UNIT 1
Part I: SAD Introduction and Overview

Introduction

In business, System Analysis and Design refers to the process of examining a business situation with the intent of improving it through better procedures and methods.

System analysis and design related to **shaping organizations, improving performance and achieving objectives for profitability and growth.** The emphasis is on systems in action, the relationships among subsystems and their contribution to meeting a common goal.

Looking at a system and determining how adequately it functions, the changes to be made and the quality of the output are parts of system analysis.

Organizations are complex systems that consist of interrelated and interlocking subsystems. Changes in one part of the system have both anticipated and unanticipated consequences in other parts of the system. The systems approval is a way of thinking about the analysis and design of computer based applications. It provides a framework for visualizing the organizational and environmental factors that operate on a system. When a computer is introduced into an organization, various functions' and dysfunction's operate on the user as well as on the organization. Among the positive consequences are improved performance and a feeling of achievement with quality information. Among the unanticipated consequences might be a possible threat to employees job, a decreased morale of personnel due to back of involvement and a feeling of intimidation by users due to computer illiteracy. The analyst's role is to remove such fears and make the system a success.

System analysis and design **focus on systems, processes and technology.**

Overview of System Analysis and Design

Systems development can generally be thought of as having two major components: **Systems analysis and Systems design.** System design is the **process of planning a new business system or one to replace or complement an existing system.** But before this planning can be done, we must t**horoughly understand the old system and determine how computers can best be used to make its operation more effective.** System analysis, then, is the process of gathering and i**nterpreting facts, diagnosing problems, and using the information to recommend improvements to the system**. This is the job of the systems analyst.

What is a System?

The word System is derived from Greek word **Systema**, which means an organized relationship between any set of components to achieve some common cause or objective.

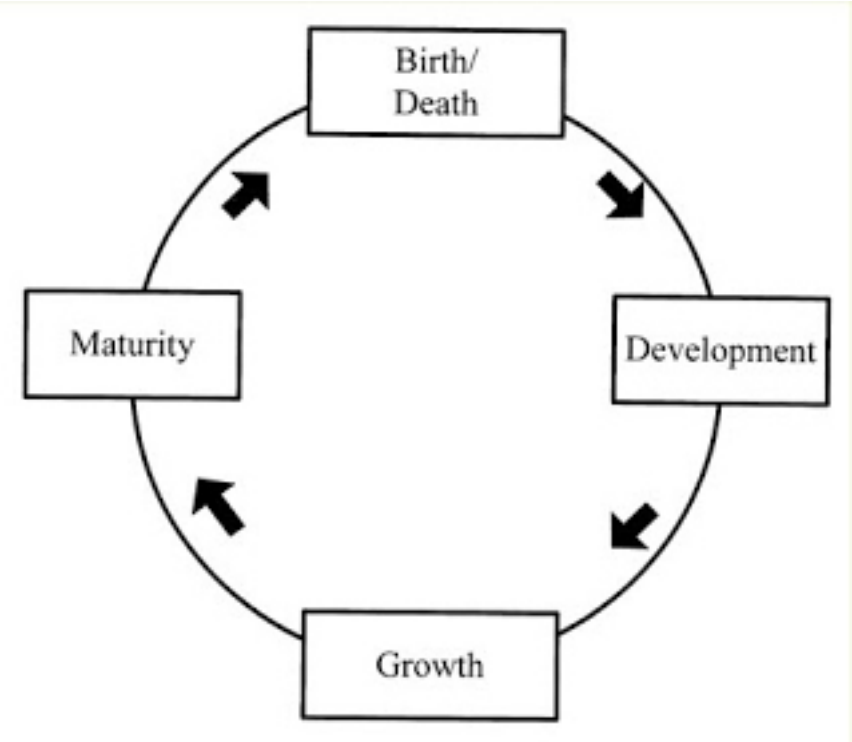


Fig: System Life Cycle

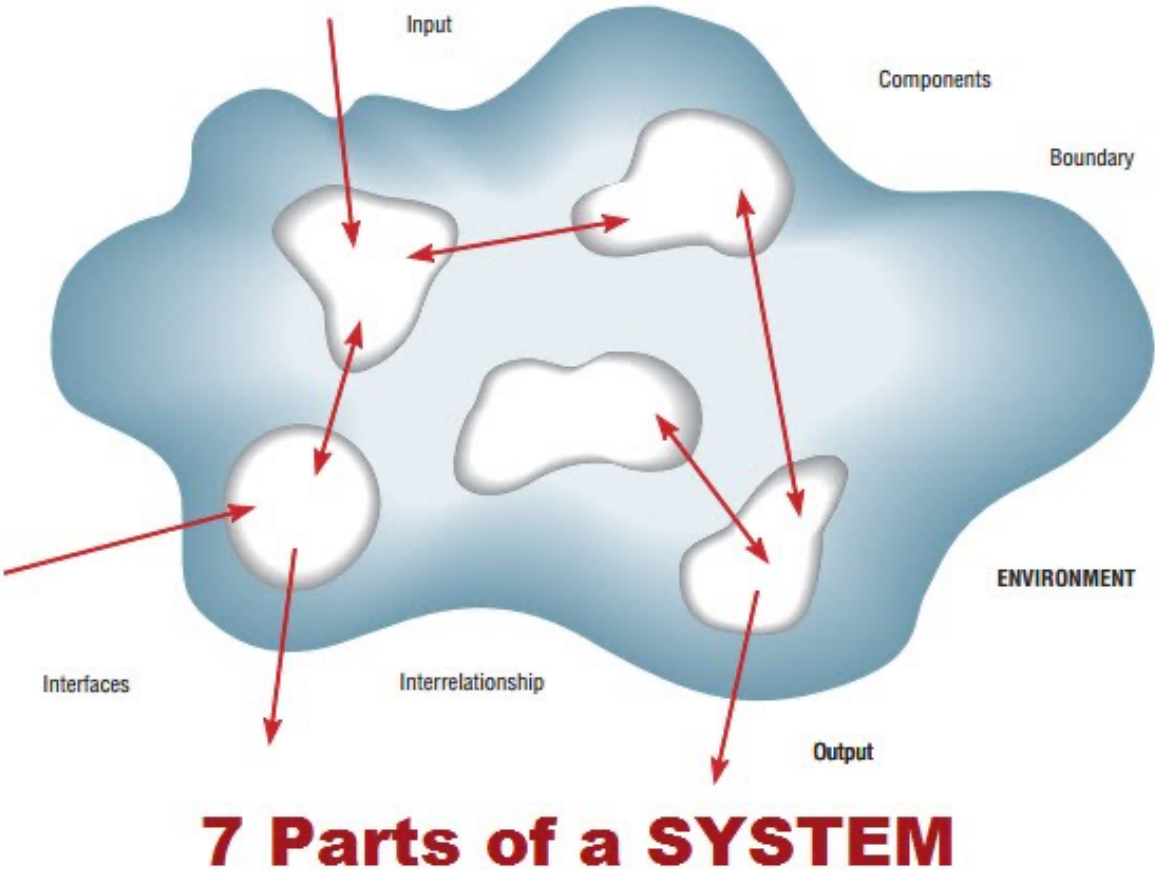
Every system has a life cycle as shown in figure. An information system is “born” when a problem is recognized. After the system is developed, it grows until it reaches maturity. Eventually, a change in the nature of the problem or increasing maintenance costs degrade the value of the system, so it “dies” and a new or replacement system is born to take its place.

System Analysis and Design, an interdisciplinary part of science, may refer to:

- **System analysis,** a method of studying a system by examining its component parts and their interactions
  - Structured data analysis (systems analysis), analysing the flow of information within an organization with data-flow diagrams
- **System design,** the process of defining the architecture, components, and data of a system to satisfy specified requirements
- **Object-oriented analysis and design**, an approach to analysis and design of an application, system, or business that emphasizes modularity and visual modeling
- **Service-oriented analysis and design,** a method of Service-oriented modeling to design business systems
- **Structured analysis,** methods in software engineering for converting specified requirements into software programs and hardware configurations

- **Structured system analysis and design method**, a systems approach to the analysis and design of information systems

System Characteristics



1. Components
2. Interrelated components
3. Boundary
4. Purpose
5. Environment
6. Interfaces
7. Constraints
8. Input
9. Output

A system is made up of components. A **component** is either an irreducible part or an aggregate of parts, also called a **subsystem**. The simple concept of a component is very powerful. For example, just as with an automobile or a stereo system, with proper design, we can repair or upgrade the system by changing individual components without having to make changes throughout the entire system. The components are **interrelated;** that is, the function of one is somehow tied to the functions of the others. For

example, the work of one component, such as producing a daily report of customer orders received, may not progress successfully until the work of another component is finished, such as sorting customer orders by date of receipt. A system has a **boundary,** within which all of its components are contained and which establishes the limits of a system, separating it from other systems. Components within the boundary can be changed, whereas systems outside the boundary cannot be changed. All of the components work together to achieve some overall **purpose** for the larger system: the system's reason for existing. A system exists within an **environment**—everything outside the system's boundary that influences the system. For example, the environment of a state university includes prospective students, foundations and funding agencies, and the news media. Usually the system interacts with its environment. A university interacts with prospective students by having open houses and recruiting from local high schools. An information system interacts with its environment by receiving data (raw facts) and information (data processed in a useful format). Those components which meets their one environment to get executed are called **interface**.

A Modern Approach to Systems Analysis and Design
1950s: focus on efficient automation of existing processes
1960s: advent of 3GL, faster and more reliable computers
1970s: system development becomes more like an engineering discipline
1980s: major breakthrough with 4GL, CASE tools, object oriented methods
1990s: focus on system integration, GUI applications, client/server platforms, Internet
The new century: Web application development, wireless PDAs, component-based applications.

System Concepts:
There are several system concepts with which systems analysts need to become familiar:

Decomposition

Modularity

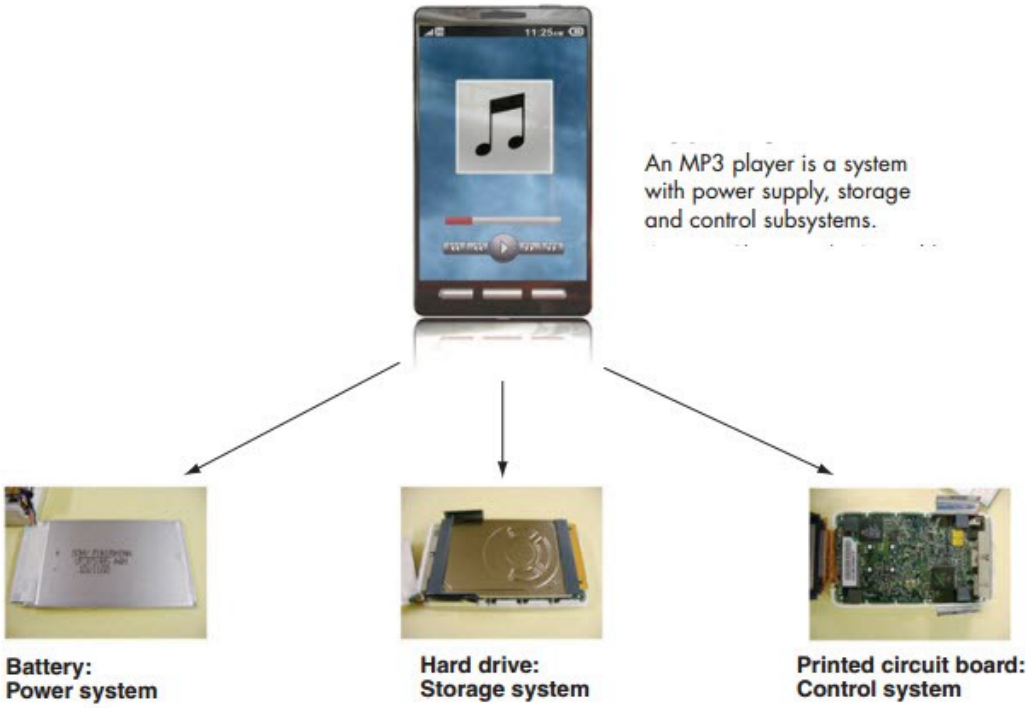
Cohesion

Coupling

i) **Decomposition**: Decomposition is the process of breaking down a system into its smaller components. These components may themselves be systems (subsystems) and can be broken down into their components as well. How does decomposition aid understanding of a system? It results in smaller and less complex pieces that are easier to understand than larger, complicated pieces. Decomposing a system also allows us to

focus on one particular part of a system, making it easier to think of how to modify that one part independently of the entire system. Decomposition is a technique that allows the systems analyst to:

- Break a system into small, manageable, and understandable subsystems.
- Focus attention on one area (subsystem) at a time, without interference from other areas.
- Concentrate on the part of the system pertinent to a particular group of users, without confusing users with unnecessary details.
- Build different parts of the system at independent times and have the help of different analysts



Modularity is a direct result of decomposition. It refers to dividing a system into chunks or modules of a relatively uniform size. Modules can represent a system simply, making it easier to understand and easier to redesign and rebuild. For example, each of the separate subsystem modules for the MP3 player in Figure above shows how decomposition makes it easier to understand the overall system.

Coupling means that subsystems are dependent on each other. Subsystems should be as independent as possible. If one subsystem fails and other subsystems are highly dependent on it, the others will either fail themselves or have problems functioning. Coupling is the measure of the

degree of interdependence between the modules. A good software will have low coupling. The best example is the control system, made up of the printed circuit board and its chips. Every function the MP3 player can perform is enabled by the board and the chips.

Cohesion is the extent to which a subsystem performs a single function. In the MP3 player example, supplying power is a single function. This brief discussion of systems should better prepare you to think about computer-based information systems and how they are built. Many of the same principles that apply to systems in general apply to information systems as well.

Types of System:

Physical or Abstract Systems

- Physical systems are tangible entities. We can touch and feel them.
- Physical System may be **static or dynamic** in nature. For example, desks and chairs are the physical parts of computer center which are static. A programmed computer is a dynamic system in which programs, data, and applications can change according to the user's needs.
- Abstract systems are non-physical entities or conceptual that may be formulas, representation or model of a real system. Eg: If any company works in PHP language, then the users or any organization might want them to design the application on different platform like Java, but this company works in PHP, where this system can't exist on Real World Trends.

Open or Closed Systems

- An open system must interact with its environment. It receives inputs from and delivers outputs to the outside of the system. For example, an information system which must adapt to the changing environmental conditions.
- A closed system does not interact with its environment. It is isolated from environmental influences. A completely closed system is rare in reality.

Adaptive and Non Adaptive System

- Adaptive System responds to the change in the environment in a way to improve their performance and to survive. For example, human beings, animals.
- Non Adaptive System is the system which does not respond to the environment. For example, machines.

Permanent or Temporary System

- Permanent System persists for long time. For example, business policies.

- Temporary System is made for specified time and after that they are demolished. For example, A DJ system is set up for a program and it is dissembled after the program.

Natural and Manufactured System

- Natural systems are created by the nature. For example, Solar system, seasonal system.
- Manufactured System is the man-made system. For example, Rockets, dams, trains.

Information System:

An information system is integrated and co-ordinate network of components, which combine together to convert data into information. An **information system** (**IS**) is a formal, sociotechnical, organizational system designed to collect, process, store, and distribute information.

The **six components** that must come together in order to produce an information system are:

1. **Hardware**: The term hardware refers to machinery and equipment. In a modern information system, this category includes the computer itself and all of its support equipment. The support equipment includes input and output devices, storage devices and communications devices. In pre-computer information systems, the hardware might include ledger books and ink.

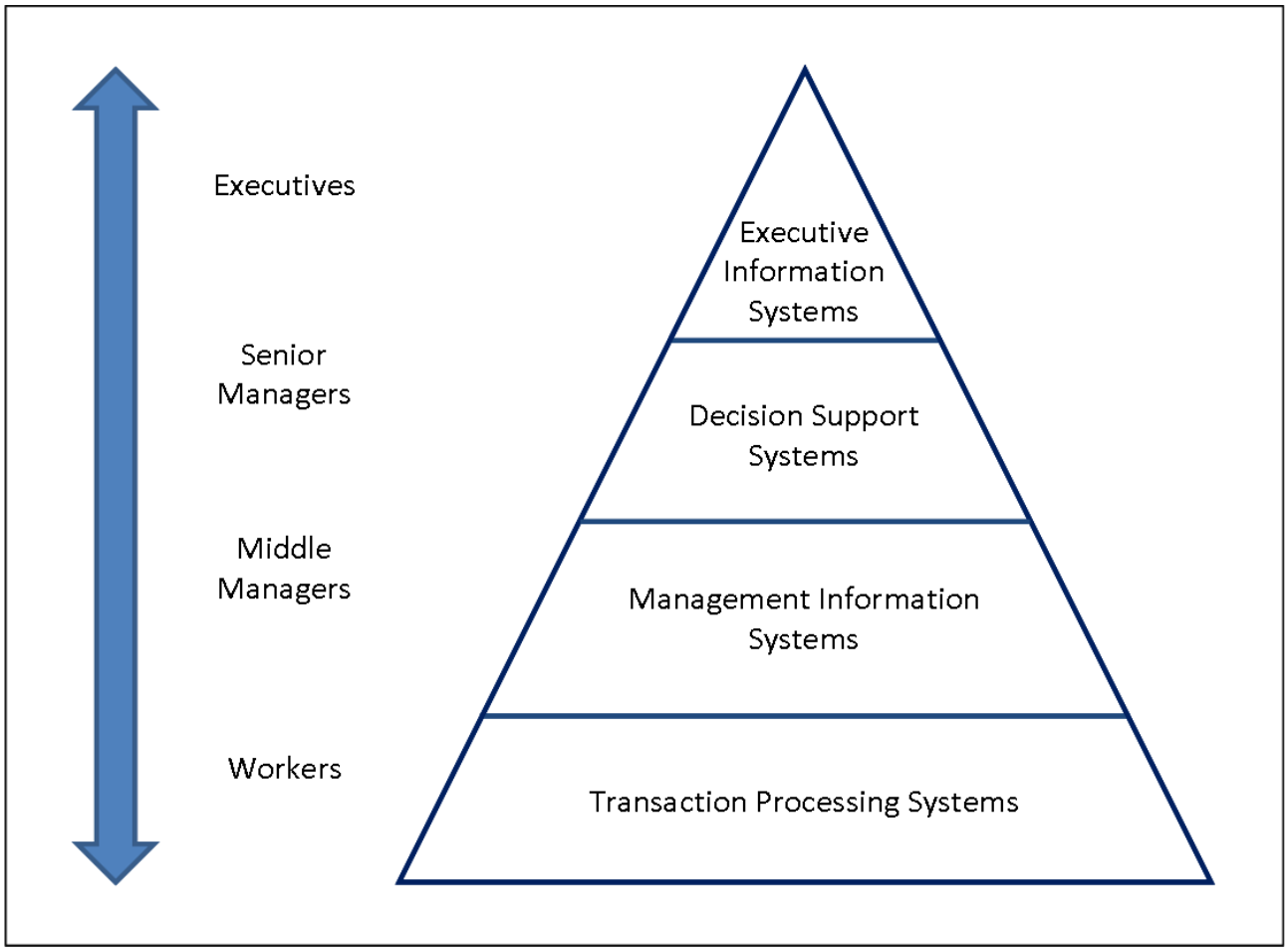
2. **Software**: The term software refers to computer programs and the manuals (if any) that support them. Computer programs are machine-readable instructions that direct the circuitry within the hardware parts of the system to function in ways that produce useful information from data. Programs are generally stored on some input/output medium, often a disk or tape. The "software" for pre-computer information systems included how the hardware was prepared for use (e.g., column headings in the ledger book) and instructions for using them (the guidebook for a card catalog).

3. **Data**: Data are facts that are used by systems to produce useful information. In modern information systems, data are generally stored in machine-readable form on disk or tape until the computer needs them. In pre-computer information systems, the data are generally stored in human-readable form.

4. **Procedures**: Procedures are the policies that govern the operation of an information system. "Procedures are to people what software is to hardware" is a common analogy that is used to illustrate the role of procedures in a system.

5. **People**: Every system needs people if it is to be useful. Often the most overlooked element of the system is the people, probably the component that most influence the success or failure of information systems. This includes "not only the users, but those who operate and service the computers, those who maintain the data, and those who support the network of computers."

6. **Feedback**: it is another component of the IS, that defines that an IS may be provided with feedback (Although this component isn't necessary to function).



Types of Information Systems:
Most businesses utilize six different information technology systems, each with functionality that assists in managing a particular business unit or organizational level.

1. **Transaction Processing Systems**
   A transaction encompasses all of the purchases and sales of products and services, along with any daily business transactions or activities required to operate a company. Quantities and the types of transactions performed vary, depending on the industry and size/scope of the

company. Examples of typical transactions include billing clients, bank deposits, new hire data, inventory counts, or a record of client-customer relationship management data. By utilizing a TPS, organizations can have a high level of reliability and accuracy in their user/customer data while minimizing the potential for human error.

2. Office Automation Systems

An office automation system is a network of various tools, technologies, and people required to conduct clerical and managerial tasks. Primarily, an office automation system assists in enhancing communication among different departments so everyone can collaborate to complete a task. An OAS can integrate with e-mail or word processing applications to ensure all communication data is easily accessible and in one centralized location. By utilizing an office automation system, businesses can improve communication between workers, streamline managerial activities, and optimize knowledge management.

3. Knowledge Management Systems

The word knowledge refers to the understanding that individuals acquire through study or practice, while management can be defined as the coordination of tasks from superiors to subordinates with a specific goal or endpoint to be achieved. Therefore, the definition of knowledge management can be identified as the process by which an organization or company collects, organizes, creates, and manages information in order to make it easily available to all employees. A knowledge management system stores and extracts information to help users enhance their knowledge and optimize collaboration efforts to complete tasks. Examples of documents found in a knowledge management system **include employee training materials, company policies, and procedures, or answers to customer questions.**

4. Management Information Systems

A management information system (**MIS**) is an information system used for **decision-making, and for the coordination, control, analysis, and visualization of information in an organization**. The study of the management information systems involves people, processes and technology in an organizational context. In a corporate setting, the ultimate goal of the use of a management information system is to increase the value and profits of the business. This is done by providing managers with timely and appropriate information allowing them to make effective decisions within a shorter period of time.

5. Decision Support Systems

A decision support system processes data to assist in management decision-making. It stores and gathers the information required for management to take the proper actions at the correct time. For example, a bank manager can

use a DSS to assess the evolving loan trends to determine which yearly loan targets to meet. Decision models are programmed into the IS to analyze and summarize large quantities of information and put it into a visual that makes it understandable. Because a DSS is interactive, management can easily add or delete data and ask important questions. This provides the evidence required for mid-management to make the right choices that will ensure the company meets its targets.

6. Executive Support System

Executive support systems are similar to a DSS but are primarily used by executive leaders and owners to optimize decision-making. An expert system helps enterprise leaders find answers to non-routine questions so they can make choices that improve the company's outlook and performance. Unlike a DSS, an executive support system provides better telecommunication functionality and a bigger computing functionality.

IS Development and SDLC

Information technology departments in larger organizations tend to strongly influence the development, use, and application of information technology in the business. A series of methodologies and processes can be used to develop and use an information system. Many developers use a systems engineering approach such as the system development life cycle (SDLC), to systematically develop an information system in stages.

System development is done in stages which include:

- Planning
- System Analysis and Requirements
- System Design
- Coding/Development
- Integration and Testing
- Implementation
- Operation and Maintenance

1. Planning

This is the first phase in the systems development process. It identifies whether or not there is the need for a new system to achieve a business"s strategic objectives. This is a preliminary plan (or a feasibility study) for a company"s business initiative to acquire the resources to build on an infrastructure to modify or improve a service. The company might be trying to meet or exceed expectations for their employees, customers and stakeholders too. The purpose of this step is to find out the scope of the problem and determine solutions. Resources, costs, time, benefits and other items should be considered at this stage.

2. Systems Analysis and Requirements

The second phase is where businesses will work on the source of their problem or the need for a change. In the event of a problem, possible solutions are submitted and analyzed to identify the best fit for the ultimate goal(s) of the project. This is where teams consider the functional requirements of the project or solution. It is also where system analysis takes place—or analyzing the needs of the end users to ensure the new system can meet their expectations. Systems analysis is vital in determining what a business"s needs are, as well as how they can be met, who will be responsible for individual pieces of the project, and what sort of timeline should be expected.
There are several tools businesses can use that are specific to the second phase. They include:

- CASE (Computer Aided Systems/Software Engineering)
- Requirements gathering
- Structured analysis

3. Systems Design
The third phase describes, in detail, the necessary specifications, features and operations that will satisfy the functional requirements of the proposed system which will be in place. This is the step for end users to discuss and determine their specific business information needs for the proposed system. It"s during this phase that they will consider the essential components (hardware and/or software) structure (networking capabilities), processing and procedures for the system to accomplish its objectives.

4. Development
The fourth phase is when the real work begins—in particular, when a programmer, network engineer and/or database developer are brought on to do the major work on the project. This work includes using a flow chart to ensure that the process of the system is properly organized. The development phase marks the end of the initial section of the process. Additionally, this phase signifies the start of production. The development stage is also characterized by instillation and change. Focusing on training can be a huge benefit during this phase.

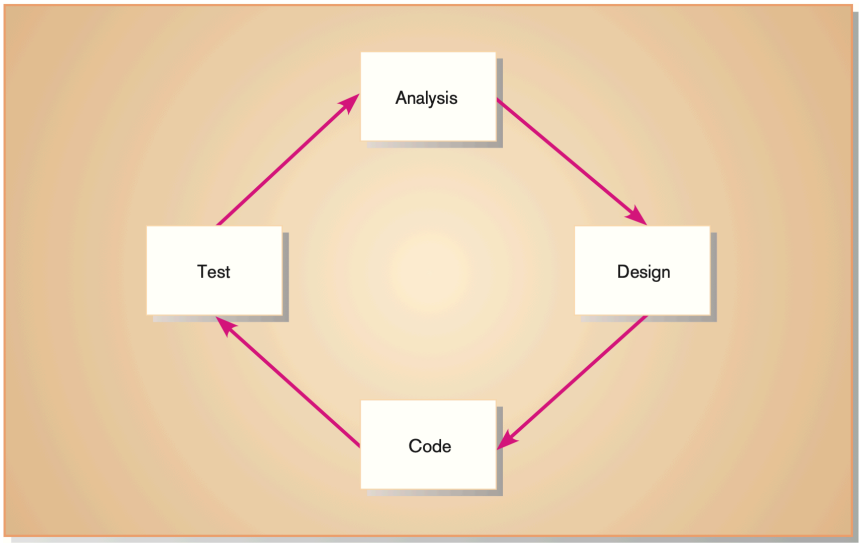
5. Integration and Testing
The fifth phase involves systems integration and system testing (of programs and procedures)—normally carried out by a Quality Assurance (QA) professional—to determine if the proposed design meets the initial set of business goals. Testing may be repeated, specifically to check for errors, bugs and interoperability. This testing will be performed until the end user finds it acceptable. Another part of this phase is verification and validation, both of which will help ensure the program"s successful completion.

6. Implementation
The sixth phase is when the majority of the code for the program is written. Additionally, this phase involves the actual installation of the newly-developed system. This step puts the project into production by moving the data and components from the old system and placing them in the new system via a direct cutover. While this can be a risky (and complicated) move, the cutover typically happens during off-peak hours, thus minimizing the risk. Both system analysts and end-users should now see the realization of the project that has implemented changes.

7. Operations and Maintenance
The seventh and final phase involves maintenance and regular required updates. This step is when end users can fine-tune the system, if they wish, to boost performance, add new capabilities or meet additional user requirements.

Importance of the SDLC
If a business determines a change is needed during any phase of the SDLC, the company might have to proceed through all the above life cycle phases again. The life cycle approach of any project is a time-consuming process. Even though some steps are more difficult than others, none are to be overlooked. An oversight could prevent the entire system from functioning as planned.

The Heart of System Development Process:
When developing information systems, most organizations use a standard of steps called the systems development lifecycle (SDLC) at the common methodology for systems development. SDLC includes phases such as planning, analysis, design, implementation, and maintenance. **At the heart of systems development, *analysis, design, code and test* are the different phases of SDLC.** These activities are the heart of systems development, as we suggest in this combination of activities started with **Rapid Application Development (RAD)**. and is seen in such current practices as the **Agile Methodologies**. A well-known instance of one of the Agile Methodologies is eXtreme Programming, although there are other variations. Further we will discuss about RAD and Agile Methodologies but first it is important that you learn about the problems with the traditional SDLC with Waterfall Model. You will read about these problems next. Then you will read about prototyping, Joint Application Design, and CASE tools, which make RAD, Agile Methodologies, and eXtreme Programming possible.



Traditional Waterfall Model

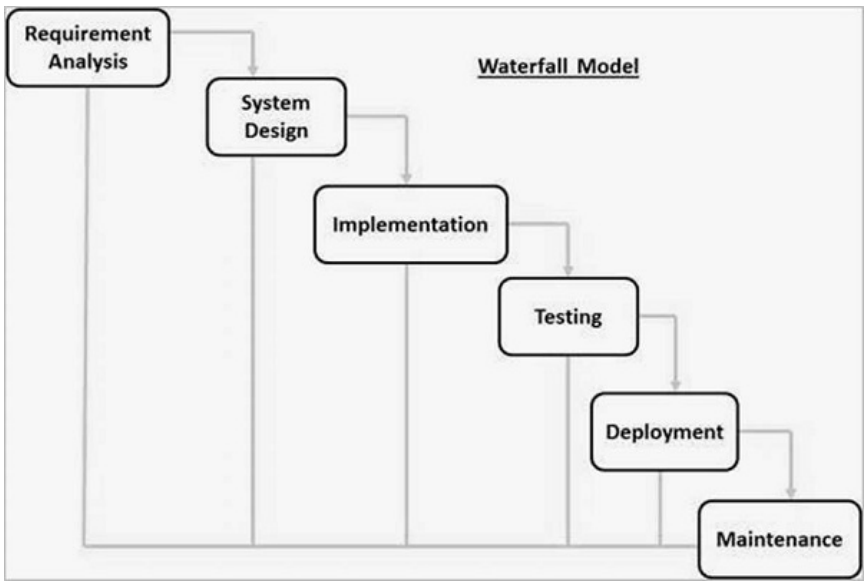


Fig: Waterfall Model

- **Requirement Gathering and analysis** − All possible requirements of the system to be developed are captured in this phase and documented in a requirement specification document.
- **System Design** − The requirement specifications from first phase are studied in this phase and the system design is prepared. This system design helps in specifying hardware and system requirements and helps in defining the overall system architecture.
- **Implementation** − With inputs from the system design, the system is first developed in small programs called units, which are integrated in the next phase. Each unit is developed and tested for its functionality, which is referred to as Unit Testing.
- **Integration and Testing** − All the units developed in the implementation phase are integrated into a system after testing

of each unit. Post integration the entire system is tested for any faults and failures.

- **Deployment of system** − Once the functional and non-functional testing is done; the product is deployed in the customer environment or released into the market.
- **Maintenance** − There are some issues which come up in the client environment. To fix those issues, patches are released. Also to enhance the product some better versions are released. Maintenance is done to deliver these changes in the customer environment.

Advantages of waterfall model

- This model is simple and easy to understand and use.
- It is easy to manage due to the rigidity of the model – each phase has specific deliverables and a review process.
- In this model phases are processed and completed one at a time. Phases do not overlap.
- Waterfall model works well for smaller projects where requirements are clearly defined and very well understood.

Disadvantages of waterfall model

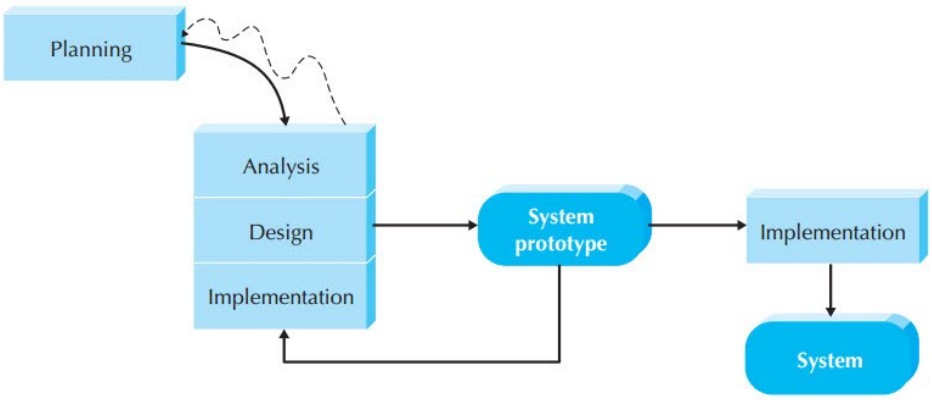
- Once an application is in the testing stage, it is very difficult to go back and change something that was not well-thought out in the concept stage.
- No working software is produced until late during the life cycle.
- High amounts of risk and uncertainty.
- Not a good model for complex and object-oriented projects.
- Poor model for long and ongoing projects.
- Not suitable for the projects where requirements are at a moderate to high risk of changing.

Alternatives to Traditional Waterfall SDLC

- Prototyping
- CASE Tools
- Joint Application Design
- Rapid Application Development
- Agile Methodologies
- eXtreme Programming

Prototyping:

A prototyping-based methodology performs the **analysis, design, and implementation** phases concurrently, and all three phases are performed repeatedly in a cycle until the system is completed. With these methodologies, the basics of analysis and design are performed, and work



immediately begins on a system prototype, a **"quick-and-dirty"** program that provides a minimal amount of features. The first prototype is usually the first part of the system that is used. This is shown to the users and the project sponsor, who provide comments. These comments are used to reanalyze, redesign, and reimplement a second prototype, which provides a few more features. This process continues in a cycle until the analysts, users, and sponsor agree that the prototype provides enough functionality to be installed and used in the organization. After the prototype (now called the system) is installed, refinement occurs until it is accepted as the new system. The key advantage of a prototyping-based methodology is that it very quickly provides a system with which the users can interact, even if it is not ready for widespread organizational use at first. Prototyping reassures the users that the project team is working on the system (there are no long delays in which the users see little progress), and prototyping helps to more quickly refine real requirements. Rather than attempting to understand a system specification on paper, the users can interact with the prototype to better understand what it can and cannot do.

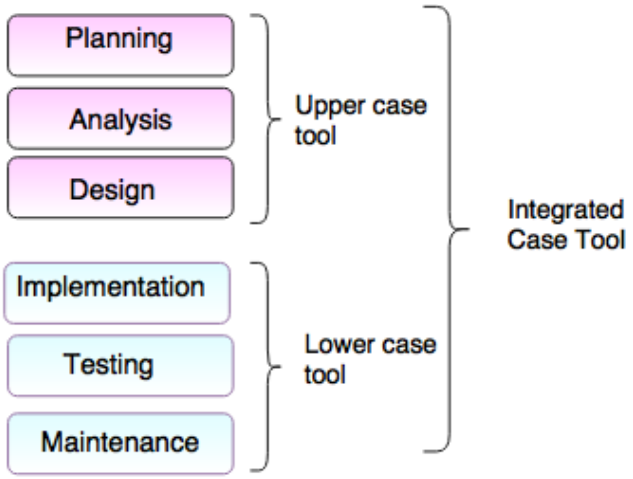
CASE Tools:
CASE stands for **Computer Aided Software Engineering**. It means, development and maintenance of software projects with help of various automated software tools.

CASE tools are set of software application programs, which are used to automate SDLC activities. CASE tools are used by software project managers, analysts and engineers to develop software system. There are number of CASE tools available to simplify various stages of Software Development Life Cycle such as **Analysis tools, Design tools, Project management tools, Database Management tools, Documentation tools** are to name a few. Use of CASE tools accelerates the development of project to produce desired result and helps to uncover flaws before moving ahead with next stage in software development.

Components of CASE Tools
CASE tools can be broadly divided into the following parts based on their use at a particular SDLC stage:

- **Central Repository** - CASE tools require a central repository, which can serve as a source of common, integrated and

consistent information. Central repository is a central place of storage where product specifications, requirement documents, related reports and diagrams, other useful information regarding management is stored. Central repository also serves as data dictionary.



- **Upper Case Tools** - Upper CASE tools are used in planning, analysis and design stages of SDLC.
- **Lower Case Tools** - Lower CASE tools are used in implementation, testing and maintenance.
- **Integrated Case Tools** - Integrated CASE tools are helpful in all the stages of SDLC, from Requirement gathering to Testing and documentation.

Case Tools Types

Diagram tools

These tools are used to represent system components, data and control flow among various software components and system structure in a graphical form. For example, Flow Chart Maker tool for creating state-of-the-art flowcharts.

Project Management Tools

These tools are used for project planning, cost and effort estimation, project scheduling and resource planning. Managers have to strictly comply project execution with every mentioned step in software project management. Project management tools help in storing and sharing project information in real-time throughout the organization. For example, Creative Pro Office, Trac Project, Basecamp.

Documentation Tools
Documentation in a software project starts prior to the software process, goes throughout all phases of SDLC and after the completion of the project.
Documentation tools generate documents for technical users and end users. Technical users are mostly in-house professionals of the development team who refer to system manual, reference manual, training manual, installation manuals etc. The end user documents describe the functioning and how-to of the system such as user manual. For example, Doxygen, DrExplain, Adobe RoboHelp for documentation.

Analysis Tools
These tools help to gather requirements, automatically check for any inconsistency, inaccuracy in the diagrams, data redundancies or erroneous omissions. For example, Accept 360, Accompa, CaseComplete for requirement analysis, Visible Analyst for total analysis.

Design Tools
These tools help software designers to design the block structure of the software, which may further be broken down in smaller modules using refinement techniques. These tools provides detailing of each module and interconnections among modules. For example, Animated Software Design (blender, animaker, autocad),etc.

Configuration Management Tools
An instance of software is released under one version. Configuration Management tools deal with –

- Version and revision management
- Baseline configuration management
- Change control management

CASE tools help in this by automatic tracking, version management and release management. For example, Fossil, Git, Accu REV.

Programming Tools
These tools consist of programming environments like IDE (Integrated Development Environment), in-built modules library and simulation tools. These tools provide comprehensive aid in building software product and include features for simulation and testing. For example, Cscope to search code in C, Eclipse, IntelliJ.

Prototyping Tools
Software prototype is simulated version of the intended software product. Prototype provides initial look and feel of the product and simulates few aspect of actual product. Prototyping CASE tools essentially come with graphical libraries. They can create hardware independent user interfaces and design. These tools help us to build rapid prototypes based on existing

information. In addition, they provide simulation of software prototype. For example: Serena prototype composer, Mockup Builder.

Web Development Tools
These tools assist in designing web pages with all allied elements like forms, text, script, graphic and so on. Web tools also provide live preview of what is being developed and how will it look after completion. For example, Fontello, Adobe Edge Inspect, Foundation 3, **Dreamweaver**.

Quality Assurance Tools
Quality assurance in a software organization is monitoring the engineering process and methods adopted to develop the software product in order to ensure conformance of quality as per organization standards. QA tools consist of configuration and change control tools and software testing tools. For example, SoapTest, AppsWatch, JMeter.

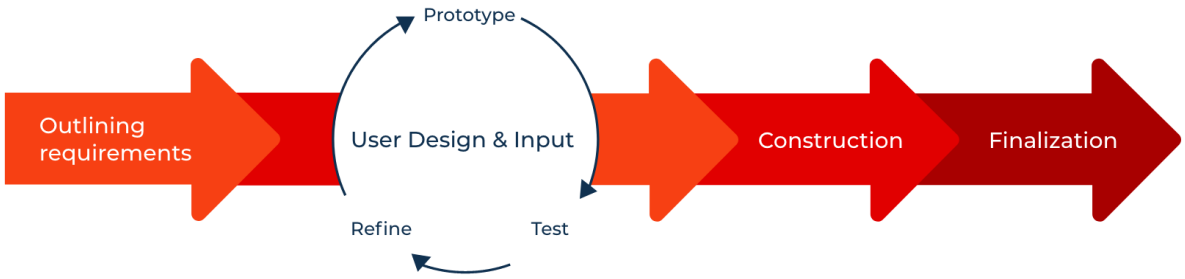
Maintenance Tools
Software maintenance includes modifications in the software product after it is delivered. Automatic logging and error reporting techniques, automatic error ticket generation and root cause Analysis are few CASE tools, which help software organization in maintenance phase of SDLC. For example, Bugzilla for defect tracking, HP ALM (Quality Center).

Joint Application Design (JAD)
Joint application design (**JAD**) is a process used in the life cycle area of the ***dynamic systems development method (DSDM)*** to collect business requirements while developing new information systems for a company. *A structured process in which users, managers, and analysts work together for several days in a series of intensive meetings to specify of review system requirements.*
"The JAD process also includes approaches for enhancing user participation, expediting development, and improving the quality of specifications." It consists of a workshop where "knowledge workers and IT specialists meet, sometimes for several days, to define and review the business requirements for the system." The attendees include high level management officials who will ensure the product provides the needed reports and information at the end. This acts as "a management process which allows Corporate Information Services (IS) departments to work more effectively with users in a shorter time frame".

Advantages
- JAD decreases time and costs associated with requirements elicitation (decision makers preferences) process. During 2-4 weeks information not only is collected, but requirements, agreed upon by various system users, are identified.

- JAD sessions help bring experts together giving them a chance to share their views, understand views of others, and develop the sense of project ownership.
- The methods of JAD implementation are well-known, as it is "the first accelerated design technique available on the market and probably best known", and can easily be applied by any organization.
- Easy integration of CASE tools into JAD workshops improves session productivity and provides systems analysts with discussed and ready to use models.

Rapid Application Development:



Outlining requirements - in the first phase, all the relevant members (managers, IT staff, users, etc.) plan and agree on the project's needs, scope, challenges, and requirements. A basic breakdown of this stage involves:

- Researching the current problem
- Defining the requirements for the project
- Finalizing the requirements with each stakeholder's approval

It is important that everyone has the opportunity to evaluate the goals and expectations for the project and weigh in. By getting approval from each key stakeholder and developer, teams can avoid miscommunications and costly change orders down the road.

User Design & input - Once the project is scoped out, it's time to jump right into development, building out the user design through various prototype iterations. During this phase, clients work hand in hand with developers to ensure their needs are being met at every step in the design process. **The RAD groups or subgroups typically use a combination of Joint Application Development (JAD) techniques and CASE tools to translate user needs into working models.** It's almost like customizable software development where the users can test each prototype of the product, at each stage, to ensure it meets their expectations. All the bugs and kinks are worked out in an iterative process. The developer designs a prototype, the client (user) tests it, and then they come together to communicate on what worked and what didn't. This method gives developers the opportunity to tweak the model as they go until they reach a satisfactory design.Both the

software developers and the clients learn from the experience to make sure there is no potential for something to slip through the cracks.

Construction - this is the other continuous phase and works hand-in-hand with user input. This step focuses on implementing the feedback provided by the users through coding, testing, and any other applicable development tasks. The phase feed into each other until the users approve the product. The phase breaks down into several smaller steps:

- Preparation for rapid construction
- Program and application development
- Coding
- Unit, integration, and system testing

Finalization or cutover - This is the implementation phase where the finished product goes to launch. It includes data conversion, testing, and changeover to the new system, as well as user training. All final changes are made while the coders and clients continue to look for bugs in the system.

Benefits of RAD methodology

RAD is one of the most successful software development programs available today, with numerous benefits for both software development teams as well as their clients.

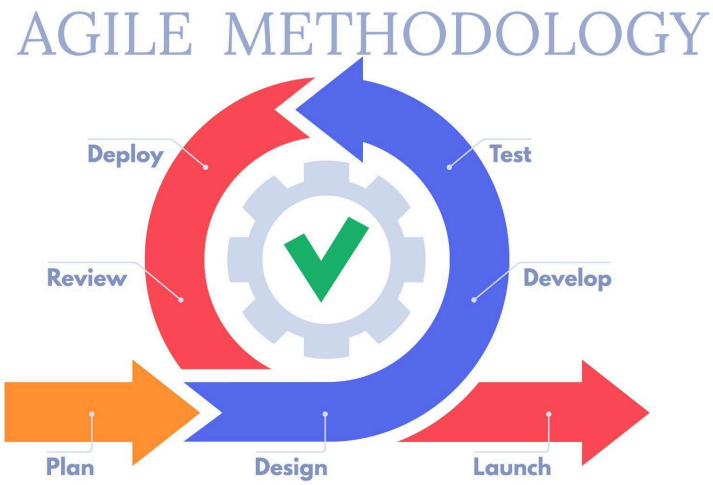
Here are just a few advantages:

- RAD lets you break the project down into smaller, more manageable tasks.
- The task-oriented structure allows project managers to optimize their team's efficiency by assigning tasks according to members' specialties and experience.
- Clients get a working product delivered in a shorter time frame.
- Regular communication and constant feedback between team members and stakeholders increases the efficiency of the design and build process.

With a shorter planning phase and a focus on highly iterative design and construction, RAD teams are able to accomplish more in less time without sacrificing client satisfaction.

A win-win condition for both project managers and clients.

Agile Methodology:



- Agile is a collection of principles used in software development and project management. Agile focuses on enabling teams to deliver work in small, workable increments, thus delivering value to their customers with ease. Evaluation of the requirements, plans, and results take place continuously. This helps the team in responding to changes in a quick manner.
- The Agile Methodologies share three key principles:
  (1) a focus on adaptive rather than predictive methodologies,
  (2) a focus on people rather than roles, and
  (3) a focus on self-adaptive processes.
- Agile software development refers to software development methodologies centered round the idea of iterative development, where requirements and solutions evolve through collaboration between self-organizing cross-functional teams. The ultimate value in Agile development is that it enables teams to deliver value faster, with greater quality and predictability, and greater aptitude to respond to change.

  Agile Methodology is dependent on following frameworks:
  - **Scrum**
  - **Kanban**
  - **eXtreme Programming**

What is Scrum?

Scrum is an agile process that helps to deliver the business value in the shortest time. It rapidly and repeatedly inspects actual working software. It emphasizes on teamwork and iterative progress of the software. Its goal is to deliver new software every 2-4 weeks. Scrum methodology can offer project management for every business, and even across life in general. By using Scrum, the development team becomes more Agile and discovering how to react quickly and respond to the sudden changes.

What is Kanban?

Kanban is a visual system for managing work. It visualizes both the process and the actual work passing through that process. The main objective of implementing Kanban is to identify potential bottlenecks in the process and fix them. Kanban goal is that work flow should proceed smoothly at an optimal speed. Kanban methodology is designed to meet minimal resistance. So it allows continuous small incremental and evolutionary changes to the current process. It also helps to achieve improvements regarding throughput, lead time and quality.

Agile Manifesto

1. **Customer Satisfaction:** Manifesto provides high priority to satisfy the costumer's requirements. This is done through early and continuous delivery of valuable software.
2. **Welcome Change:** Making changes during software development is common and inevitable. Every changing requirement should be welcome, even in the late development phase. Agile process works to increase the customers' competitive advantage.
3. **Deliver the Working Software:** Deliver the working software frequently, ranging from a few weeks to a few months with considering the shortest time period.
4. **Collaboration:** Business people (Scrum Master and Project Owner) and developers must work together during the entire life of a project development phase.
5. **Motivation:** Projects should be build around motivated team members. Provide such environment that supports individual team members and trust them. It makes them feel responsible for getting the job done thoroughly.
6. **Face-to-face Conversation:** Face-to-face conversation between Scrum Master and development team and between the Scrum Master and customers for the most efficient and effective method of conveying information to and within a development team.
7. **Measure the Progress as per the Working Software:** The working software is the key and primary measure of the progress.
8. **Maintain Constant Pace:** The aim of agile development is sustainable development. All the businesses and users should be able to maintain a constant pace with the project.
9. **Monitoring:** Pay regular attention to technical excellence and good design to maximize agility.
10. **Simplicity:** Keep things simple and use simple terms to measure the work that is not completed.
11. **Self-organized Teams:** The Agile team should be self-organized. They should not be depending heavily on other teams because the best architectures, requirements, and designs emerge from self-organized teams.

12. **Review the Work Regularly:** The work should be reviewed at regular intervals, so that the team can reflect on how to become more productive and adjust its behavior accordingly.

eXtreme Programming:
eXtreme programming (XP) is one of the most important software development framework of Agile models. It is used to improve software quality and responsive to customer requirements. The extreme programming model recommends taking the best practices that have worked well in the past in program development projects to extreme levels.

Extreme Programming is based on the following values −

- Communication
- Simplicity
- Feedback
- Courage
- Respect

Good practices needs to practiced extreme programming:

Some of the good practices that have been recognized in the extreme programming model and suggested to maximize their use are given below:

- **Code Review:** Code review detects and corrects errors efficiently. It suggests pair programming as coding and reviewing of written code carried out by a pair of programmers who switch their works between them every hour.
- **Testing:** Testing code helps to remove errors and improves its reliability. XP suggests ***test-driven development (TDD)*** to continually write and execute test cases.
- **Incremental development:** Incremental development is very good because customer feedback is gained and based on this development team come up with new increments every few days after each iteration.
- **Simplicity:** Simplicity makes it easier to develop good quality code as well as to test and debug it.
- **Design:** Good quality design is important to develop a good quality software. So, everybody should design daily.
- **Integration testing:** It helps to identify bugs at the interfaces of different functionalities. Extreme programming suggests that the developers should achieve continuous integration by building and performing integration testing several times a day.

Object-oriented analysis and Design

Object-oriented analysis and design (**OOAD**) is a technical approach for analyzing and designing an application, system, or business by applying object-oriented programming, as well as using visual modeling throughout the software development process to guide stakeholder communication and product quality.

Object-oriented analysis

The purpose of any analysis activity in the software life-cycle is to create a model of the system's functional requirements that is independent of implementation constraints.

The main difference between object-oriented analysis and other forms of analysis is that by the object-oriented approach we organize requirements around objects, which integrate both behaviors (processes) and states (data) modeled after real world objects that the system interacts with. In other or traditional analysis methodologies, the two aspects: processes and data are considered separately. For example, data may be modeled by **ER diagrams, and behaviors by flow charts or structure charts or use case diagrams.**

Common models used in OOA are use cases and object models. Use cases describe scenarios for standard domain functions that the system must accomplish. Object models describe the names, class relations (e.g. Circle is a subclass of Shape), operations, and properties of the main objects. User-interface mockups or prototypes can also be created to help understanding.

Object-oriented design

During object-oriented design (OOD), a developer applies implementation constraints to the conceptual model (where designers convert the structural form of system into visualization of system) produced in object-oriented analysis. Such constraints could include the hardware and software platforms, the performance requirements, persistent storage and transaction, usability of the system, and limitations imposed by budgets and time. Concepts in the analysis model which is technology independent, are mapped onto implementing classes and interfaces resulting in a model of the solution domain, i.e., a detailed description of how the system is to be built on *concrete technologies.*

Part II: Source of Software -

In the early days of computers, programming was considered to be something of a black art, performed by mad scientists who were clever enough (and probably mad enough) to program these strange beasts of technology known as computers. These scientists sat for endless hours ploughing through strange computer codes known as programs which were, undoubtedly, incomprehensible to anyone but themselves. These early programmers did not follow any methods or rules when constructing their programs; rather they applied their high intellects to the problem in a way that lesser mortals could not be expected to achieve.
The truth of the matter was that programming was difficult in those times, not least because of the fact that, in the very early days, no real programming languages existed. However, languages such as **FORTRAN and COBOL** soon did emerge, but they were not accompanied by any disciplined or formalized approaches to software development. In addition, there were no effective tools available to help the programmer in his/her task.
Thus programs tended to be thrown together in a haphazard manner, with no real attention being paid to trying to structure them. The prime objective in those days was to produce a program that was efficient and could be squeezed into as small an amount of computer storage as possible, as at that time the cost of computer power was quite prohibitive.

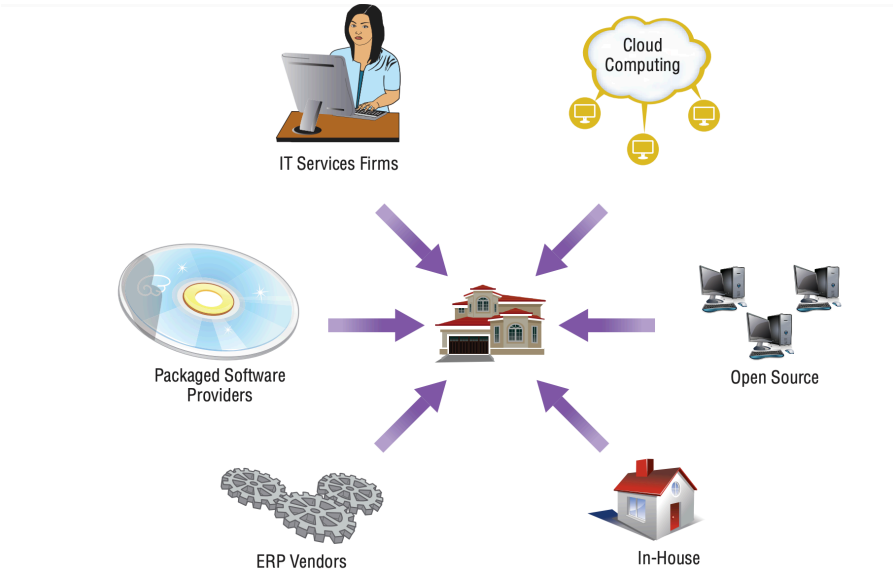
System Acquisition:
Software acquisition includes the processes typically associated with the software engineering life cycle. However, acquisition also includes processes that fund, manage, integrate, deploy and support software systems before, during, and after their software engineering life cycle. The need to address processes for systems and software engineering, inter-organization coordination and overall project management together is what establishes our baseline of interest in modeling and simulating software acquisition processes.

If we talk about the information system development, the first System was developed as General (GE) Electric Payroll System in United Kingdom. Even for more than decade of GE Payroll System development, no any software companies existed. Later on the method of Outsourcing was introduced during the time of early 70's. The practice of turning over responsibility for some or all of an organization’s information systems applications and operations to an outside firm is called **Outsourcing**. Eg: there exists one college called Pascal National College. But they have made a contract of canteen management with third party to give canteen facility in their college, which is called outsourcing. Same thing is implemented in organization level that to run their institute. Another Example: Our Pascal National College requires students on every semester session. So our organization did

partnership with *Saralshikshya.com* to bring students in this college. This kind of problem solving ability of one organization doing partnership with another organization to fulfill the job done is outsourcing. Now let us look at the sources of softwares:

Sources of Software

We can group organizations that produce software into six major categories:
(1) information technology services firms,
(2) packaged software providers,
(3) vendors of enterprise solutions software,
(4) cloud computing,
(5) open source software, and
(6) in-house development



Information Technology Services Firms

If a company needs an information system but does not have the expertise or the personnel to develop the system in-house and a suitable off-the-shelf system is not available, the company will likely consult an information technology (IT) services firm. IT services firms help companies develop custom information systems for internal use; they develop, host, and run applications for customers, or they provide other services. These well-known companies specialize in services, including custom systems development. These firms employ people with expertise in the development of information systems. Their consultants may also have expertise in a given business area. For example, consultants who work with banks understand financial institutions as well as information systems. Consultants use many of the same methodologies, techniques, and tools that companies use to develop systems in-house. Eg: IBM is the largest software developing company which help to build the different software products globally for the fulfillment of desire of every firms. Eg of IT service Firms are: Accenture, Capgemini, Sogenti, Computer Sciences Corporation (CSC), IBM, HP, etc.

Packaged Software Producers

The growth of the software industry has been phenomenal since its beginnings in the mid-1960s. Now, some of the largest computer companies in the world are companies that produce software exclusively. Software companies develop what are sometimes called prepackaged or off-the-shelf systems. Microsoft Products, Adobe Packages are popular examples of such software. The packaged software development industry serves many market segments. Its software offerings range from general, broad-based packages, such as general ledger, to more narrow, niche packages, such as software to help manage a day-care center. Software companies develop software to run on many different computer platforms, from microcomputers to large mainframes. The companies range in size from just a few people to thousands of employees. Software companies consult with system users after the initial software design has been completed and after an early version of the system has been built. The systems are then tested in actual organizations to reveal any problems or determine any improvements that can be made. *Until testing is completed, the system is not offered for sale to the public. Eg:* Microsoft, Intuit, Symantec, Adobe, etc.

Enterprise Software Solutions

On the modern days, organizations are choosing complete software solutions, called enterprise solutions or **enterprise resource planning (ERP)** systems, to support their operations and business processes. These ERP software solutions consist of a series of integrated modules. Each module supports an individual traditional business function, such as accounting, distribution, manufacturing, or human resources. The difference between the modules and traditional approaches is that the modules are integrated to focus on business processes rather than on business functional areas. For example, a series of modules will support the entire order-entry process, from receiving an order to adjusting inventory to shipping to billing to after-the-sale service. The traditional approach would use different systems in different functional areas of the business, such as a billing system in accounting and an inventory system in the warehouse. Using ERP systems, a firm can integrate all parts of a business process in a unified information system. There are needs like managing finance data, the employee records, sales records, payroll data, access for various people to various systems in the company, requests and approvals for expense re-reimbursements and what not. All aspects of a single transaction occur seamlessly within a single information system, rather than in a series of disjointed, separate systems focused on business functional areas. The benefits of the enterprise solutions approach include a single repository of data for all aspects of a business process and the flexibility of the modules. A single repository ensures more consistent and accurate data, as well as less maintenance. The modules are flexible because additional modules can be added as needed once the basic system is in place. Added modules are immediately integrated into the existing system. Softwares like Oracle provides software solutions for automating such kind of requirements in an enterprise. It provides Analytics service to analyze the daily works of every employee too. Eg: Oracle, SAP, AG, etc.

Cloud Computing

Cloud computing is the on-demand availability of computer system resources, especially data storage (cloud storage) and computing power, without direct active management by the user. The term is generally used to describe data centers available to many users over the Internet. Large clouds, predominant today, often have functions distributed over multiple locations from central servers. If the connection to the user is relatively close, it may be designated an edge server. The provision of computing resources,

including applications, over the Internet, so customers do not have to invest in the computing infrastructure needed to run and maintain the resources.
Cloud computing refers to the provision of applications over the Internet, where customers do not have to invest in the hardware and software resources needed to run and maintain the applications. You may have seen the Internet referred to as a cloud in other contexts, which comes from how the Internet is depicted on computer network diagrams. Eg: Amazon (AWS), Google (GCP), Microsoft(Azure), Salesforce(Trailhead/Trailblazer), etc.

Open Source
Open-source software is computer System software that is released under a license in which the copyright holder grants users the rights to use, study, change, and distribute the software and its source code to anyone and for any purpose. **Open source** is source code that is made freely available for possible modification and redistribution. That means it usually **includes a license for programmers** to change the software in any way they choose: They can fix bugs, improve functions, or adapt the software to suit their own needs. Eg: Linux, Apache, Firefox, Python, etc.

In House Development:
In-house refers to conducting an activity or operation within a company, instead of relying on outsourcing. A firm uses its own employees and time to keep a division or business activity. We have talked about several different types of external organizations that serve as sources of software, but in-house development remains an option. Of course, in-house development need not entail development of all of the software that will compose the total system. The in-house development gives you complete control of delivery of your product. You can use the full strength of the team to get the delivery done. You can grow your knowledge about your development domain if you have an in-house team. As the complete team is at your place you can understand workflow and manage the project well planned. With an in-house team, the company can ensure they are trained to the requisite level and, via exclusive contracts, can hang on to any star performers whose work differentiates the company from the competition.

Off-the-Shelf Software:
Off the shelf software are standardised software applications that are mass-produced, available to the general public, and fit for immediate use. They are designed for a broad range of customers, offering a comprehensive set of features to streamline operations. Eg: MS Office, adobe photoshop, ms windows, etc.....
The most common criteria, highlighted in off-the-self software are as follows:
- Cost effective
- Well Maintained Functionality
- Vendor support

■ Flexibility
■ Well Documentation
■ Quick Response time
■ Ease of installation

Reuse:
In System Development technique, the term Reuse refers to the feature of using the existing code or document or designed objects so that the coder or designer or analysts do not need to re-analyse or redesign the system even whenever they need to update the new versions. The use of previously written software resources, especially objects and components, in new applications is called *reuse*.
Reuse is the use of previously written software resources in new applications. Because so many bits and pieces of applications are relatively generic across applications, it seems intuitive that great savings can be achieved in many areas if those generic bits and pieces do not have to be written anew each time they are needed. Reuse should increase programmer productivity, because being able to use existing software for some functions means they can perform more work in the same amount of time. Reuse should also decrease development time, minimizing schedule overruns. Because existing pieces of software have already been tested, reusing them tends to result in higher-quality software with lower defect rates, decreasing maintenance costs.

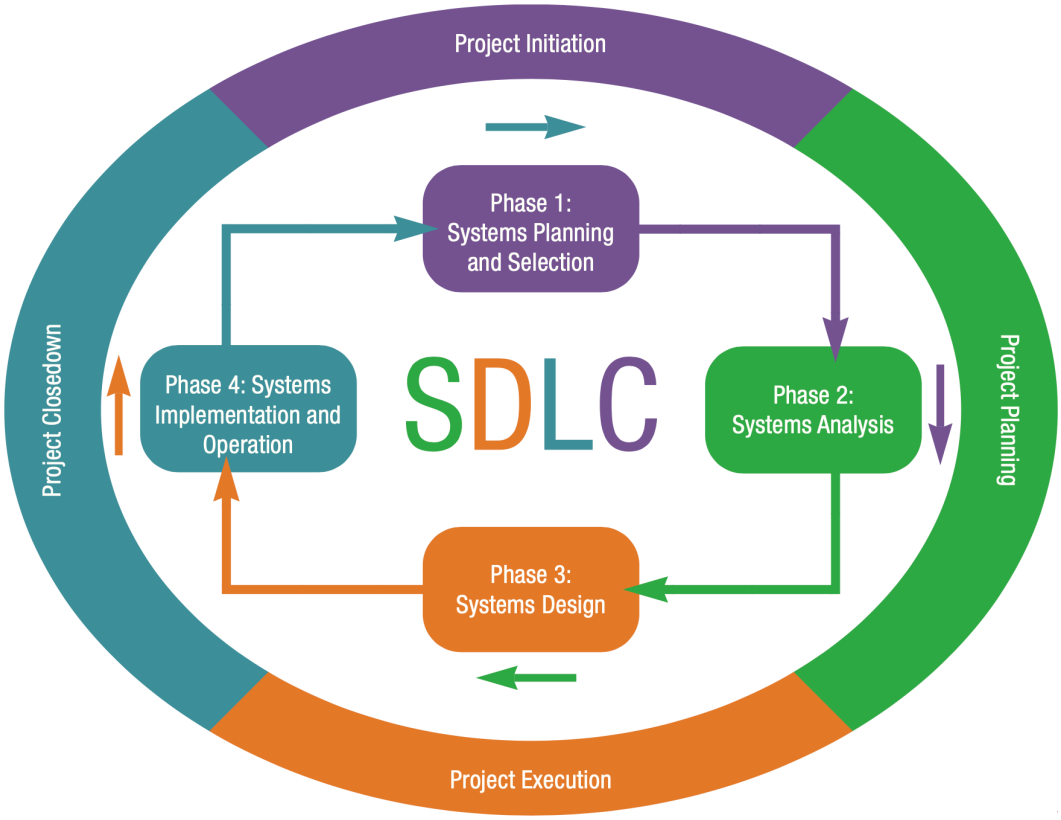
Part III: Managing the Information System Projects

Project management is an important aspect of the development of information systems and a critical skill for a systems analyst. The focus of project management is to ensure that system development projects meet customer expectations and are delivered within budget and time constraints.

Project manager:

A systems analyst with a diverse set of skills: management, leadership, technical, conflict management, and customer relationship who is responsible for *initiating, planning, executing, and closing down a project*. As a project manager, your environment is one of continual change and problem solving.

A **project** is a planned undertaking of a series of related activities, having a beginning and an end, to reach an objective. The first question you might ask yourself is, Where do projects come from? After considering all the different things that you could be asked to work on within an organization, the next question may be, How do I know which projects to work on? The ways in which each organization answers these questions vary.



1) Initiating the Project:

During **project initiation** the project manager performs several activities that assess the size, scope, and complexity of the project, and establishes procedures to support subsequent activities.

- Establishing the Project Initiation Team
- Establishing a Relationship with the Customer
- Establishing the Project Initiation Plan
- Establishing Management Procedures
- Establishing the Project Management Environment and Project Workbook
- Developing the Project Charter:
  - *Project title and date of authorization*
  - *Project manager name and contact information*
  - *Customer name and contact information*
  - *Projected start and completion dates*
  - *Project description and objectives*
  - *Key assumptions or approach*

2) Planning the Project

The next step in the project management process is **project planning**. Project planning involves defining clear, discrete activities and the work needed to complete each activity within a single project. It often requires you to make numerous assumptions about the availability of resources such as hardware, software, and personnel.

- Describing Project Scope, Alternatives, and Feasibility
- Dividing the Project into Manageable Tasks
- Estimating Resources and Creating a Resource Plan
- Developing a Preliminary Schedule
- Developing a Communication Plan
- Determining Project Standards and Procedures
- Identifying and Assessing Risk
- Creating a Preliminary Budget
- Developing a Project Scope Statement
- Setting a Baseline Project Plan

3) Executing the Project

Project execution puts the baseline project plan into action. Within the context of the SDLC, project execution occurs primarily during the analysis, design, and implementation phases. During the development of the Purchasing Fulfillment System, Project Manager is responsible for five key activities during project execution.

1. Executing the Baseline Project Plan
2. Monitoring Project Progress Against the Baseline Project Plan
3. Managing Changes to the Baseline Project Plan
4. Maintaining the Project Workbook
5. Communicating the Project Status

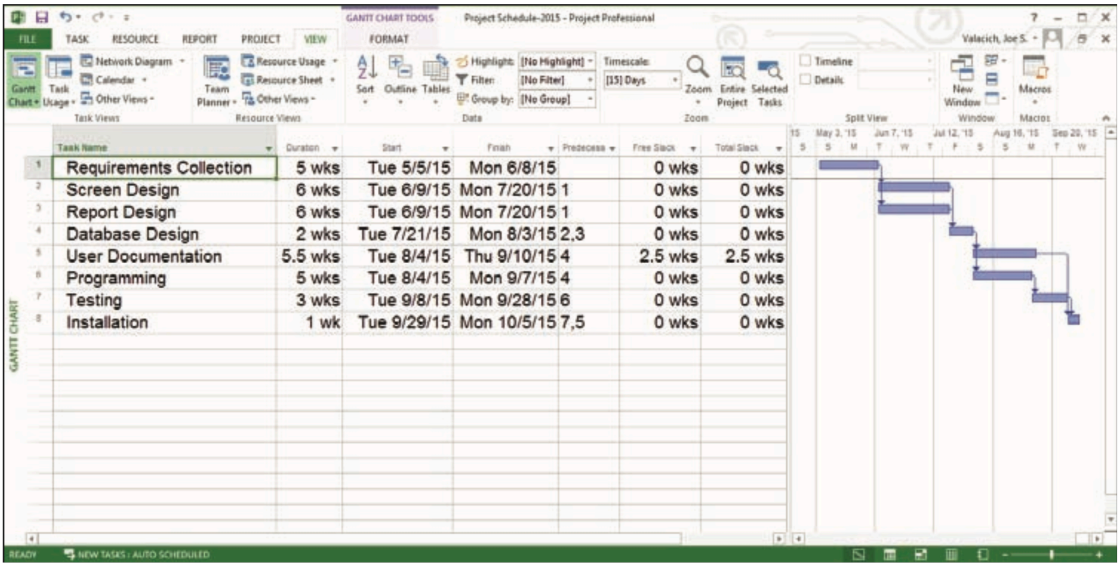
4) Closing Down the Project:

The focus of **project closedown** is to bring the project to an end. Projects can conclude with a natural or unnatural termination. A natural termination occurs when the requirements of the project have been met the project has been completed and is a success. An unnatural termination occurs when the project is stopped before completion. Several events can cause an unnatural termination of a project.

1. Closing Down the Project
2. Conducting Post project Reviews
3. Closing the Customer Contract

Representing and Scheduling Project Plans:

A project manager has a wide variety of techniques available for depicting and documenting project plans. These planning documents can take the form of graphical or textual reports, although graphical reports have become most popular for depicting project plans. The most commonly used methods are **Gantt charts and Network diagrams**. Because Gantt charts do not show how tasks must be ordered (precedence) but simply show when a task should begin and when it should end, they are often more useful for depicting relatively simple projects or subparts of a larger project, the activities of a single worker, or for monitoring the progress of activities compared to scheduled completion dates. Recall that a Network diagram shows the ordering of activities by connecting a task to its predecessor and successor tasks.



	Task Name	Duration	Start	Finish	Predecessors	Free Slack	Total Slack
1	Requirements Collection	5 wks	Tue 5/5/15	Mon 6/8/15		0 wks	0 wks
2	Screen Design	6 wks	Tue 6/9/15	Mon 7/20/15	1	0 wks	0 wks
3	Report Design	6 wks	Tue 6/9/15	Mon 7/20/15	1	0 wks	0 wks
4	Database Design	2 wks	Tue 7/21/15	Mon 8/3/15	2,3	0 wks	0 wks
5	User Documentation	5.5 wks	Tue 8/4/15	Thu 9/10/15	4	2.5 wks	2.5 wks
6	Programming	5 wks	Tue 8/4/15	Mon 9/7/15	4	0 wks	0 wks
7	Testing	3 wks	Tue 9/8/15	Mon 9/28/15	6	0 wks	0 wks
8	Installation	1 wk	Tue 9/29/15	Mon 10/5/15	7,5	0 wks	0 wks

Fig: Gantt Diagram

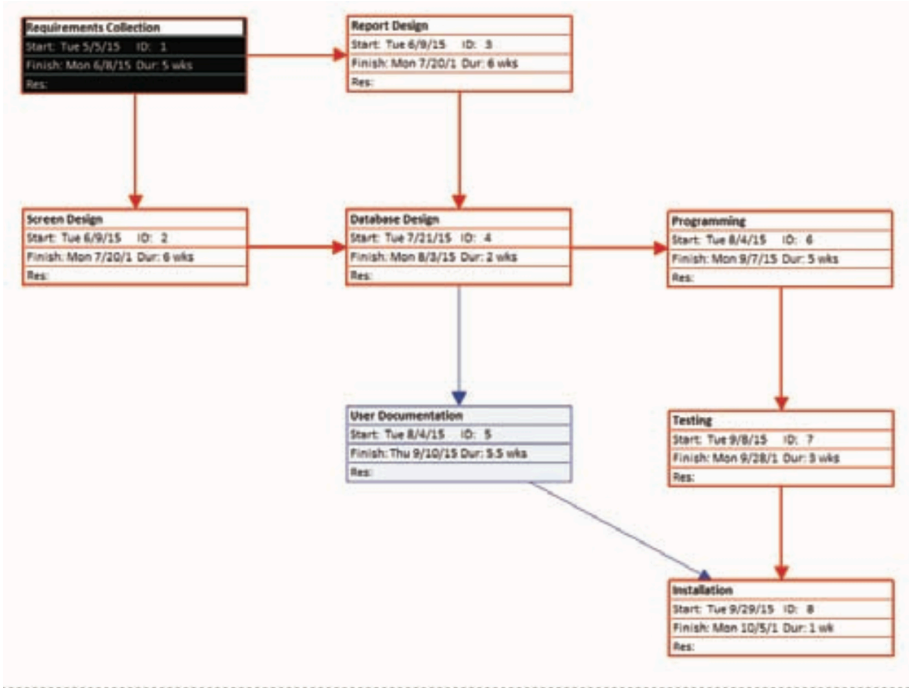


Fig: Network Diagram or PERT Chart

Representing and Scheduling Project Plans

Project scheduling and management requires that *time, costs, and resources* be controlled. **Resources** are any person, group of people, piece of equipment, or material used in accomplishing an activity. Network diagramming is a **critical path scheduling** technique used for controlling resources. A critical path refers to a sequence of task activities whose order and durations directly affect the completion date of a project. A network diagram is one of the most widely used and best-known scheduling methods.

A major strength of network diagramming is its ability to represent how completion times vary for activities. Because of this, it is more often used than Gantt charts to manage projects such as information systems development where variability in the duration of activities is the norm. Network diagrams are composed of circles or rectangles representing activities and connecting arrows showing required work flows.

Formula- Program Evaluation and Review Technique (PERT) evaluation:

ET = (O+4R+P)/6

- *Optimistic Time (O): the minimum possible time required to accomplish a task, assuming everything proceeds better than is normally expected.*

- *Pessimistic Time (P): the maximum possible time required to accomplish a task, assuming everything goes wrong (excluding major catastrophes).*
- *Most likely Time (R): the best estimate of the time required to accomplish a task, assuming everything proceeds as normal.*

Q) Your team members tell you that an activity you are working on is most likely to be completed in 20 days. However, the worst case, it might take 30 days, and if all conditions are favorable, it might be completed in 15 days. Determine the PERT time estimate for this activity.

Solution
We need the Optimistic Time, Pessimistic Time, and Most Likely Time for the activity to determine the PERT estimate.

The question says it is most likely that the task can be completed in 20 days, hence hence its assumed as max time:
Most Likely Realistic Time(R) = 20 days

It also says that in the worst case it may take 30 days, hence:
Pessimistic Time(P) = 30 days

Finally, it says that if all conditions are favorable, it will take 15 days to complete the task, hence its assumed as min time:
Optimistic Time (O) = 15 days.

Now,

PERT Estimate (ET) = (O+4R+P) / 6
= [15 + 4X20 + 30] / 6
= [15 + 80 + 30] / 6
= 125 / 6
= 20.83 days

Hence, the PERT estimate for this activity is 20.83 days.

How to Make a Project Schedule:

Project scheduling occurs during the planning phase of the project life cycle. When beginning to put together a schedule for your project, you should ask yourself four things to start:

What needs to be done?

When will it be done?

Who will do it?

Where will it be done?

The answers to these four questions will greatly inform your project schedule moving forward, as you'll use this information to plan start and end dates, link activities, set the duration, create milestones and manage resources.

Follow these steps to create a project schedule of your own!

1. Create the Project Scope

The project scope statement is created during the initial planning. It's a document that contains the specific goals, deliverables, features, budget, etc of your project. All of the tasks needed to complete the project successfully are listed here (which requires understanding the stakeholder's requirements.)

Be thorough when putting a task list together, you don't want to leave anything out. Understanding task dependencies and their sequence is very important for schedule management.

2. Establish the Sequence of Tasks

Tasks are the small jobs that lead to the final deliverable, and it's fairly crucial to map out the sequence of those tasks before diving into them. Oftentimes a task will be dependent on another to start or finish. You don't want to get halfway through a task before you realize you can't complete it due to hanging objectives.

3. Group Tasks

Once you've collected your tasks and have them in proper order, you should take the opportunity to divide your tasks by importance. You need to know which is critical to the project and must be done first and those less important that can be done if there's time. You could use a priority matrix to facilitate this process.

Then, break your tasks down with milestones that relate to the five project phases—*initiation, planning, execution, monitoring and close*. Organizing your tasks with milestones makes it easier to track progress, and gives your teams a sense of accomplishment that boosts morale and productivity.

4. Link Task Dependencies

Some tasks can be done simultaneously, but some tasks are dependent on others to start or finish before they can start or finish. These task

dependencies must be mapped out in your schedule to keep you aware of them, or you risk bottlenecks and blocking your team.

5. Find the Critical Path

The critical path is a method for scheduling tasks in a project to find those which are critical to the success of the project. This allows you to make smart choices about tasks that can be ignored if time and costs become constrained. This method is commonly used for schedule risk analysis.

6. Assign Resources

Resource management and project scheduling are closely related. Every task on your schedule should have the related resources and costs associated with completing it. Tasks aren't done on their own, and without mapping the resource availability to each task you're in danger of going wildly over budget. *With resources attached to tasks, you can more accurately plan your team's time and keep their workload balanced.*

Using Project Management Software:

Project management software are the tools and techniques required to deliver your projects successfully. Projects are comprised of stakeholders, their vision/goal, the resources required to achieve that goal, and management processes and tools to make sure you get there.
Project management is key to this success. Project management includes the technology, methodology, and resources that enable project completion. Over time, technology and methodology have gotten incredibly sophisticated.
Project management software helps managers control their costs and hit deadlines. It helps people deliver the projects they commit to. It even helps them track whether or not they've been paid on time and in the right amount. Project management software has many high-level bucket features.

Benefits of Using Project Management Software

Project management software provides small to large businesses, who undoubtedly juggle a number of tasks, with a solution that helps keep them organized. There are a wide variety of project management software packages available, including web-based applications accessible from any location. Each software boasts its own set of features, but they all share common benefits businesses can appreciate.

1. Collaboration on Projects Modules

Employees are often assigned individual tasks that are a part of a larger project an entire team is working to complete. Project management software gives employees a way to collaborate on projects by sharing documents, timelines and status updates.

2. Delegate Tasks

As a business owner, you likely weigh the knowledge, skills and abilities of employees before delegating tasks to them. Use project management software to easily delegate tasks to the appropriate employees. By assigning roles in the system, each employee has access to necessary information and knows who they should contact if they have questions or concerns, or need information about a particular topic.

3. Stay on Schedule

Project management software lets project managers add a start and expected completion date to projects and tasks they include in the system. This information alerts employees to upcoming deadlines, allowing them to manage their time appropriately to complete tasks before or on the listed due date.

4. Track Projects

Keep track of the progress of projects with project management software. The software will let you know what's been completed, as well as by whom, and what still needs to be done. Employees can provide updates as to what they're working on and share their updates with the project manager and team members. The software eliminates the need for status update meetings and emails.

5. Provide a Snapshot

When training new staff members and introducing them to projects your company works on, project management software offers a snapshot of the project you can share to get new staff up-to-speed. The snapshot allows you to show employees the project from start to finish, give them background information and let them know how the project will move forward.

6. Communicate with Clients and Vendors

Project management software enables businesses to share and collaborate with clients and vendors in addition to employees. Companies using project management software can provide their clients with usernames and passwords giving them access to project files. Clients can give feedback, make edits and review progress. Thats why businesses need to be connected to vendors and clients, project management technology is essential.

7. Identify risks

Every single project comes with its fair share of risks. It goes without saying that you will encounter one or the other. The advantage of project management is that it aids in earlier identification. When you are able to identify risk at an earlier stage, it becomes easier to deal with it effectively.

The last thing you want is to end up having to fix everything in the last. Addressing these problems at an early stage helps in preventing further problems and thus helps in better execution of the project.

8. Easy documentation

There is no doubt that documentation is an important part of project management. The management of project document is an essential part of every project. As the documents range from reports to project plans and files related to specific tasks. You can connect your files to Dropbox, Google Drive for additional document storage and collaboration features. Using a project management software guarantees easy to read, well-arranged documentation and accurate documentation.