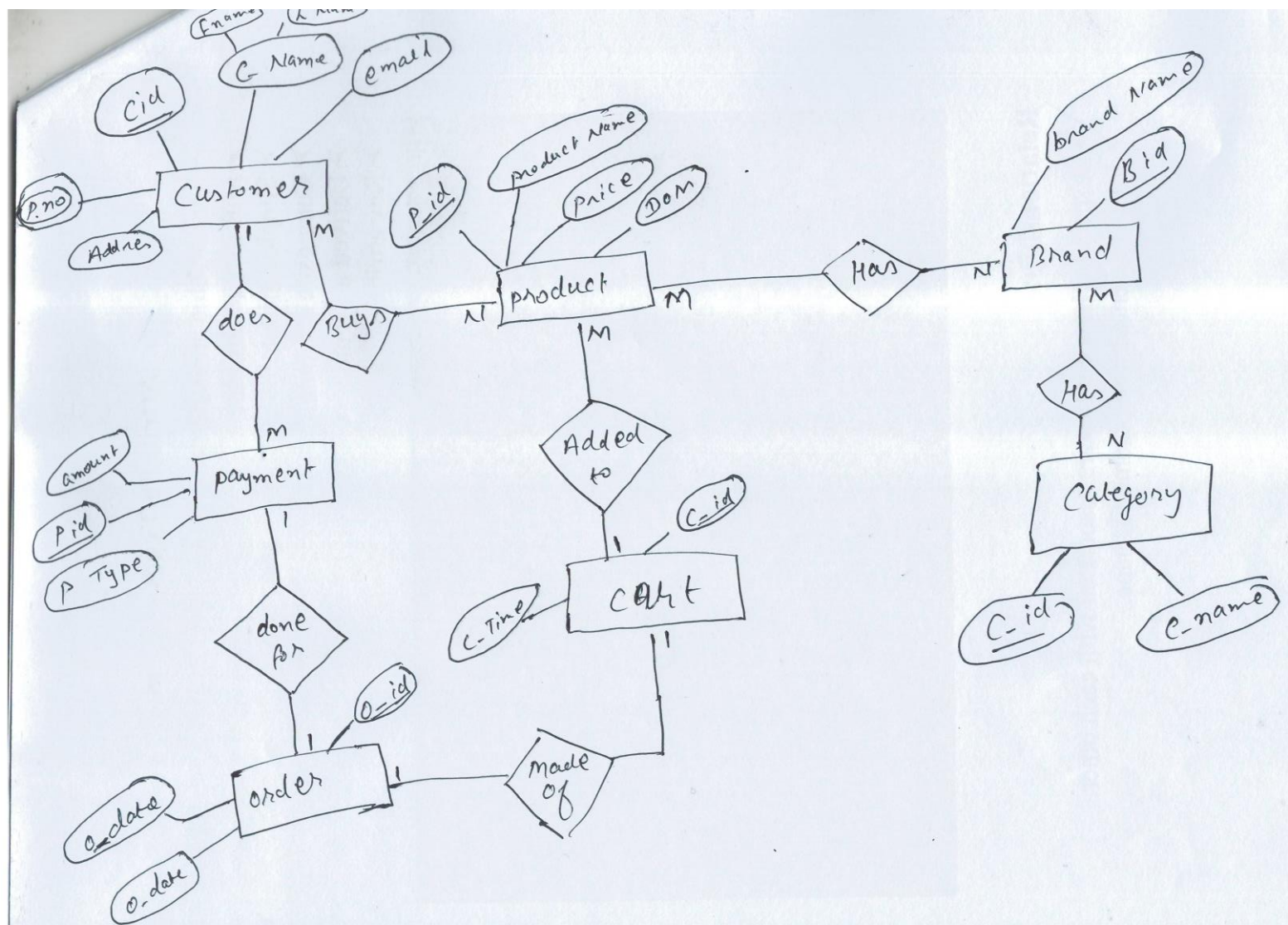
Course		
Name	Duration	Course id
Core java	125	101
Advance java	150	102

Importance and uses of ERD

- Database design
- Database debugging
- Problem in requirement gathering

Entity

- ① Customer
2. product
3. category
4. payment
5. order
6. cart
7. Brand



Different types of cardinal relationships are:

- One-to-One Relationships
- One-to-Many Relationships
- May to One Relationships
- Many-to-Many Relationships

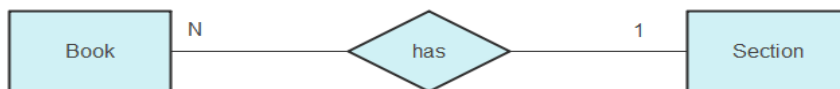
one-to-one (1:1)



one-to-many (1:N)



many-to-one (N:1)

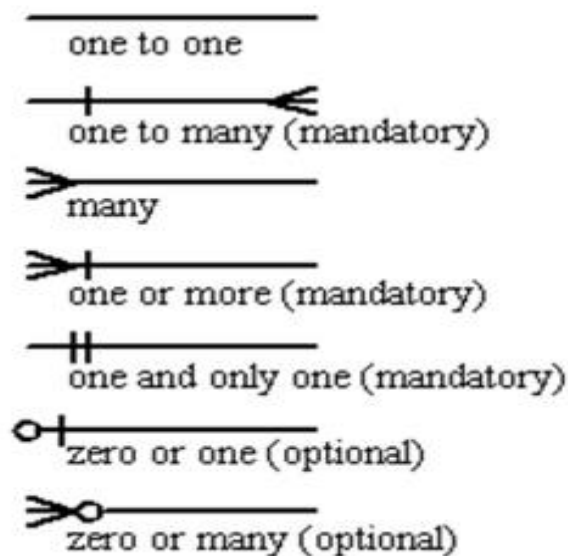


many-to-many (M:N)



Cardinality Notations

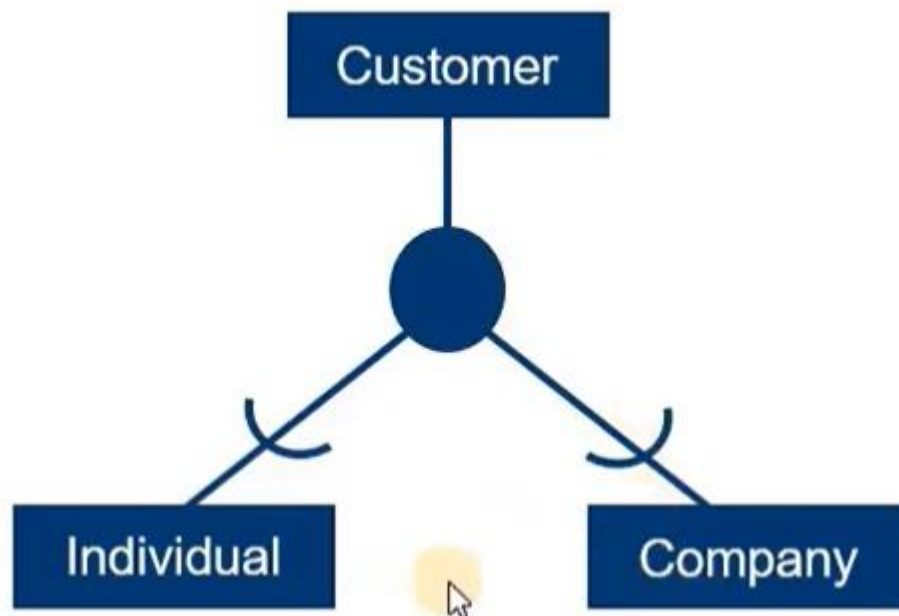
Information Engineering style



Representing Super- Types and Sub-Types

Supertype is an entity type that has got relationship (parent to child relationship) with one or more subtypes and it contains attributes that are common to its subtypes.

Subtypes are subgroups of the supertype entity and have unique attributes, but they will be different from each subtype.



CUSTOMER contains

<u>CustomerID</u>
Address
Phone

INDIVIDUAL contains

NIC
Profession
Designation

ORGANIZATION contains

<u>RegistrationID</u>
<u>ContactPerson</u>
<u>TaxID</u>

DATA AND PROCESS MODELLING TECHNIQUES

- Data and Process Modelling Techniques are used by systems analyst during system analysis to depict how various processes of a system transform data into useful information in graphical form.
- Output of process modelling activity is a logical model that supports business operations and meets requirements of managers and users.
- A logical model shows what the system must do, regardless of how it will be accomplished physically.
- System analyst may use one or more of the following traditional and structured analysis tools to represent system data and process graphically.
 1. System flow chart
 2. Data flow diagram (DFD)
 3. Data dictionary
 4. Decision tree
 5. Decision table

What is DFD?

- DFD stands for Data Flow Diagram.
- DFD is used to show the flow of data through software system.
- DFD is a graphical or pictorial representation of logical flow of data.
- Main advantage of DFD is that it provides a logical model of the system that shows what a system does rather than how it is being done.
- Aim is to clarify system requirements and identify major processes, transformations of data, data stores and external entities to ensure developed model is accurate and easy to understand.
- DFD is also known as bubble charts as it consists of a series of bubbles joined by lines.

Components of the DFD

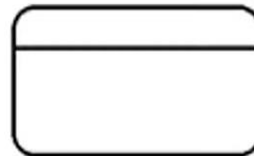
- **External Entity:** It provide the data to your software or consume data through it.
 - External entity is a source or destination of data.
 - It can be files, departments, person, vendors, customer, other information systems etc.
 - External entities show the system boundaries and how a system interacts with outside world.



- **Process:** It take data and convert it into information.



Process



- Process is the people or procedures involved in the system that are used to transform data.
- It receives input data, processes it and produces output.

- **Data Flow:**

- It shows the flow of data incoming or outgoing.
- Data flow is a path that shows movement of data from one process and to another process in a system from origin to destination.



- **Data stores:** This is your database or files where we store data



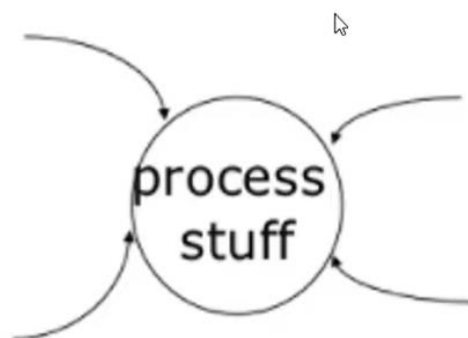
Some important notes

- Data cannot flow from one data store to other directly.
- Data cannot flow between two external entities.
(A process should involve between)

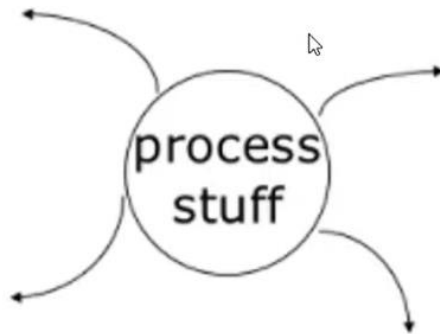
Data store cannot sink or source only

- Sink all data coming in
- Source all data going out (possible for level 2)

➤ **No black hole process**



➤ **No miracle**



Level of DFD

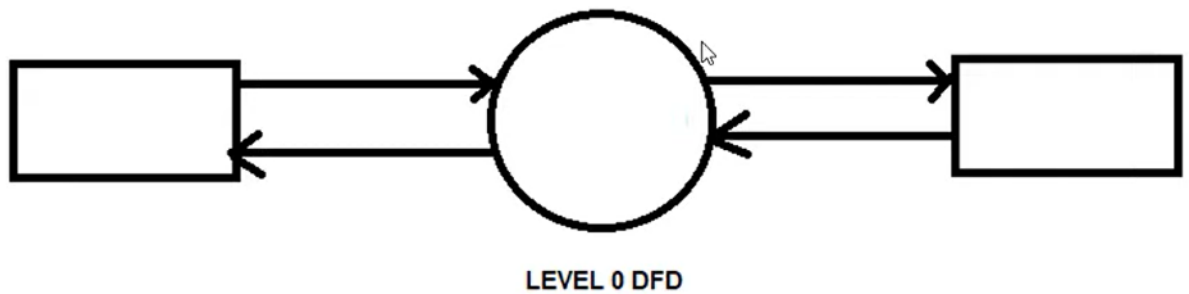
- Level 0 DFD or Context diagram
- Level 1 DFD
- Level 2 DFD

PROCEDURE FOR DRAWING A DFD:

STEP 1

- Represent the whole system by a single process and identify its input and outputs.
- This is known as context diagram or zero level DFD.
- A context DFD is a top level view of a system that shows boundaries and scope of the system.
- To draw context level DFD put a single process symbol in the centre which represents the entire system. Then place external entities around the central process symbol and use data flows to connect entities to it.
- Data stores are not being used in context diagram because these are internal to the system.
- Information about names and contents of external entities and data flows are collected during fact gathering process.

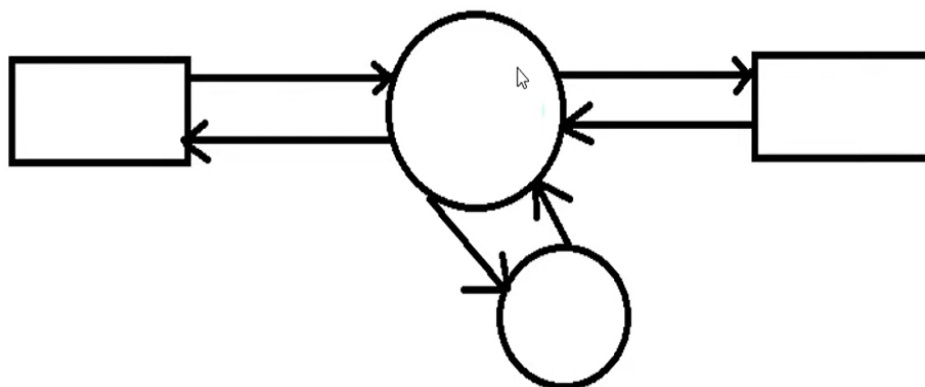
Level 0 DFD or Context diagram



STEP 2

- Context diagram shows the most general view of a system which contains only one process.
- To know more details about the system next level of DFD are drawn.
- To draw next level of DFD, identify major processes of the system and draw DFD considering these major processes only and also identify inputs and outputs for each and every major process.
- This is known as top level DFD or first level DFD.

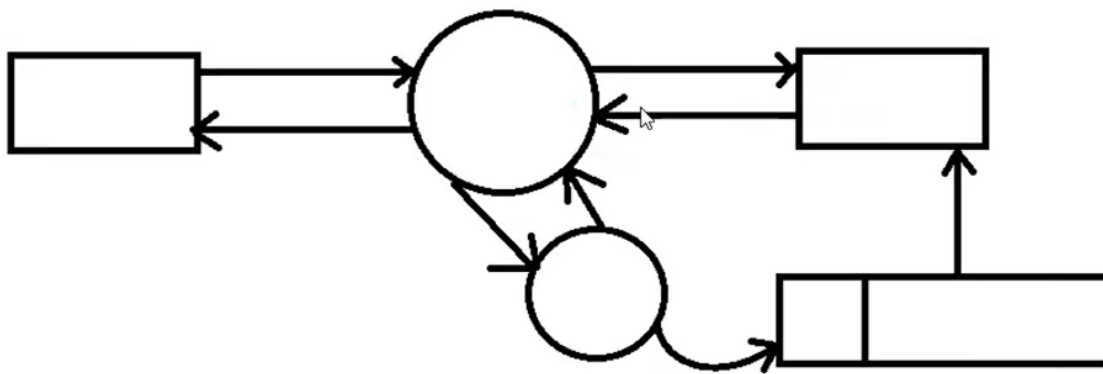
Level 1 DFD



STEP 3

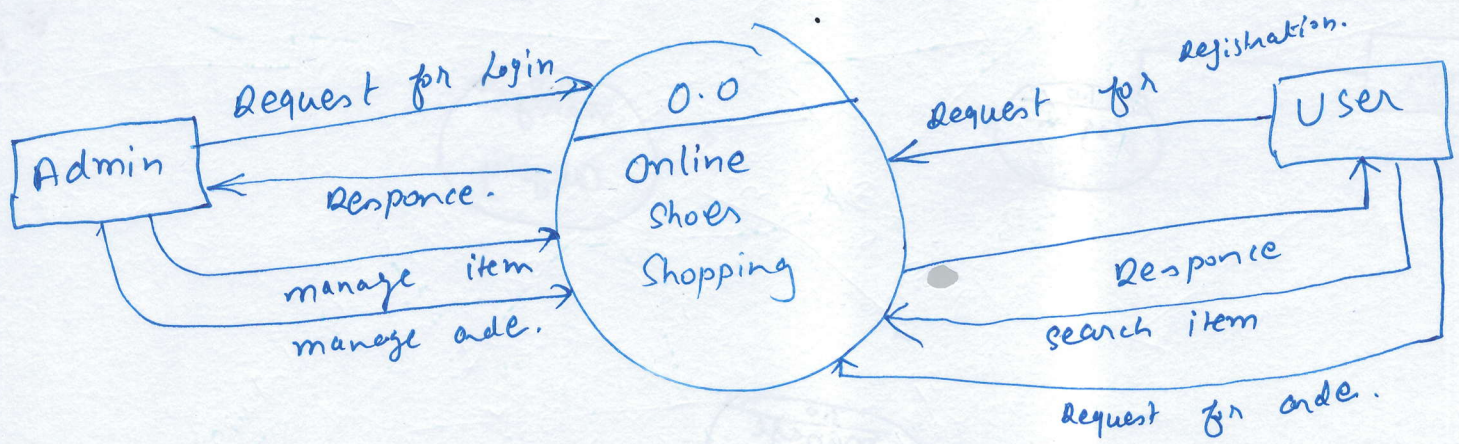
- After identification of major processes in first level DFD, an analyst has to identify processes from first level DFD which can be further expanded.
- Analyst then draws DFD of a particular process identified with expansion. This is known as exploded DFD or Expanded DFD.

Level 2 DFD

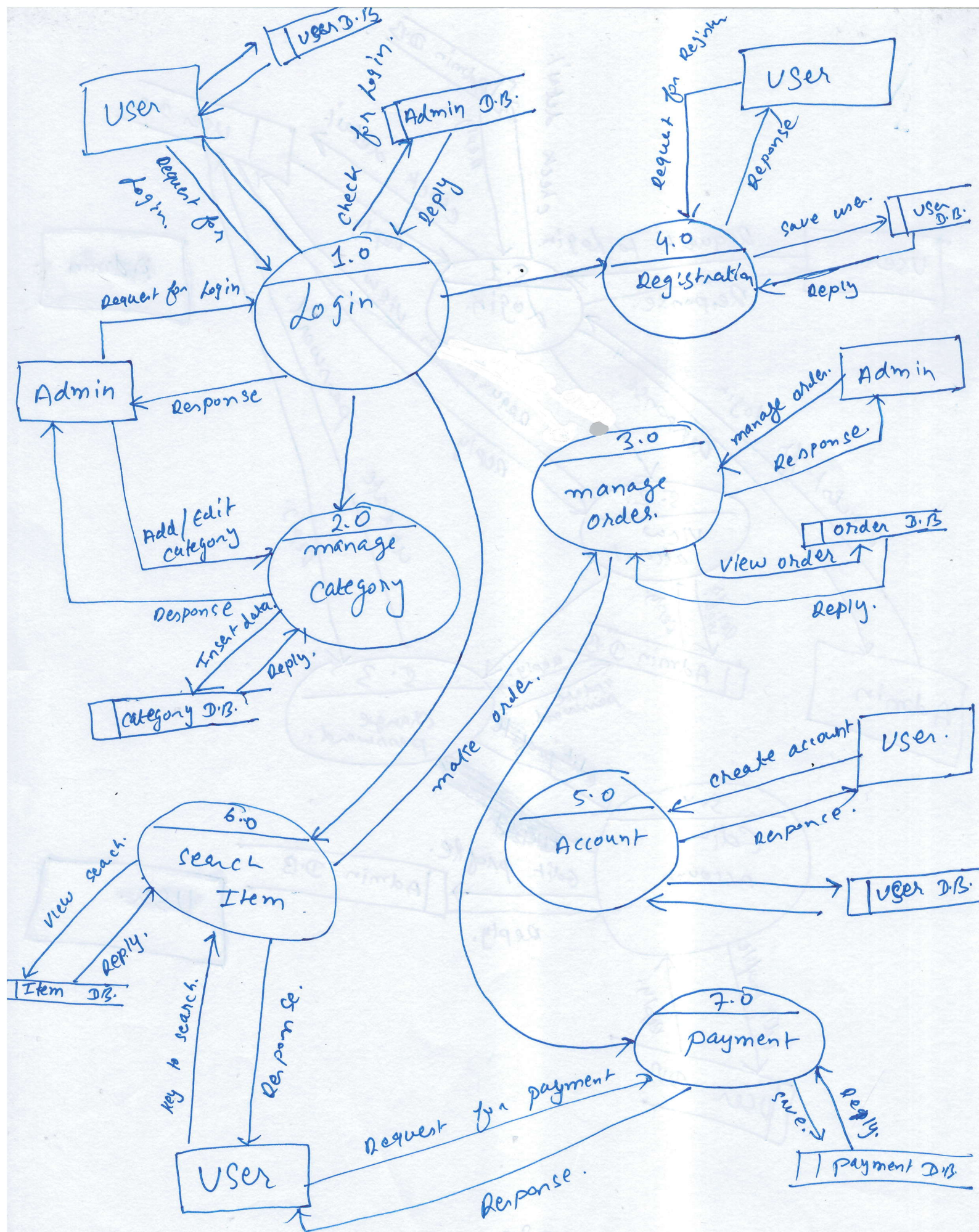


GENERAL RULES FOR DRAWING DFD

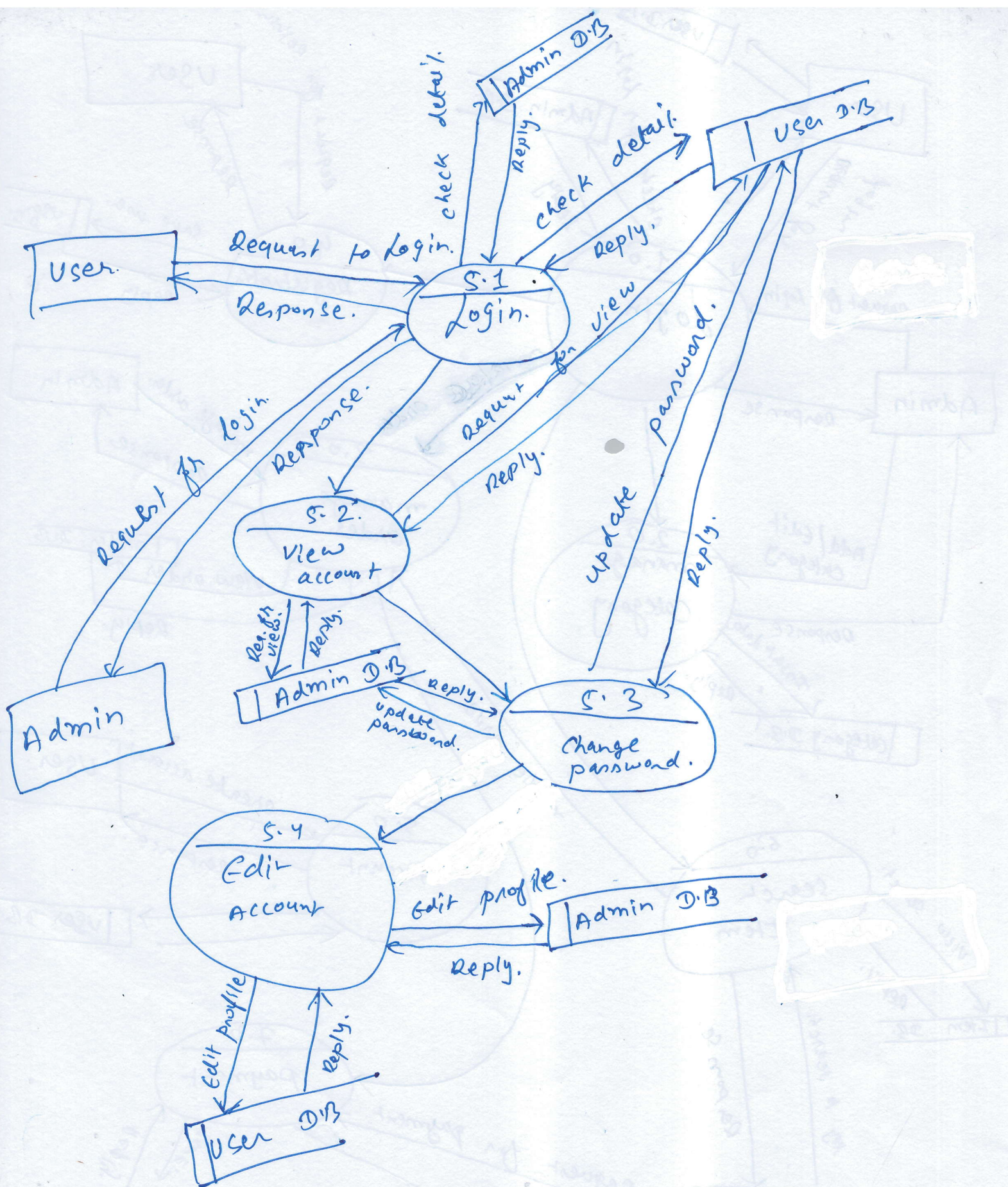
- Processes should be uniquely named and numbered
- Data flows within the system must be named
- Context DFD must fit on the single page
- Process name in the context DFD should be the name of the system
- Do not cross the lines



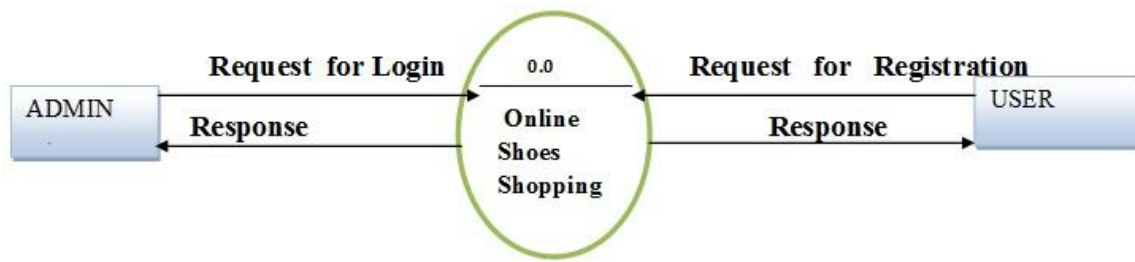
Level 0 DFD.



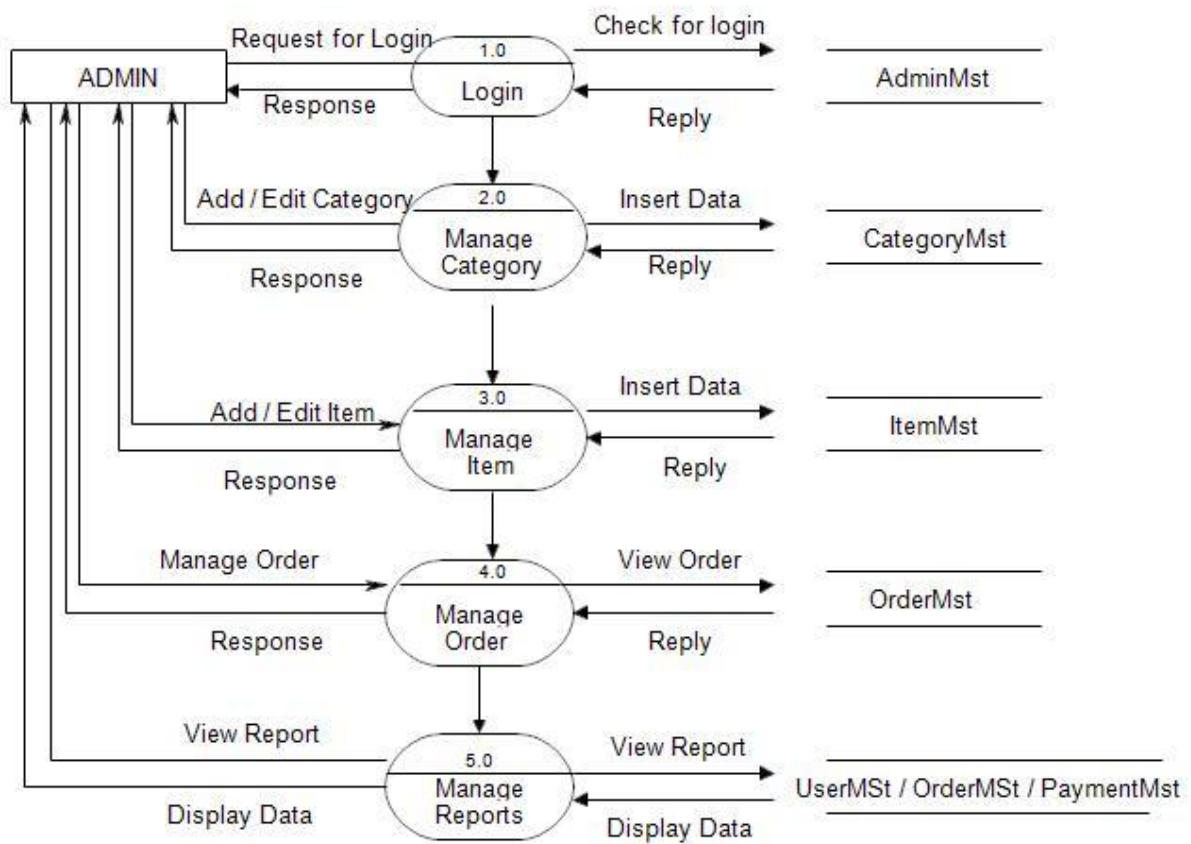
1st Level DFD



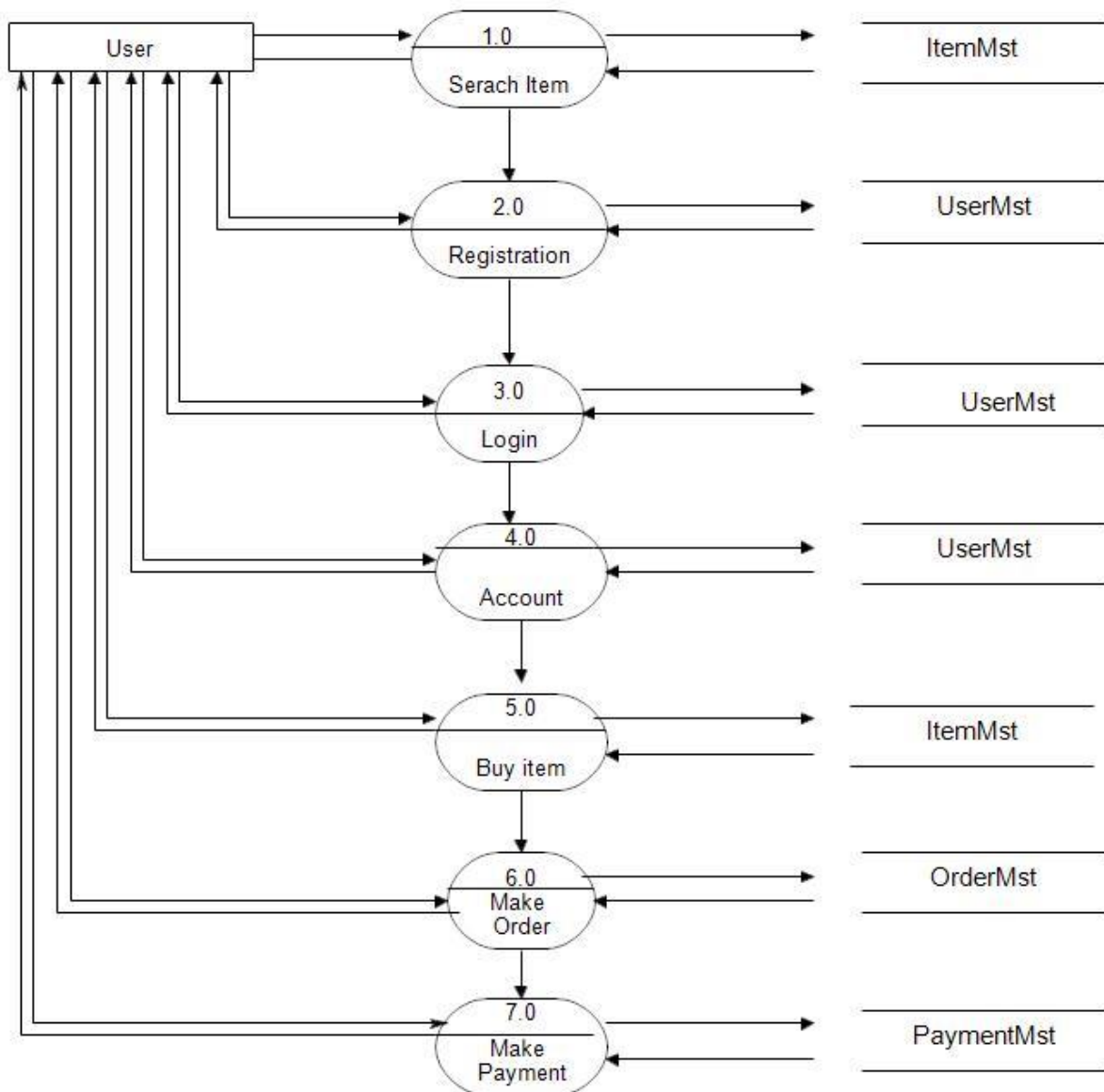
2nd level DFD



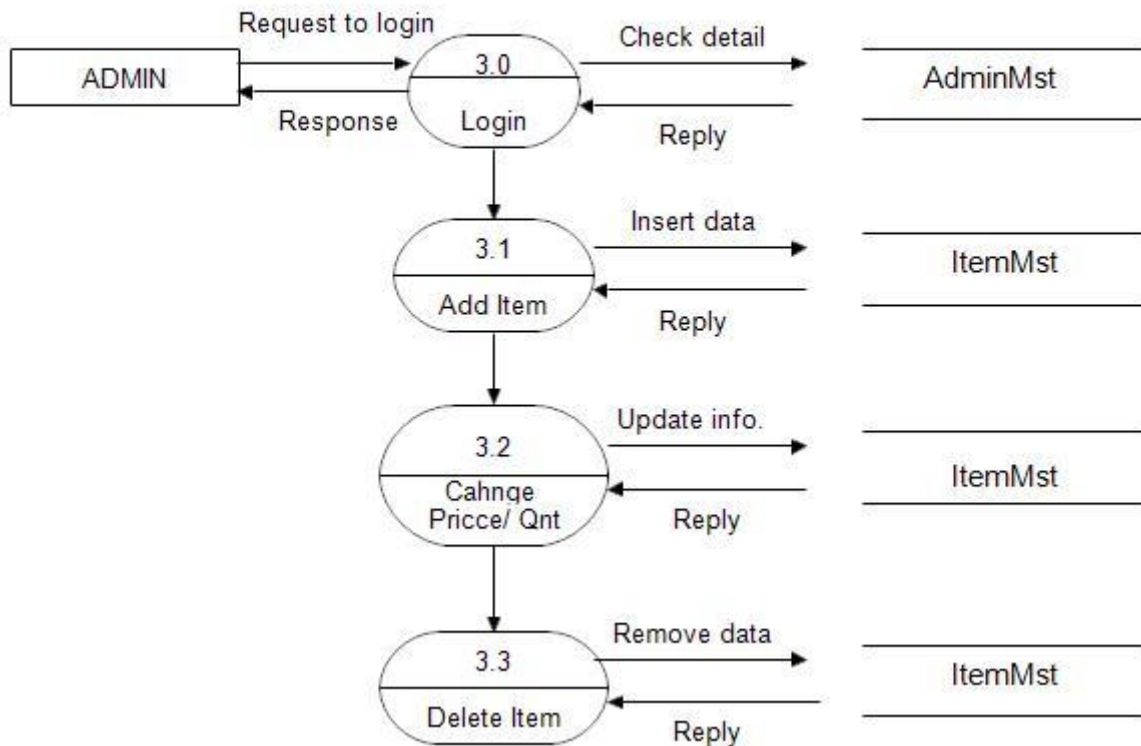
Admin Side DFD - 1st Level



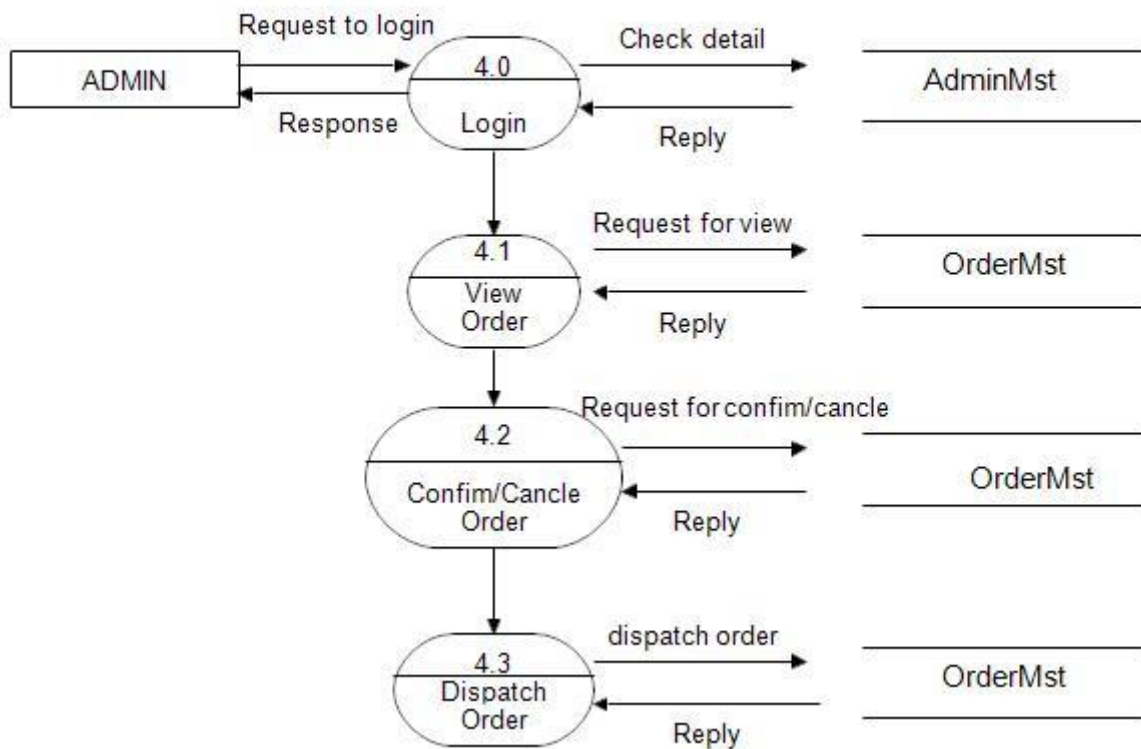
1st Level User side DFD



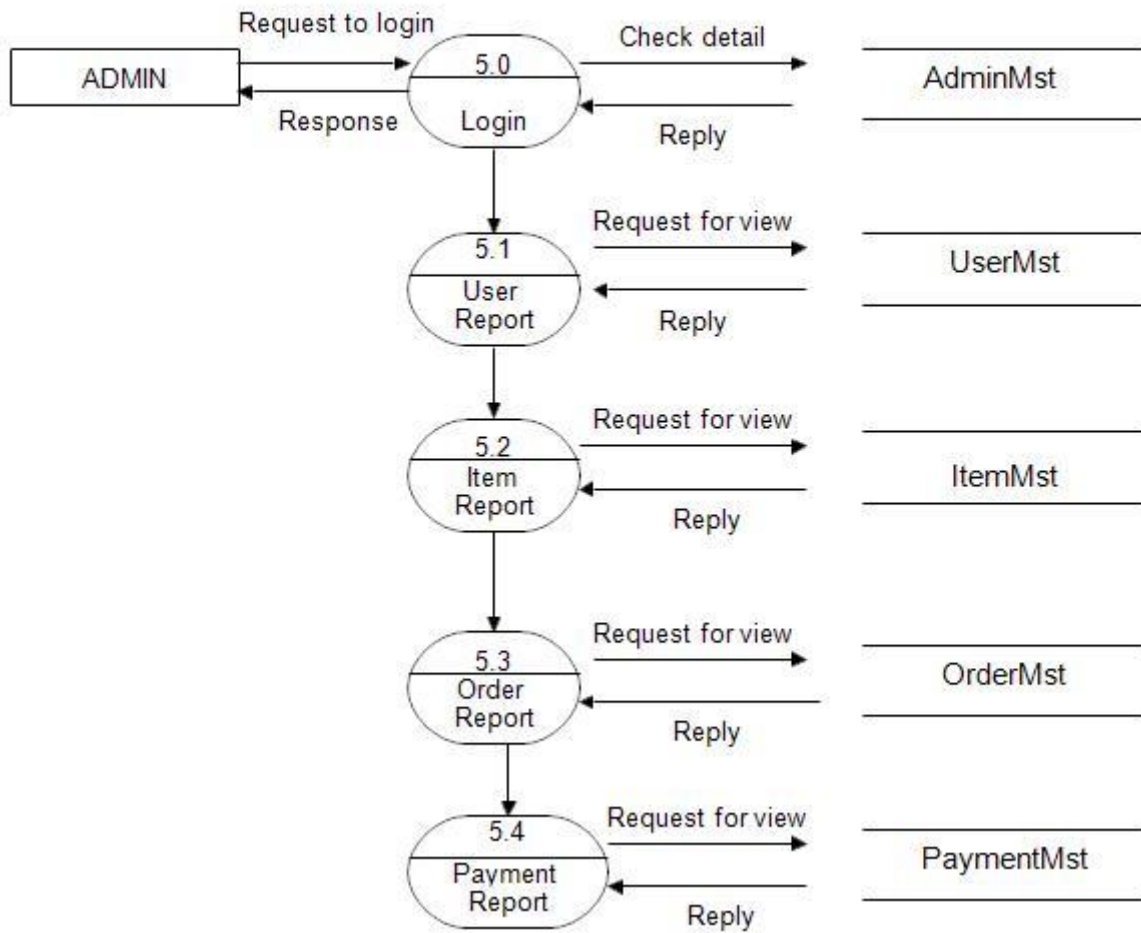
2nd Level Admin DFD - (3.0)



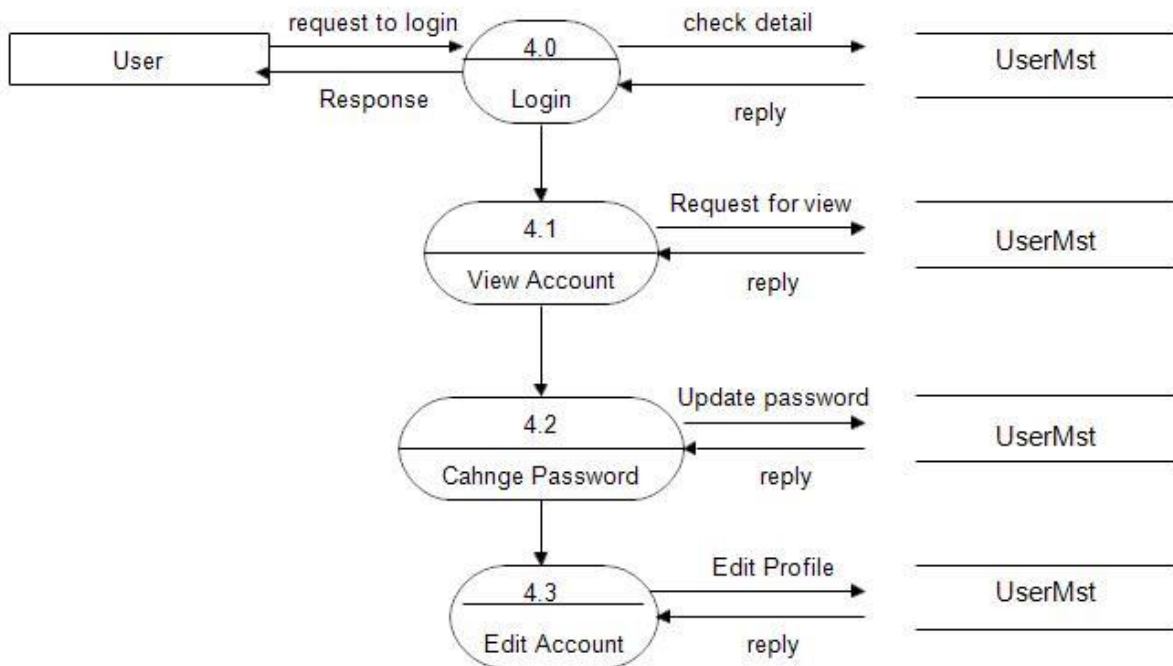
2nd Level Admin DFD - (4.0)



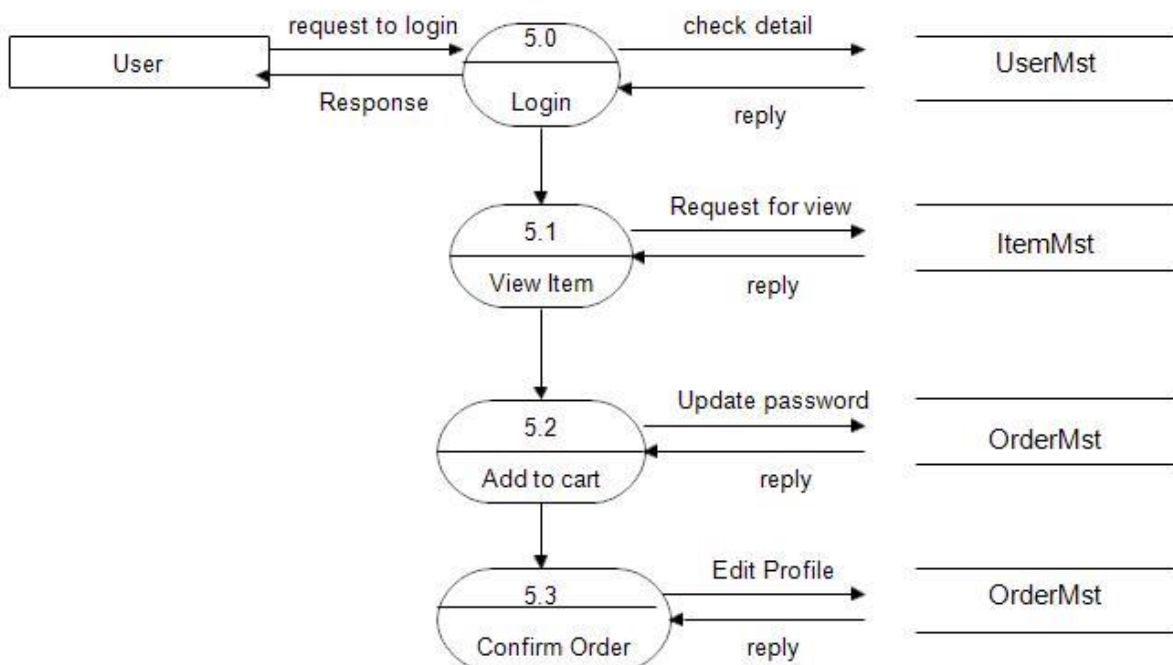
2nd Level Admin DFD - (5.0)



2st Level User DFD - (4.0)



2st Level User DFD - (5.0)



Issues in Project Management

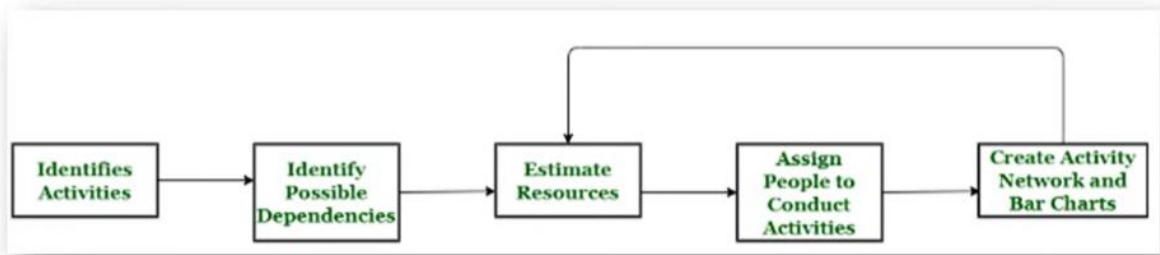
- Project is not completed on schedule.
- Changing customer requirements affect on schedule.
- Technical difficulties are generate.
- Miscommunication among project management.
- Essential software & hardware may be delivered late.
- In large project, Software engineer perform multiple tasks parallel.
- Tasks interdependencies are in project.
- Risk is not considered at beginning of project.

About Project Scheduling

- Project scheduling is responsible activity of project manager.
- Project schedule is a mechanism that is used to communicate and know about that tasks are needed and has to be performed in project.
- Project manager separate total work task in project into different activities. I.e. WBS
- Project Manager Estimate time & recourses required to complete activities & organize them into coherent sequence.
- Effective project scheduling leads to success of project, reduced cost and increased customer satisfaction.

Project Scheduling Process

1. Identify all the functions / modules required to complete the project.
2. Break down large functions into small activities i.e. Develop WBS structure.
3. Determine the dependency among various activities.
4. Allocate resources to activities.
5. Assign people to conduct different activities.
6. Plan the beginning and ending dates for different activities.
7. Create Activity Network & Bar or Gantt charts.



Basic Principles of Project Scheduling

1. **Compartmentalization:** Project divide into number of manageable activities & tasks.
2. **Interdependency:** Certain tasks occur in sequence whereas other tasks occur in parallel.
3. **Time Allocation:** Each task has to be assigned specific time period i.e a start date & a completion date.
4. **Effort Validation:** PM ensure that allocated number of people work on given task.
5. **Defined Responsibilities:** Each task is assigned to specific member of the software team.
6. **Defined Outcomes:** Each task has a defined outcome. I.e. Work product
7. **Defined Milestones:** Every task should be associated with a project milestone. A milestone is accomplished when one or more work products has been reviewed for quality.

Project Scheduling Techniques

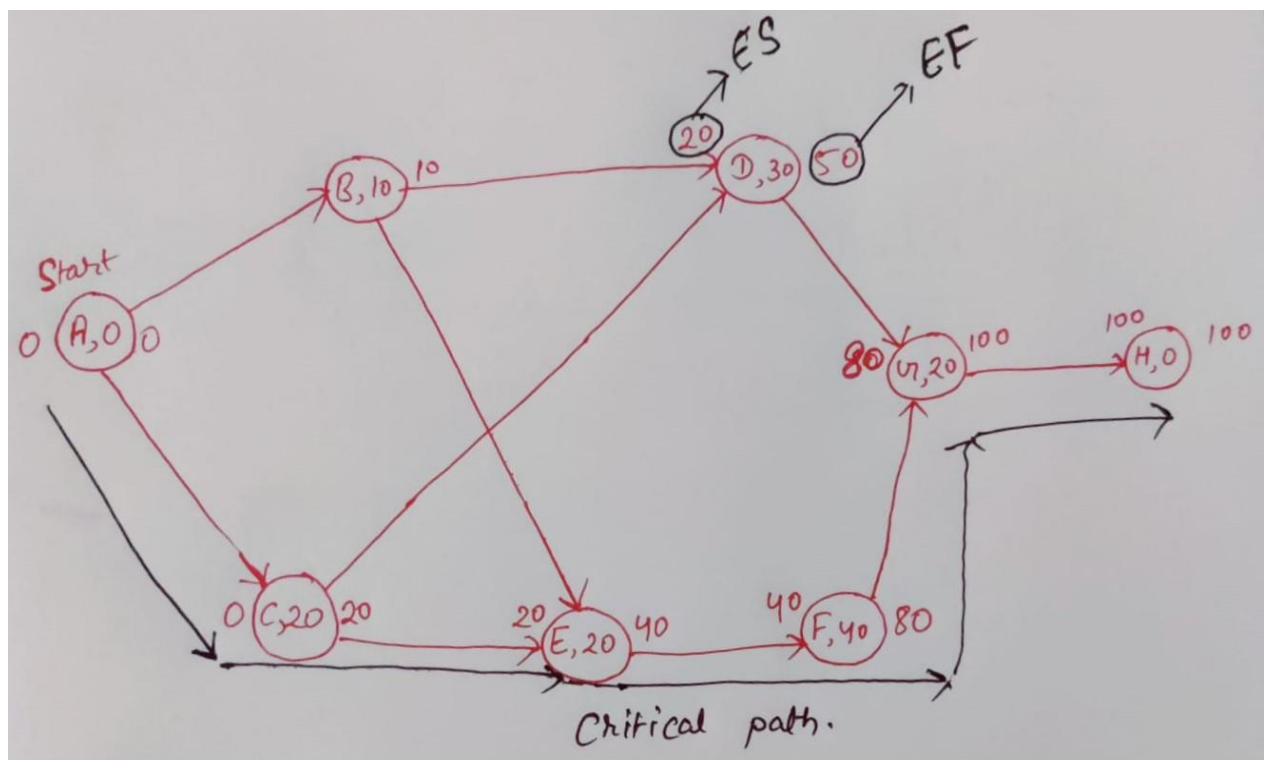
1. Critical Path Method:
2. Program evaluation and review technique(PERT)
3. Gantt Chart
4. Task list
5. Fast tracking
6. Crashing

1. Critical Path Method:

- It help you determine both longest and shortest possible time take to complete a project.
- There are three essential elements: The tasks required to complete the project, Which tasks depend on the completion of others, A time estimate for each activity
- Critical path method is a diagram which shows the sequence of activities of a project along with their **ES** and **EF**.
- Whereas **ES** is the Earliest Start and **EF** is the Earliest Finish time of an activity in the critical path method Diagram.
- We develop this diagram to find out the critical path of project.
- Critical path is the flow of the activities in the diagram which take the most time in a project.
- To find out that Critical path we have to make a complete diagram

- We should have all the activities their dependence list and their time duration to make this diagram.

Task ID	Task Description	Task Predecessors	Task Duration (hours)
A	Project start		0
B	Buy materials for A	A	10
C	Buy materials for B	A	20
D	Build A	B, C	30
E	Build B	B, C	20
F	Polish and finish B	E	40
G	Join A and B	D, F	20
H	Project finish	G	0



Path 1:

A → B → D → G → H = 60 Hours

Path 2:

A → B → E → F → G → H = 90 hours

Path 3:

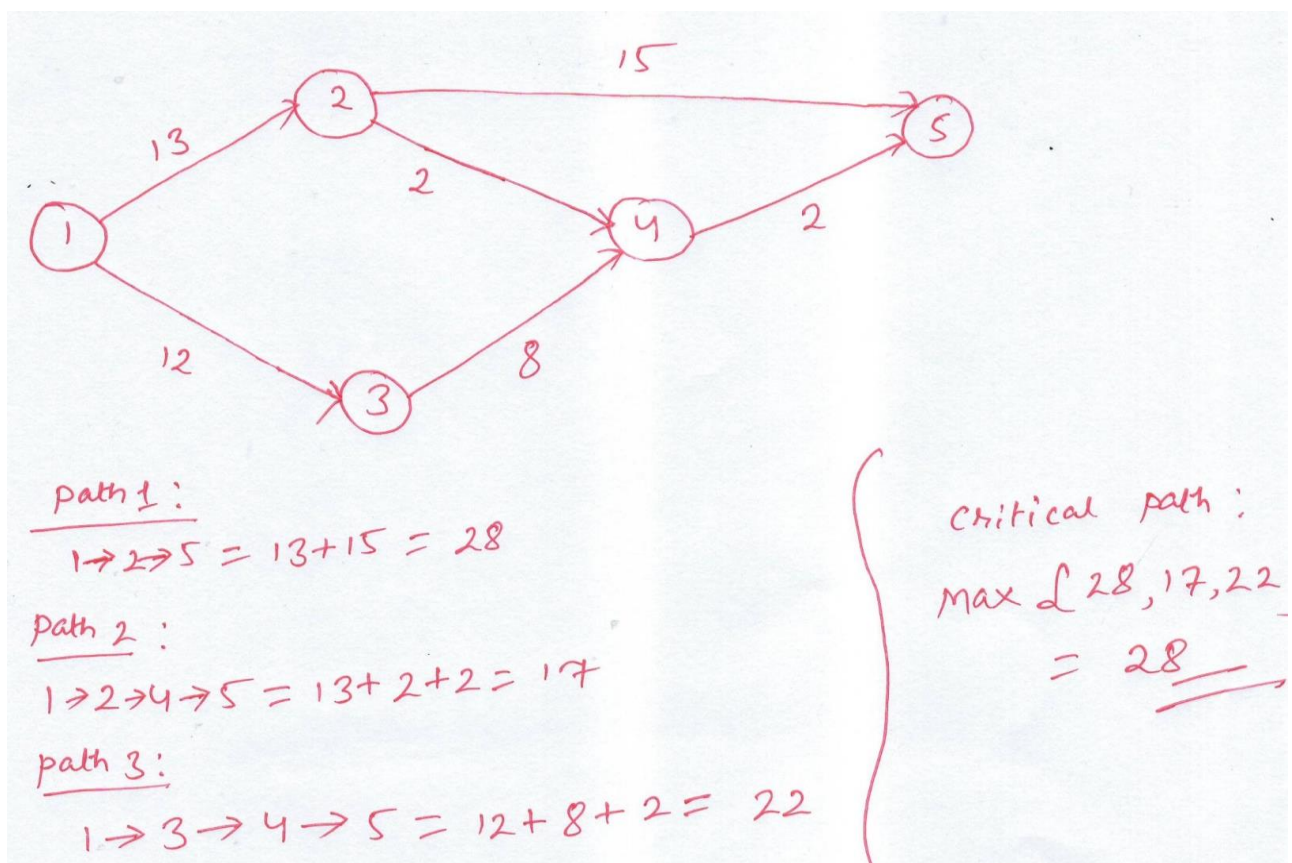
A → C → D → G → H = 70 Hours

Path 4:

A → C → E → F → G → H = 100 hours

Another Example

Activity	Duration
1-2	13
1-3	12
2-4	2
3-4	8
2-5	15
4-5	2



2. Program evaluation and review technique(PERT):

- PERT is a project scheduling technique.
- It is a way to schedule flow of tasks in a project and estimate total time taken to complete it.
- PERT charts offer a visual representation of the major activities (and dependencies) in a project

It calculate:

- **Optimistic time (O)/ (To):** Quickest time you can complete a project in favourable condition.
- **Pessimistic time (P)/ (Tp):** Longest time it'll take to complete your project in unfavourable condition.
- **Most likely time (M)/ (Tm):** How long it'll take to finish your project if there are no problems or in normal condition. Also known as **realistic time (Tr)**.

- $T_e = \frac{(1T_o + 4T_m + 1T_p)}{6}$

6

Example:

PERT

Activity	To	Tm	Tp	$T_e = (T_o + 4T_m + T_p)/6$
1-2	3	6	9	
1-6	2	5	8	
2-3	6	12	18	
2-4	4	5	6	
3-5	8	11	14	
4-5	3	7	11	
6-7	3	9	15	
5-8	2	4	6	
7-8	8	16	18	

3. Gantt Chart:

- A Gantt chart is a type of bar graph that project managers use for planning and scheduling in complex project.
- An activity in which you decide how you will divide the project into the smaller tasks, how it will be executed and how much time a task will take to complete.
- It represent each task horizontally on a bar chart, which shows the start and end dates & they frequently include deadlines & dependencies of tasks.
- It easier to visualize the progress of a project and see how different tasks interact with one another.

Things to keep in mind while making Gantt chart

Step 1: Set Milestone

- Here you set the milestones.
- May be your 1st milestone will be consist on first two tasks if your projects is divided into so many tasks.

What is Milestone?

It can be any like your software is consist of three functions the first function is consist on 1st two tasks so you will achieve it after completing first two tasks.

Step 2: Estimate resources

- If your Project is divided into 6 tasks let suppose so you should have knowledge what kind of and how much recourses are required to complete specific task.
- **For Example :** To complete Task 1 I need 5 GB of storage as it will be consist on High graphics and Movie clips so I should check If I have enough space for task 1 or not.

Key Points

- Some tasks will go on in parallel
- Sometime things go wrong don't make schedule too much tight
- People working on the project may fall ill
- Make estimate as nothing will go wrong and then add few more days to the final estimation as if goes something wrong so we have at least some extra days.

Schedule Representation

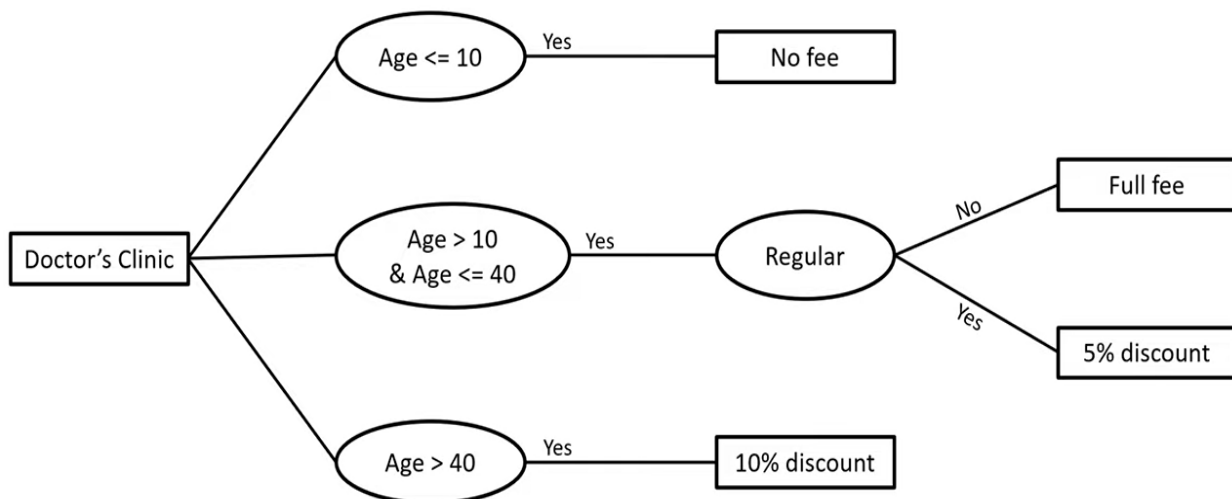
Task	Duration (Days)	Dependencies
T1	14	
T2	7	T1
T3	20	(M1)
T4	6	T3
T5	27	T1 (M2)
T6	7	
T7	10	(M3)

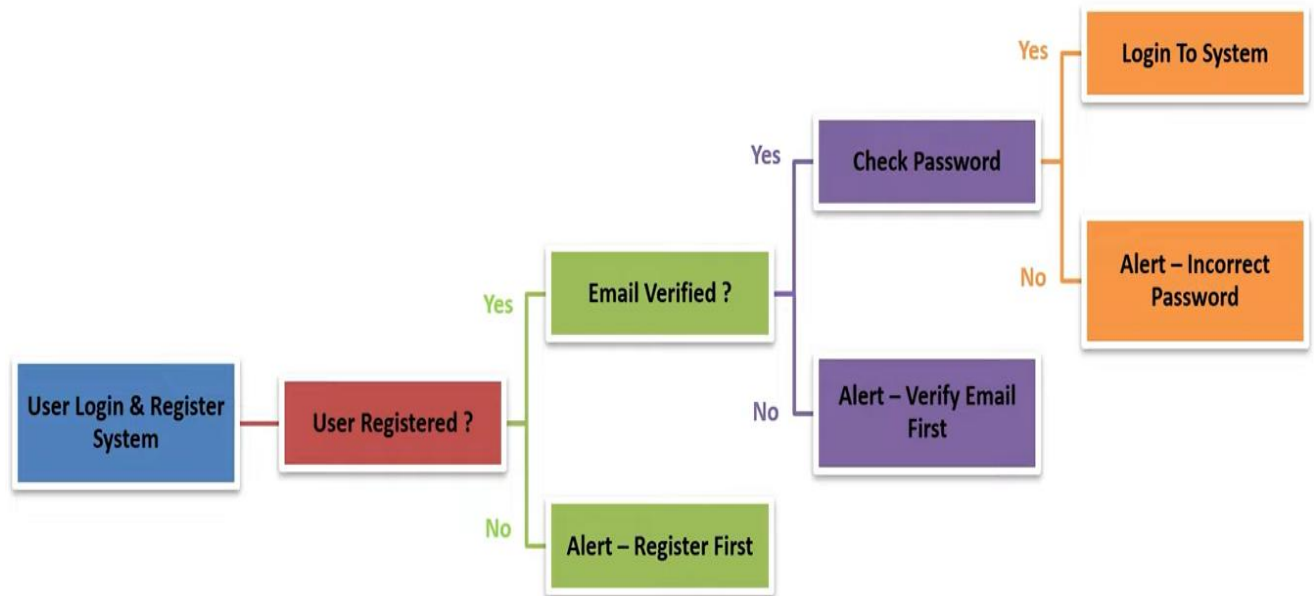
Activity Bar Chart Weekly basis

Week	1	2	3	4	5	6
T1						
T2			T2			
T3	T2(M1)					
T4				T3		
T5			(M2)			
T6						
T7					(M3)	

Decision Tree

- Decision Tree is a graph which uses a branching method in order to display all the possible outcomes of any decision.
- Decision tree is a pictorial representation of decision point and corresponding action based upon some business rule.
- It helps in processing logic involved in decision making and corresponding actions taken.
- It is a diagram that shows conditions and their alternative actions within horizontal tree framework.
- It helps analyst to consider the sequence of decisions and identifies the accurate decision that must be made.





Decision Table

- Decision Table is just a tabular representation of all conditions and actions.
- It is a tabular representation of a decision points and all the action related to those decision in a system.
- It is used for describing the complex problem in a precise manner which is easily understandable.
- We can derive decision table from decision tree.
- It helps testers to search the effects of combinations of different inputs and other software states that must correctly implement business rules.

Components of Decision Table

- ❑ **Condition stub:** It is the upper left quadrant which lists all the condition to be checked.
- ❑ **Action Stub:** It is the lower left quadrant which lists all the action to be carried out to meet such condition.

❑ **Condition Entry:** It is the upper right quadrant which provides answers to questions asked in condition stub quadrant.

❑ **Action Entry:** It is the lower right quadrant which provides the appropriate action resulting from the answers to the conditions in the condition entry quadrant.

Condition stub	Condition entry
Action stub	Action entry

User Login & Register System

Condition	Rule 1	Rule 2	Rule 3	Rule 4
Username	F	T	F	T
Password	F	F	T	T
Action	-	-	-	-
Output	Error	Error	Error	Login

Doctor's Clinic

Condition	Rule 1	Rule 2	Rule 3	Rule 4
Age<=10	Yes	-	-	-
Age>10 & Age<=40	-	-	Yes	Yes
Age>40	-	Yes	-	-
Regular	-	-	Yes	No
Action				
Fee	No	Yes	Yes	Yes
Discount	-	10%	5%	No

Project management and its phase

➤ Project management is the process of planning, organizing, and controlling resources to achieve specific goals. It is typically used to achieve a specific goal within a specific timeframe and within budget constraints. Project management involves several phases, including:

1. **Initiation:** This phase involves defining the project and obtaining approval to start it. The project charter and project plan are developed during this phase.
2. **Planning:** During this phase, the project team develops a detailed plan that outlines the tasks and resources needed to complete the project. The project schedule and budget are also developed during this phase.
3. **Execution:** This phase involves carrying out the project according to the plan. The project team works on the tasks and deliverables outlined in the project plan.
4. **Monitoring and controlling:** During this phase, the project team monitors progress and makes adjustments as necessary to keep the project on track. This may involve revising the project plan or budget.
5. **Closing:** This phase involves completing all the tasks and deliverables outlined in the project plan, and formally closing the project.

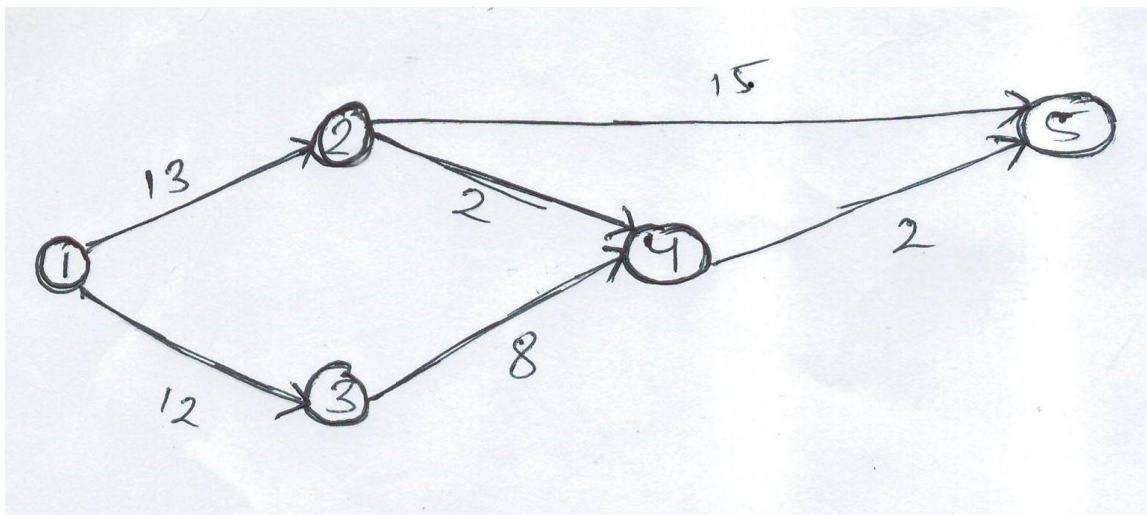
Each phase of the project management process involves specific tasks and activities that are necessary for the successful completion of the project.

Network diagram

- A network diagram is a graphical representation of a project's schedule that shows the dependencies between the project's tasks.
- It is a useful tool for planning and scheduling a project, as it helps to clearly visualize the relationships between tasks and the order in which they must be completed.
- In a network diagram, tasks are represented by boxes or circles, and the dependencies between tasks are represented by arrows.
- The length of the arrow indicates the duration of the task.
- The network diagram can also show the resources required for each task and the critical path of the project, which is the series of tasks that must be completed on schedule in order for the project to be completed on time.

Example:

Activity	Duration
1-2	13
1-3	12
2-4	2
3-4	8
2-5	15
4-5	2



Path 1: 1 to 2 to 5 = $13+15=28$ Hours

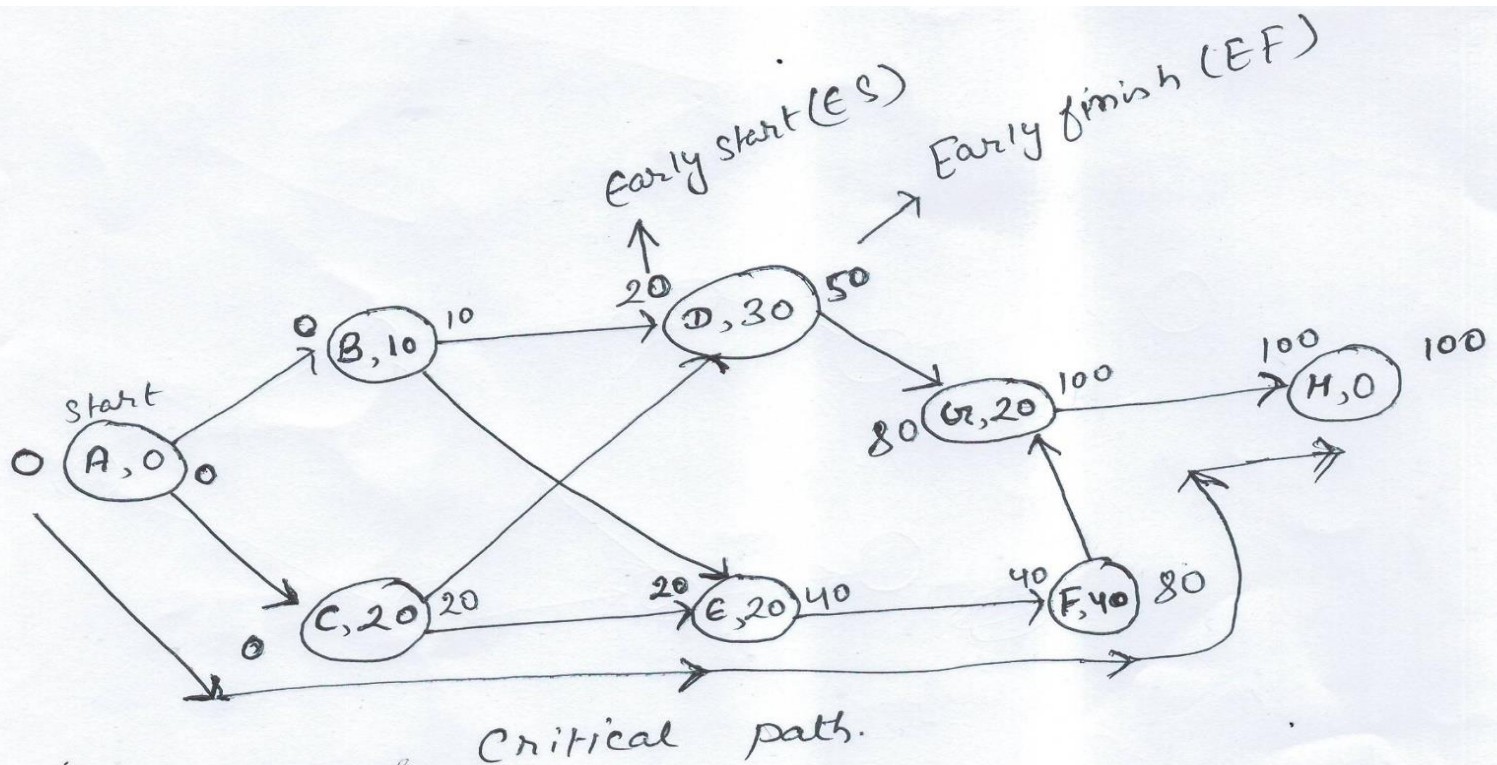
Path 2: 1 to 2 to 4 to 5 = $13+2+2=17$ Hours

Path 3: 1 to 3 to 4 to 5 = $12+8+2=22$ Hours

Critical Path: $\text{MAX}\{28, 17, 22\} = 28$ hours

Example 2:

Task ID	Task Predecessors	Task Duration(Hours)
A	-	0
B	A	10
C	A	20
D	B,C	30
E	B,C	20
F	E	40
G	D,F	20
H	G	0



PERT (Program Evaluation Review Technique)

- PERT (Program Evaluation Review Technique) is a project management tool used to estimate the duration of a project by analyzing the tasks involved in the project and the dependencies between those tasks.
- By using this information, the project manager can estimate the minimum amount of time required to complete the project.
- It is developed by the U.S. Navy in the 1950s to manage the Polaris submarine missile program.
- It is based on the assumption that the duration of a task is represented by a probability distribution, rather than a single fixed value.
- To calculate the expected time duration of a task using PERT, you will need to know the optimistic time, the most likely time, and the pessimistic time for the task.

The expected time duration is then calculated using the following formula:

$$\text{Expected time} = \frac{(\text{Optimistic time} + 4 * \text{Most likely time} + \text{Pessimistic time})}{6}$$

For example, if the optimistic time for a task is 2 days, the most likely time is 4 days, and the pessimistic time is 6 days, the expected time for the task would be calculated as follows:

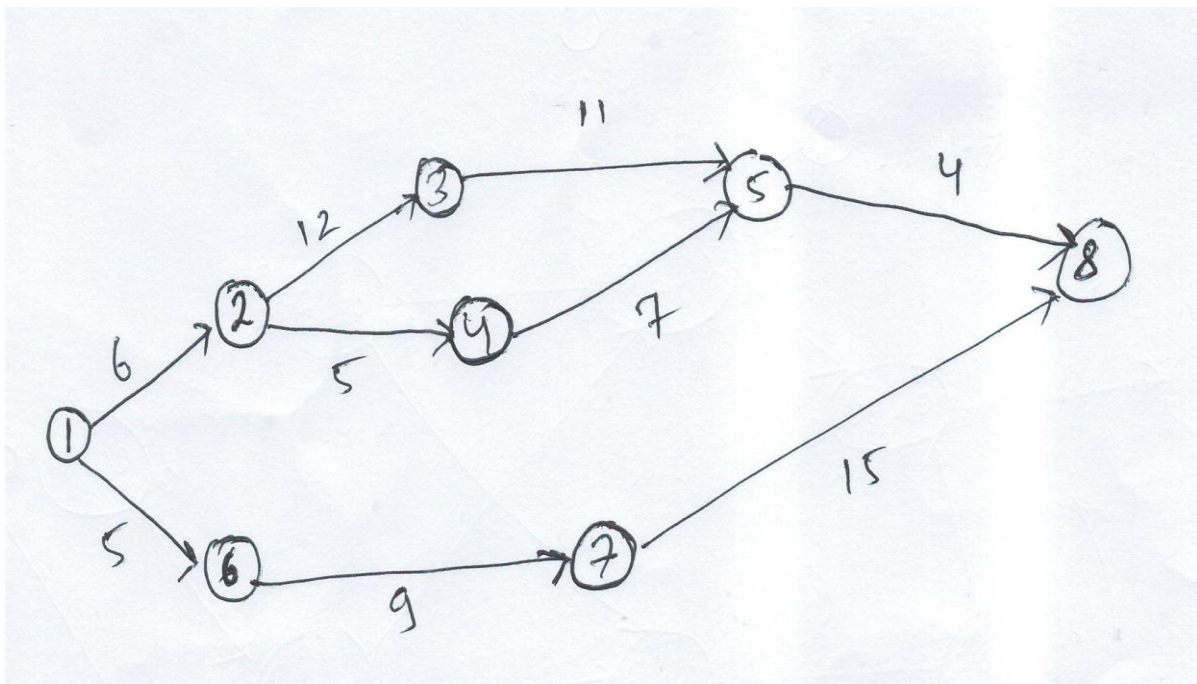
$$\text{Expected time} = (2 + 4 * 4 + 6) / 6 = 4 \text{ days}$$

- PERT is a useful tool for project management because it helps to account for uncertainty and allows for more accurate estimates of project duration. It is particularly useful in complex projects with many tasks and dependencies.

Example

Activity	Optimistic time(TO)	Most likely time(TM)	Pessimistic time(TP)	Expected time
1-2	3	6	9	?
1-6	2	5	8	?
2-3	6	12	18	?
2-4	4	5	6	?
3-5	8	11	14	?
4-5	3	7	11	?
6-7	3	9	15	?
5-8	2	4	6	?
7-8	8	16	18	?

1. Calculate time estimate for each activity.
2. Identify the critical path



Decision table/Matrix

Library policy/rules

1. A teacher can take book for full semester.
2. A Student can take book for three month.
3. A visitors can take book for 1 week after submitting a security deposit.
4. If a book is not available, then teacher and student can reserve book.
5. If there is delay, then student is fined RS. 100
6. If there is delay, then visitor is fined RS. 200
7. The visitor's security is refunded on return of book.

CONDITIONS	CONDITION ENTRY									
TEACHER	Y	Y	Y	N	N	N	N	N	N	N
STUDENT	N	N	N	Y	Y	Y	Y	N	N	N
VISITORS	N	N	N	N	N	N	N	Y	Y	Y
ISSUE	Y	Y	N	Y	Y	N	N	Y	N	N
AVAILABLE	Y	N	-	Y	N	-	-	Y	-	-
RETURN	N	N	Y	N	N	Y	Y	N	Y	Y
DELAY	-	-	-	-	-	N	y	-	Y	N
ACTIONS	ACTION ENTRY									
1 WEEK								✓		
3 MONTH				✓						
FULL SEMESTER	✓									
RESERVE BOOK		✓			✓					
PAY 100 RS FINE							✓			
PAY 200 RS FINE									✓	
DEPOSITE SECURITY								✓		
REFUND SECURITY									✓	✓
RETURN BOOK			✓			✓	✓		✓	✓

Cost-Benefit Analysis

- It is a part of economic feasibility study.
- It is an analysis for estimating cost involved in the development of a project and the possible profit to be derived from it.
- It works on a simple rule, i.e., identifying the benefits from the proposal and associated costs, and subtracting the costs from benefits.
- It helps us to assign value to a proposal, calculate the future cash flow and measure the financial worthiness of it.

How to Perform Cost-benefit Analysis

- Identify costs and benefits from the system
- Assign Values to the costs and benefits
- Determine the total cash flow
- Determine the project's Economic value

- ❖ Return on Investment (ROI)

$$\text{ROI} = (\text{Total Benefits} - \text{Total Costs}) / \text{Total Costs} * 100$$

- ❖ Break-Even Point (BEP)
- ❖ Net Present Value (NPV)

COST-BENEFIT ANALYSIS

- Cost-benefit analysis can be explained as a procedure for estimating all costs involved and possible profits to be derived from a business opportunity or proposal.
- Cost-benefit analysis determined by identifying costs and benefits associated with the system, assigning values to them, calculating future cash flows, and measuring the financial worthiness of the project.
- As a result of this analysis, the financial opportunities and risks of the project can be understood.
- Keep in mind that organizations have limited capital resources and multiple projects will be competing for funding.
- The more expensive the project, the more rigorous and detailed the analysis should be.

The steps involved in performing cost-benefit analysis are as follows:

1. Identify Costs and Benefits
2. Assign Values to Costs and Benefits
3. Determine Cash Flow
4. Assess Project's Economic Value
 - A. Return on Investment (ROI)
 - B. Break-Even Point (BEP)
 - C. Net Present Value (NPV)

1. Identify Costs and Benefits

- The systems analyst's first task when developing cost-benefit analysis is to identify the kinds of costs and benefits the system will have and list them.

➤ The costs and benefits can be broken down into four categories:

1. Development Costs
2. Operational Costs
3. Tangible Benefits
4. Intangibles Benefits

1. Development Costs

Development costs are those tangible expenses that are incurred during the creation of the system, such as salaries for the project team, hardware and software expenses, consultant fees, training, and office space and equipment.

2. Operational Costs

Operational costs are those tangible costs that are required to operate the system, such as the salaries for operations staff, software licensing fees, equipment upgrades, and communications charges. Operational costs are usually thought of as ongoing costs.

3. Tangible Benefits

Tangible benefits include revenue that the system enables the organization to collect, such as increased sales. In addition, the system may enable the organization to avoid certain costs, leading to another type of tangible benefit: cost savings. For example, if the system produces a reduction in needed staff, lower salary costs result. Similarly, a reduction in required inventory levels due to the new system produces lower inventory costs.

4. Intangibles Benefits

Of course, a project also can affect the organization's bottom line by reaping intangible benefits or incurring intangible costs. Intangible costs and benefits are more difficult to incorporate into the cost-benefit analysis because they are based on intuition and belief rather than on "hard numbers." Nonetheless, they should be with the tangible items.

2. Assign Values to Costs and Benefits

- Once the types of costs and benefits have been identified, the analyst needs to assign specific currency (dollar/rupee etc.) values to them.
- The most effective strategy for estimating costs and benefits is to rely on the people who have the best understanding of them.
- The company also can consider past projects, industry reports, and vendor information, although these sources probably will be a bit less accurate. Likely, all of the estimates will be revised as the project proceeds.
- If predicting a specific value for a cost or benefit is proving difficult, it may be useful to estimate a range of values for the cost or benefit and then assign a likelihood (probability) estimate to each value.
- What about the intangible benefits and costs? Sometimes, it is acceptable to list intangible benefits, such as improved customer service, without assigning a dollar value. Other times, estimates have to be made regarding how much an intangible benefit is "worth."

3. Determine Cash Flow

- A formal cost—benefit analysis usually contains costs and benefits over a selected number of years (usually, three to five years) to show cash flow over time.

- For example, Figure below lists the same amount for customer complaint calls, inventory costs, hardware, and software for four years.

	Year 0	Year 1	Year 2	Year 3	Total
Total Benefits		45,000	50,000	57,000	152,000
Total Costs	100,000	10,000	12,000	16,000	138,000
Net Benefits (Total Benefits – Total Costs)	(100,000)	35,000	38,000	41,000	14,000
Cumulative Net Cash Flow	(100,000)	(65,000)	(27,000)	14,000	

4. Assess Project's Economic Value

- Evaluate the project's expected returns in comparison to its costs. Use one or more of the following evaluation techniques:
 - A. Return on Investment (ROI)
 - B. Break-Even Point (BEP)
 - C. Net Present Value (NPV)

A. Return on Investment (ROI)

- The return on investment (ROI) is a calculation that measures the average rate of return earned on the money invested in the project. ROI is a simple calculation that divides the project's net benefits by the total costs. The ROI formula is:

$$\text{ROI} = \text{Total Benefits} - \text{Total Costs} / \text{Total Costs} * 100$$

A high ROI suggests that the project's benefits far outweigh the project's cost.

B. Break-Even Point (BEP)

- Another common approach to measuring a project's worth is the break-even point.
- The break-even point (also called the payback method) is defined as the number of years it takes a firm to recover its original investment in the project from net cash flows.

$$\text{BEP} = \text{No. of years of negative cash flow} + \frac{\text{That year's net cash flow} - \text{that year's cumulative cash flow}}{\text{that year's net cash flow}}$$

C. Net Present Value (NPV)

- Net Present Value (NPV) The NPV is simply the difference between the total present value of the benefits and the total present value of the costs.

$$\text{NPV} = \Sigma \text{PV of Total Benefits} - \Sigma \text{PV of Total Costs}$$

- As long as the NPV is greater than zero, the project is considered economically acceptable.

Economic feasibility analysis for an Information System (1)

Consider the given data:

Monetary Benefits of IS = \$85,000 per year.

One-time costs = \$75,000

Recurring costs = \$35,000 per year.

Discount rate = 12%.

Time Period = 5 years.

The worksheet given below (figure) shows the Present Value calculations of all costs and benefits, Break-Even Analysis, Overall Return on Investment for this problem using the above data.

The spreadsheet displays the following items:

1. Net Present Values (NPV) of all Benefits and Costs
2. Overall Return On Investment (ROI)
3. Break-Even Analysis (BEA)

1. Net present value of Benefits and Costs:

Present Value of Benefits or Costs can be calculated using the below formula:

$$PV_n = Y \times \frac{1}{(1+i)^n}$$

Here, PV_n is the present value of Y dollars n years from now, and i is the discount rate.

Present Value (PV) calculations for Benefits:

Benefits start from year 1, so the calculation of PV from year 1 onwards.

$$PV_1 = 85,000 \times \frac{1}{(1+.12)^1} = 85,000 \times 0.8929 = 75,897$$

$$PV_2 = 85,000 \times \frac{1}{(1+.12)^2} = 85,000 \times 0.7972 = 67,762$$

$$PV_3 = 85,000 \times \frac{1}{(1+.12)^3} = 85,000 \times 0.7118 = 60,503$$

$$PV_4 = 85,000 \times \frac{1}{(1+.12)^4} = 85,000 \times 0.6355 = 54,018$$

$$PV_5 = 85,000 \times \frac{1}{(1+.12)^5} = 85,000 \times 0.5674 = 48,229$$

Net Present Value of Benefits: The net value of benefit will be the sum of overall benefits and will be calculated using following formulae:

$$\begin{aligned} NPV &= PV_1 + PV_2 + PV_3 + PV_4 + PV_5 \\ &= 75,897 + 67,762 + 60,503 + 54,018 + 48,229 \\ &= 306,408 \text{ (Cell H8)} \end{aligned}$$

Present Value (PV) calculations for Costs: Here, the one-time cost (\$75,000) is treated as cost occurring in year 0 (now).

$$PV_0 = 75,000 \times \frac{1}{(1+.12)^0} = 75,000 \times 1 = 75,000$$

Recurring cost (\$35,000) happens every year starting at year 1.

$$PV_1 = 35,000 \times \frac{1}{(1+.12)^1} = 35,000 \times 0.8929 = 31,252$$

$$PV_2 = 35,000 \times \frac{1}{(1+.12)^2} = 35,000 \times 0.7972 = 27,902$$

$$PV_3 = 35,000 \times \frac{1}{(1+.12)^3} = 35,000 \times 0.7118 = 24,913$$

$$PV_4 = 35,000 \times \frac{1}{(1+.12)^4} = 35,000 \times 0.6355 = 22,243$$

$$PV_5 = 35,000 \times \frac{1}{(1+.12)^5} = 35,000 \times 0.5674 = 19,859$$

Net Present Value of Costs,

$$\begin{aligned} NPV &= PV_0 + PV_1 + PV_2 + PV_3 + PV_4 + PV_5 \\ &= 75,000 + 31,252 + 27,902 + 24,913 + 22,243 + 19,859 \\ &= 201,168 \text{ (Cell H16)} \end{aligned}$$

2. Overall Return On Investment (ROI)

$$\text{Overall ROI} = (\text{Overall NPV} / \text{NPV of All COSTS})$$

$$\text{Overall NPV} = (\text{NPV of All Benefits} - \text{NPV of All COSTS})$$

$$= 306,408 - 201,168$$

$$= 105,240$$

From the spreadsheet, *Overall NPV* is 105,240 (Cell H18) and

NPV of All COSTS is 201,168 (Cell H16).

Therefore, the overall

$$\text{ROI} = 105,240 / 201,168$$

$$= 0.5231448 \text{ (Cell H20)}$$

3. Break-Even Analysis (BEA)

This analysis is carried out by first determining the NPV of cash flows on yearly basis.

Subtract the present value of recurring cost from the yearly benefits for computing the Yearly NPV Cash Flows. The Overall NPV Cash Flow is the total flow of cash of all the preceding years.

If line 24 of spreadsheet is observed, it would be find that occurrence of break-even happens between year one and two. This is because year two is the very first year in which Overall NPV is non-negative. (Positive)

$$\text{Break - Even Ratio} = \frac{\text{Yearly NPV Cash Flow} - \text{Overall NPV Cash Flow}}{\text{Yearly NPV Cash Flow}}$$

Using the info from below Figure,

$$\text{Break - Even Ratio} = \frac{39,860 - 9,505}{39,860}$$

$$= 0.7615$$

Therefore, Project break-even occurs at approximately 1.8 years (rounded).

	A	B	C	D	E	F	G	H	I
1	Information System1(Economic Feasibility Analysis)								
2		Year of Project							
3		Year 0	Year 1	Year 2	Year 3	Year 4	Year 5	TOTALS	
4	Net economic benefit	\$0	\$85,000	\$85,000	\$85,000	\$85,000	\$85,000		
5	Discount Rate (12%)	1.0000	0.8929	0.7972	0.7118	0.6355	0.5674		
6	PV of Benefits	\$0	\$75,897	\$67,762	\$60,503	\$54,018	\$48,229		
7									
8	NPV of all BENEFITS	\$0	\$75,897	\$143,659	\$204,162	\$258,179	\$306,408	\$306,408	
9									
10	One time COSTS	(\$75,000)							
11									
12	Recurring Costs	\$0	(\$35,000)	(\$35,000)	(\$35,000)	(\$35,000)	(\$35,000)		
13	Discount Rate (12%)	1.0000	0.8929	0.7972	0.7118	0.6355	0.5674		
14	PV of Recurring Costs	\$0	(\$31,252)	(\$27,902)	(\$24,913)	(\$22,243)	(\$19,859)		
15									
16	NPV of All COSTS	(\$75,000)	(\$106,252)	(\$134,154)	(\$159,067)	(\$181,309)	(\$201,168)	(\$201,168)	
17									
18	Overall NPV							\$105,240	
19									
20	Overall ROI - (Overall NPV / NPV of All COSTS)							0.5231448	
21									
22	Break-Even Analysis								
23	Yearly NPV Cash Flow	(\$75,000)	\$44,645	\$39,860	\$35,590	\$31,775	\$28,370		
24	Overall NPV Cash Flow	(\$75,000)	(\$30,355)	\$9,505	\$45,095	\$76,870	\$105,240		
25									
26	Project break even occurs between years 1 and 2								
27	Use first year of positive cash flow to calculate break even fraction $((39,860 - 9,505) / 39,860)$ = 0.7615								
28	Actual break-even occurred at 1.8 years								

Relationship b/w Future Value & Present Value

$$FV = PV (1 + k)^n$$

In Other words $PV = FV \times \frac{1}{(1 + k)^n}$

FV = Future Value
PV = Present Value
K = Discounted Rate
n = Number of Years

Let's build on Future Value

How do we calculate the Future Value ?

Example

If \$ 100 dollars is invested in a bank today may earn 8% per year.
what is the future value of the \$ 100 dollars for 1st, 5th and 15th year?

$$FV = PV (1 + k)^n$$

FV = Future Value
PV = Present Value
K = Discounted Rate
n = Number of Years

For 1 year

$$100 \times \left(1 + \frac{8}{100}\right)^1$$

For 5 years

$$100 \times \left(1 + \frac{8}{100}\right)^5$$

$$100 \times (1.08)^5$$

For 15 years

$$100 \times \left(1 + \frac{8}{100}\right)^{15}$$

PV =	100	=	=	=	100
K =	8%	=	8/100	=	0.08
(1+K)=	(1+0.08)	=	1.08	=	1.08

After 1 year(n=1):

$$FV = 100 \times (1.08)^1 = 100 \times 1.08 = \$108$$

After 5 years (n=5):

$$FV = 100 \times (1.08)^5 = 100 \times 1.08 \times 1.08 \times 1.08 \times 1.08 \times 1.08 = \$146.93$$

After 15 years (n=15):

$$FV = 100 \times (1.08)^{15} = \$317.22$$

Let's build on Present Value

How do we calculate the Present Value ?

Example

If \$ 100 dollars is to be received after 1 year, what is the present value of \$100 dollars today?

If \$ 100 dollars is to be received after 5 years, what is the present value of \$100 dollars today?

If \$ 100 dollars is to be received after 15 years, what is the present value of \$100 dollars today?

Note: Discounted rate is 8% per year.

$$PV = FV \times \frac{1}{(1 + k)^n}$$

FV=	100	=	=	=	100
K =	8%	=	8/100	=	0.08
(1+K)=	(1+0.08)	=	1.08	=	1.08

$$100 \times \frac{1}{(1.08)^5}$$

For 1 year

$$100 \times \frac{1}{\left(1 + \frac{8}{100}\right)^1}$$

For 5 years

$$100 \times \frac{1}{\left(1 + \frac{8}{100}\right)^5}$$

For 15 years

$$100 \times \frac{1}{\left(1 + \frac{8}{100}\right)^{15}}$$

The Present value of \$ 100 to be received after 1 year is \$93 dollars today.

The Present value of \$ 100 to be received after 5 years is \$68 dollars today.

The Present value of \$ 100 to be received after 15 year is \$32 dollars today.

Net Present Value

Net present value = "Present value of cash all Inflows" – "Present value of all cash Outflows".

Example: Your pay slip

Net Salary = Gross Salary - Deductions

Similarly:

Net Present value is : All Cash Inflows – All Cash Outflows.

➤ IF NPV > 0 (positive), The project can be accepted, The greater the NPV, the better the project financial benefits.

Net Present Value

Example: Calculating NPV

A sum of \$ 400,000 dollars invested today in an IT project may give a series of below cash inflows in future:

\$ 70,000 in year 1
\$ 120,000 in year 2
\$ 140,000 in year 3
\$ 140,000 in year 4
\$ 40,000 in year 5

If Opportunity cost of capital is 8% per annum, then should we accept or reject the project?

$$\text{Formula for calculating Present value} = PV = FV \times \frac{1}{(1+k)^n}$$

Solution:

Step 1: Calculate the PV value of year 1, year2, year3, year4, and year5

Step 2: Sum up the PV of all years

Step3: NPV = Present value of all cash inflows – Present value of all cash outflow.

Step 4: If NPV is positive, Accept the project, if not Reject the project.

Net Present Value

PV for year1	=	70000/1.08	64814.81
PV for year2	=	120000/(1.08) ²	102880.7
PV for year3	=	140000/(1.08) ³	111136.5
PV for year4	=	140000/(1.08) ⁴	102904.2
PV for year5	=	40000/(1.08) ⁵	27223.33
Cash Inflow of all PVs	=		408959.5

Cash Inflow of all Present Values is : \$ 408,959

Present value of Cash outflow is : \$400,000

Net Present Value = PV of Cash inflows – PV of Cash Outflows
= (\$408959 – \$400000) = \$8959 dollars.

Since NPV is positive, (i.e., \$8959, This project can be accepted)

Net Present Value

Same example: Calculating NPV however with Discount rate or Opportunity cost of capital at 15%
A sum of \$ 400,000 dollars invested today in an IT project may give a series of below cash inflows in future:

- \$ 70,000 in year 1
- \$ 120,000 in year 2
- \$ 140,000 in year 3
- \$ 140,000 in year 4
- \$ 40,000 in year 5

If Opportunity cost of capital is 15% per annum, then should we accept or reject the project?

Solution: Calculating NPV

Step 1: Calculate the PV value of year 1, year2, year3, year4, and year5

Step 2: Sum up the PV of all years

Step3: NPV = Present value of all cash inflows – Present value of all cash outflow.

Step 4: If NPV is positive, Accept the project, if not Reject the project.

Net Present Value

Formula for calculating Present value = $PV = FV \times \frac{1}{(1 + k)^n}$

PV for year1	=	70000/1.15	60869.57
PV for year2	=	120000/(1.15) ²	90737.24
PV for year3	=	140000/(1.15) ³	92052.27
PV for year4	=	140000/(1.15) ⁴	80045.45
PV for year5	=	40000/(1.15) ⁵	19887.07
Cash Inflow of all PVs	=		343591.6

Cash Inflow of all Present Values is : \$ 343591

Present value of Cash outflow is : \$400,000

Net Present Value = PV of Cash inflows – PV of Cash Outflows = (\$343591– \$400000) = \$ -56408 dollars.

Since NPV is Negative, (i.e., -\$56408, This project should be rejected)

Pay back period

The time it takes for the project to generate money to pay for itself.

Payback period is the number of years required to recover the cash outflow invested in the project.

The project would be accepted if its payback period is less than the maximum or standard payback period set by Industry, Senior Leadership.

In terms of Projects ranking, it gives highest ranking to the project with the shortest payback period.

Note: In general, the discounted cash flow is not considered for Pay back period. Some do, but most don't!

Pay back period

A sum of \$25,000 invested today in an IT project, may give a series of cash inflows in future as described below.

\$ 5,000 in year 1
\$ 9,000 in year 2
\$ 10,000 in each of year 3
\$ 10,000 in each of year 4
\$ 3,000 in year 5

What is the Payback Period (Non-discounted)?

Year	Cash Inflow	Cumulative Cash inflow		
1	5000	5000		
2	9000	14000		
3	10000	24000		
4	10000	34000	833.333333	pm
5	3000	37000		

Initial Cash Outlay = \$25,000

Cumulative Non-discounted Cash Inflow in \$ dollars

End of Year 1: 5,000

End of Year 2: 14,000

End of Year 3: 24,000

End of Year 4: 34,000

Payback Period (Non-discounted) = In between 3 years 1 month and 3 years 2 months

SYSTEM ANALYSIS AND DESIGN

- Q1. Define system & list its various characteristics.
- Q2. Write in brief about various elements of a system
- Q3. Discuss the primary characteristics of open systems. In what way is a system entropic?
- Q4. Distinguish between:
 - Interaction & interdependence
 - Physical & Abstract Systems
 - Open & Closed systems
 - Schematic & static systems models
- Q5. How important is the informal information system in system analysis?
- Q6. Explain the various categories of information relevant to decision making in business?
- Q7. Discuss the concepts of MIS & DSS. How are they related? How do they differ?
- Q8. Describe the role and tasks of system analyst.
- Q9. What is system development life cycle. How does it relate to system analysis?
- Q10. List all the phases of system development life cycle in order & explain them in detail.
- Q11. Distinguish between initial investigation & feasibility study. In what way are they related?
- Q12. A number of activities are carried out under implementation. Elaborate.
- Q13. What are the considerations in deciding on a candidate system and why are they important?
- Q14. Explain briefly the levels of structuring work units in system development.
- Q15. Discuss top-down approach to system planning.
- Q16. What do you mean by Strategic and operational planning?
- Q17. What fact-finding techniques would use for investigating the information requirement of a large organization?
- Q18. Discuss the utility of Decision table & Structure chart.
- Q19. Where are brain storming & Delphi methods used? Discuss.
- Q20. What important information does the user's request form provide? Why is it so important in the initial investigation? Explain in detail.
- Q21. What do you mean by prototyping?
- Q22. What are the data flow diagrams? How are they different from structure charts?
- Q23. A question may be open or closed-ended. Illustrate the difference.
- Q24. Describe the process of design. Also discuss Logical & physical design.
- Q25. Describe the various goals considered for input & output design?
- Q26. Describe the concept and procedure used in constructing DFDs. Give an example.
- Q27. Discuss the behavioral issues involved in understanding the analyst/user interface.
- Q28. List four situations that make use of questionnaire appropriate.
- Q29. Elaborate on the pros and cons of data analysis & decision analysis method.

- Q30. Discuss the procedure for constructing a questionnaire.
- Q31. Under what circumstances would the analyst depend more heavily on external than internal information? Why?
- Q32. Distinguish between validity & reliability. How are they related?
- Q33. What traditional information gathering tools are available for the analyst?
- Q34. For what purpose would one use an interview rather than the data collection methods? Explain.
- Q35. What points should be considered in constructing a data dictionary?
- Q36. What basic rules are relevant to construct a DFD?
- Q37. What do you mean by leveling in DFDs? Explain with the help of an example.
- Q38. "A data dictionary is a structured repository of data about data". Discuss.
- Q39. Discuss the advantages being offered by data dictionary in the documentation area.
- Q40. In what way is a decision tree and DFD related? What about a decision tree and structured English?
- Q41. List and illustrate the primary uses and elements of a decision table.
- Q42. Who are the important users that must be interviewed? Describe the various steps to be taken to carry out an interview.
- Q43. What is meant by structured walk through? Why and how are they conducted?
- Q44. What makes up a system performance definition? Explain the steps to prepare the definition.
- Q45. What considerations are involved in feasibility analysis? Which consideration do you think is the most crucial? Why?
- Q46. How important is a project team in a feasibility analysis? Is it mandatory in every study? Where are the exceptions?
- Q47. What makes up a feasibility report? How would you change it?
- Q48. What cost elements are considered in cost/benefit analysis? Which element do you think is the most difficult to estimate.
- Q49. Discuss various cost estimation techniques.
- Q50. Define and explain then procedure for cost/benefit determination.
- Q51. How do NPV and present value analyses differ? Illustrate.
- Q52. What are the pros and cons of Payback method, cash-flow analysis and break-even analysis?
- Q53. Why do we test systems? How important is testing? Discuss.
- Q54. Discuss the various types of testing in detail.
- Q55. Outline the various activities that represent a test plan.
- Q56. What design specifications are considered in preparing a test plan?
- Q57. There are two ways of debugging program software: bottom-up and top-down. How do they differ?
- Q58. What is syntax error? How does it differ from a logic error? Give an example.
- Q59. Elaborate on the steps taken in system testing that lead to the user's acceptance of the system.
- Q60. Explain the differences between White-box and Black-box testing.
- Q61. How is stress testing different from volume testing?
- Q62. What types of test data are used in system testing?
- Q63. Define the term Quality Assurance. Discuss its importance in system design.

- Q64. List and briefly describe the factors that affect the quality of a system.
- Q65. What levels of quality assurance must a system meet. Explain.
- Q66. Discuss the role of the Data Processing Auditor in system testing.
- Q67. What is implementation? How does it differ from conversion? Elaborate.
- Q68. Explain the various activities in conversion. Which activity is the most important? Why?
- Q69. What is involved in converting files? Explain briefly.
- Q70. Distinguish between System maintenance and enhancement.
- Q71. What is the role of audit control trial in conversion? Who performs it? Explain.
- Q72. How System modification is different from software system audit.
- Q73. Review the primary activities of a maintenance procedure.
- Q74. Discuss the various training aids used for training users on a new system.
- Q75. Briefly explain the procedure and makeup of the post-implementation review. Can one perform maintenance on a system without a post implementation review? Why?