SYSTEM ANALYSIS AND DESIGN (CSC 314)
(3 CREDIT UNITS)

Course Content

General Systems Concept: System Development Life Cycle
Planning: Project initiation, project management
Analysis: Fact gathering techniques, data flow diagrams, Process description, data modeling
System Design: Architecture design, interface design, data storage design, program design, Structure Charts, form designs, security, automated tools for design
Implementation: Site preparation, system installation, system testing, system conversion, report writing and presentation, system documentation, post installation evaluation

OBJECTIVES

After going through this course, the students should be able to:

- define a system
- know the need for system analysis and design
- know the roles and skills of a System Analyst
- describe the basic elements in system analysis
- understand project initiation, planning and management
- explain the different phases of system development life cycle
- enumerate the components of system analysis
- explain the components of system design and implementation

SYSTEM ANALYSIS AND DESIGN

Introduction

System analysis can be defined as the study of a system and interrelation between the elements of the system. It is conducted for the purpose of studying a system or its parts in order to identify its objectives. It is a problem-solving technique that is used to improve the system and ensures that all the components of the system work efficiently to accomplish their purpose. It consists of the steps to identify the weakness in order to propose and adopt a new system or improve an existing system.

System design is a process of planning a new system or replacing an existing system by defining its components or modules to satisfy specific requirements. It is usually preceded by analysis because it requires thorough understanding of the old system to determine how computers can best be used in order to operate efficiently.

Hence, System Analysis and Design is a methodology required to develop and deploy a system in an organization to guarantee effectiveness and efficiency.

System

The word “system” is a broad concept, which refers to an organized relationship among functioning units or components. There are different kinds of system, which include; the transportation system, the communication system, the accounting system, the production system, the economic system, a business system and the computer system. The component of a system

may refer to physical parts (engines, wheels of car), managerial steps (planning, organizing, controlling) or a subsystem in a multi-level structure. The components may be simple or complex, basic or advanced. In any case, each component is a part of the whole system and has to do its own share of work for the system to achieve the desired goal.

A system is a collection of components that work together to realize some objectives. The objective of an organization will be fulfilled only if the elements of that organization work together with proper planning and system. Organizations rely heavily on computer-based system for speed and efficiency. Basically, there are three major components in every system, namely input, processing and output.

Basic Characteristics of a System:

- Structure: This is defined by parts and their composition;
- Behavior: This involves inputs, processing and outputs of material, or information;
- Inter-connectivity: These are the relationships that exist among the different components.

In Computer Science, we are more concerned with the computer-based systems. Computer-based systems consist of all components necessary to capture, process, transfer, store, display, and manage information. These components include software, hardware, processes, networks, firmware, storage devices, and humans (users). The computer-based systems are classified into various types, which include:

- Transaction Processing System (TPS)
- Management Information System (MIS)
- Decision Support System (DSS)
- Office Automation System (OAS)

1. Transaction Processing Systems (TPS)

A transaction processing system is a computer-based information system that captures, classifies, stores, maintains, processes, updates and retrieves transaction data for record keeping and input for other systems. The most fundamental computer-based systems in organizations are used mainly used for the processing of business transactions. A transaction is any event or activities that are carried out in organizations, examples, placing orders, billing customers, hiring of employees, etc. The types of transaction that occur vary from organization to organization. Transaction processing systems provide speed and accuracy and can be programmed to follow routines without any inconsistency. Their main goal is to help improve routine business activities of organizations.

2. Management Information System (MIS)

Management Information System is more concerned with management functions. MIS is an information system that can provide all levels of management with information essential for the running of smooth business. This information must be as relevant, timely, accurate, complete and concise. Data processing by computer has been extremely effective because of several reasons;

the main reason is that huge amount of data relating different transactions are processed very quickly. In the past, most of the computer applications were concerned with record keeping and the automation of routine clerical processes. However, in recent years, computer applications are more focused on providing information for policy making, management planning and control purposes. MIS provides information to management of organizations for decision making, control, coordination, analysis, etc.

3. Decision Support System (DSS)

A decision support system is an information system that supports business or organizational decision-making activities. It analyzes massive amounts of data, compiling comprehensive information that can be used in decision-making. This is a kind of information system that offers information that may not be predictable; these kinds of information are not required often. These systems do not produce regularly scheduled management activities; instead, they are designed to respond to rapidly changing problems that may be not easily predicted in advance. They are used for decisions that are not of recurring nature and non-routine; thus, DSS assist managers to make unstructured or semi-structured decisions. A decision is considered unstructured if there are no clear procedures for making the decision and most of the factors to be considered in the decision cannot be readily identified in advance. Judgment of the manager plays a vital role in decision-making where the problem is not structured. The decision support system supports, but does not replace the judgment of managers.

4. Office Automation Systems (OAS)

Office automation systems are the common and most rapidly expanding computer-based information systems. They are developed with the goal of increasing efficiency and productivity in organizations. They are mainly used by office workers such as typists, secretaries, administrative assistants, professionals, managers, etc. to improve services. Many organizations have taken the first step towards office automation; this involves the use of word processing equipment to facilitate typing, storing, editing, revising and printing of text documents. An office automation system can be described as a multi-functional, integrated computer-based system that allows many office activities to be performed in an electronic mode.

Role of Systems Analyst

System Analyst assesses users' interaction with technology and business functions by systematically examining the input, processing and the output of information with the intention of improving organizational processes to increase efficiency. A system analyst commonly acts as systems consultant to organizations and thus, may be hired specifically to address information systems issues within an organization. A system analyst must be able to work with people of all categories and must be widely experienced in working with computers. The analyst plays many roles, sometimes balancing several at the same time. Each organization needs to define the specific roles and responsibilities that an analyst plays in their organization. Basically, the role of system analyst is to help organizations understand the challenges related to developing and using a system to ensure that their expectations are well represented.

Roles and responsibilities of an analyst include:

- Analyzing and understanding the status of current processes to ensure that the context and implications of change are understood by the client and the project team.

- Developing an understanding of how the solution will impact present and future business needs.
- Identifying the all requirements and understanding how they will help to determine the validity of the requirements.
- Properly documenting all the identified requirements.
- Developing a requirement management plan and disseminating the plan to all stakeholders.
- Working with the clients to prioritize and rationalize the requirements.
- Helping to define acceptance criteria for completion of a software development project.

Qualities of a System Analyst

A system analyst must possess a wide range of qualities, these include:

- A system analyst is a problem solver. An analyst is a person who views the analysis of problems as a challenge and enjoys developing workable solutions. When necessary, the analyst must be able to systematically tackle the situation at hand through skillful application of tools, techniques, and experience. The analyst must also be a communicator capable of relating meaningfully with other people over extended periods of time with the goal of solving problems.

- Systems analysts need to be able to understand human needs in interacting with technology, and they need enough computer experience to understand the capabilities of computers, to gather information requirements from users, and to communicate what is needed to programmers. They also need to strictly adhere to personal and professional principles to help them shape client relationships.

- A system analyst should be well educated and smart because they are expected to work with people at all levels in every aspect of organizations. Thus, they must possess the knowledge that will enable them work with different individuals and gain their confidence. Analysts must be smart and equipped with the ability to quickly understand how people carry out their task and provide better ways.

- A system analyst must be a self-disciplined and self-motivated individual who is able to manage and coordinate other people, as well as innumerable project resources. System analysis is a demanding career due to ever changing requirements as a result of very dynamic technology.

Skills of a System Analyst

Generally, a system analyst must have a good set of skills to be successful. These include:

- Interpersonal Skills: Analysts must have good verbal and written communication skills, including active listening skills. They should be well organized and equally have the ability to build effective relationships with clients to develop joint vision for a project. They also assist project managers by managing client expectations through careful and proactive communications regarding requirements and changes. They also ensure that stakeholders know the implications of their decisions and providing options and alternatives when necessary.

- Analytical Skills: These include the ability to gather and analyze information as well as identify problems and solutions to the problems. Systems analysts gather information about an organization's needs and use analytical and problem-solving skills to identify a solution.

- Technical Skills: A systems analyst ought to have good knowledge of certain technical areas like programming languages, information technology infrastructure, and database administration and management. An analyst should be up-to-date with modern development and system design tools in addition to extensive knowledge about new technologies.

- Management Skills: A systems analyst works with several individuals in different departments of organizations, so they need good managerial skills for effective resource management. An analyst should have the ability to understand users jargon and practices, project management, risk management and understand the management functions thoroughly.

Systems Development Life Cycle (SDLC)

Systems Development Life Cycle (SDLC) is a systematic approach that is used to design and build high-quality software. It generates a structure for designing, creating and delivering quality software based on customer requirements and needs. Its primary goal is to minimize project risks through advance planning to produce high-quality products that are effective and efficient. It involves different phases that comprise a detailed plan that describes how to develop, replace and maintain a system. Each phase builds on the results of the previous one. Adhering to the SDLC standard enhances development speed and minimizes project risks and costs associated with alternative methods.

SDLC is a framework that identifies all the activities required to research, build, deploy, and often maintain an information system. It is a conceptual model which includes policies and procedures for developing or altering systems throughout their life cycles. It helps in establishing a system project plan, because it gives overall list of processes and sub-processes required for developing a system. In other words, we can say that various activities put together to develop a viable system is referred to as system development life cycle. In software engineering, it is also referred to as software development life cycle.

Importance of SDLC

i. It provides a standardized framework that defines activities and deliverables.
ii. It aids in project planning, estimating, and scheduling.
iii. It makes project tracking and control easier.
iv. It increases visibility on all aspects of the life cycle to all stakeholders involved in the development process.
v. It increases the speed of development to ensure timely delivery of products.
vi. It decreases project risks, management expenses and the overall cost of production.

Phases of SDLC

The following are the different phases of system development life cycle:

a) Planning

b) Analysis (Feasibility Study)
c) System Design
d) Implementation (Coding)
e) Program Testing
f) Deployment
g) Maintenance

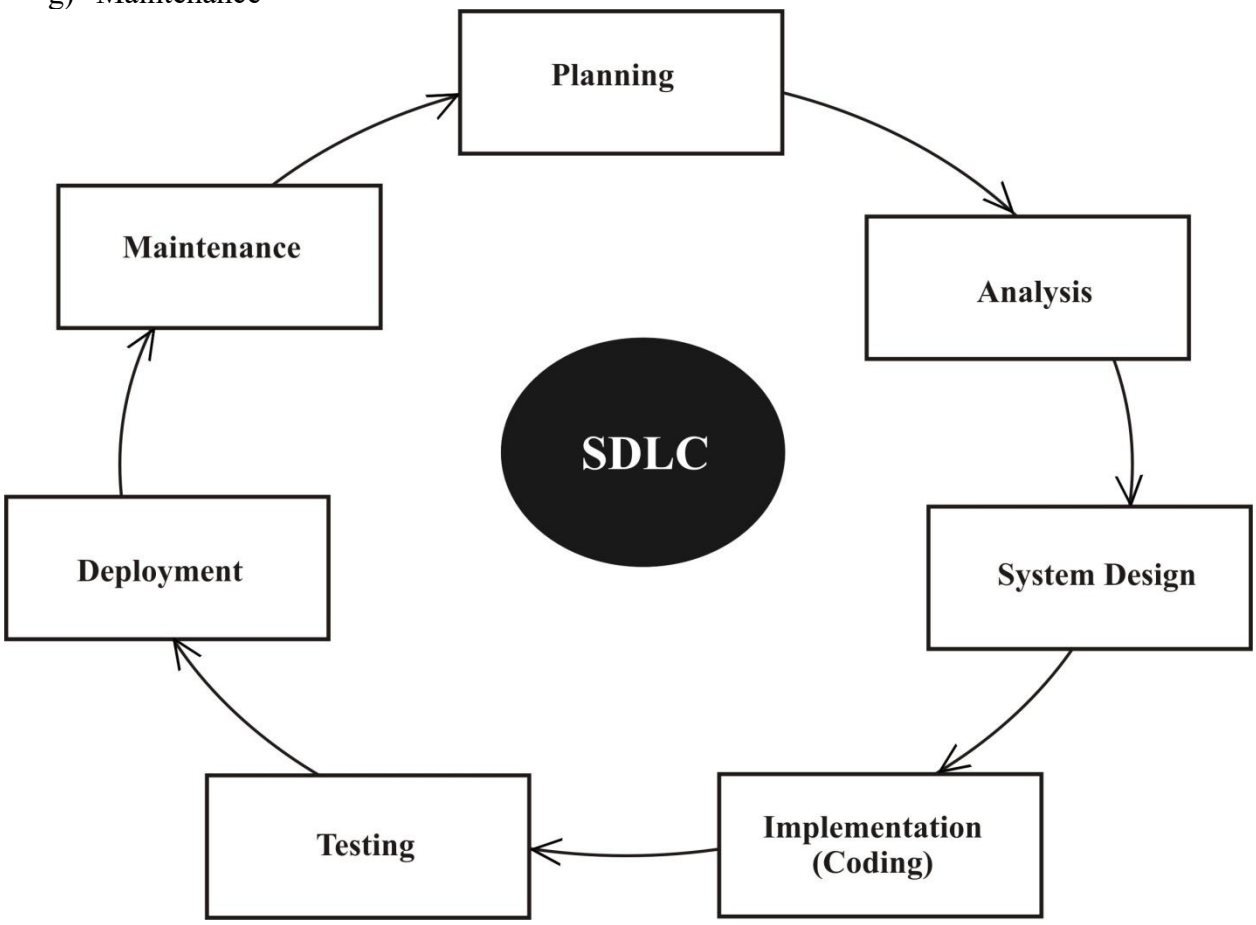


Figure 1: Systems Development Life Cycle

a) Planning: This can also be referred to as the preliminary investigation or preliminary system study. It is the first stage of system development life cycle. It is a brief investigation of the system under consideration; it gives a clear picture of the actual physical system. It is imperative to fully evaluate the existing system and identify the problems. This can be done by interviewing users of the existing system and consulting with support personnel. It also involves determining the resources and costs needed to complete the project, as well as estimating the overall total cost for developing the new system.

In practice, the initial system study involves the preparation of a system proposal which lists the problem definition, objectives of the study, terms of reference for study, constraints, expected benefits of the new system, etc. based on user requirements. The system proposal is carried out by the System Analyst. Finally, the planning process clearly defines the outline of system development. The new solution, which is the system under consideration should then be put in writing and approved by all stakeholders. There is the possibility that the proposal may be

accepted and the cycle proceeds to the next stage; or accepted with modifications or out rightly rejected.

b) Analysis: This involves involved a detailed study of the current system, leading to specifications of a new system. It is a detailed study of various operations performed by a system and their relationships within and outside the system. During analysis, data is collected on the available files, decision points and transactions handled by the present system. Interviews, on-site observation and questionnaire are tools used for system analysis. Using the following steps it becomes easy to draw the exact boundary of the new system under consideration. Is also involve carrying out feasibility study to determine the viability of the system; this include technical feasibility, economic feasibility, operations feasibility and others. The organization evaluates the proposed system on the basis of its cost, its flexibility and all other criteria. If the proposed system is not satisfactory, an alternate solution is proposed.

c) System Design: This stage of system development involves building a model of the system; it is an essential precursor to the actual system development. It involves extensive prototyping; it is sometimes referred to as logical design. The project team defines the structure of system components as well as key elements of the system by defining the interfaces that will exchange data within the workflow. It is very common for the project teams to use UML diagrams and other design tools in this phase to design the systems architecture. Prototyping tools, which now offer extensive automation and AI features can also be used to simplify the design. These tools ensure that best practices are strictly adhered to. Once the design specification is prepared, all the stakeholders will review this plan and provide their feedback and suggestions. It is absolutely mandatory to collect and incorporate stakeholders input in the document, as a small mistake can lead to cost overrun. System design involves specifying the following; architecture design, database design, interface design, specification of all system components.

d) Implementation (Coding): This is the stage where developers actually write code and build the application according to the earlier designed model and outlined specifications. Developers could follow any coding guidelines as defined by the organization and utilize different tools such as compilers, debuggers and interpreters. It involves coding, integration of system components, development of databases, creation of system interfaces. It entails writing a program based on the design using a programming language. This is the phase where the programmer converts the program specifications into computer instructions known as program. Developers choose the right programming language based on the project specifications and requirements. Programming languages can include Java, Python, C#, C++, PHP, and more. Activities involved in this stage include coding, integration of system components, development of databases, creation of system interfaces.

e) Testing: This is software testing conducted on a completed system to evaluate the overall performance. It is used to determine if a system meets its actual specified requirements and if it is suitable for delivery to end-users. Testing is carried out prior to deployment; it helps to detect likely issues for modifications. It is an important phase in SDLC that is used to certify a successful software project. The functionality of the entire system is tested to ascertain that the system meets customer requirements. Any issue or defect detected after testing is communicated to the developers for possible fixing and re-testing. This process continues until the software is stable, bug-free and working according to the user requirements. Many automated testing tools are available to make it more efficient. Testing is intended to increase the quality of the resulting software as well as the efficiency of the overall development process.

f) Deployment: At this state, the new system is integrated into its environment and installed in the actual machines where they are intended to be used. Deployment entails installation of the new software and providing all the hardware and software required to operate the system. It includes implementation of cloud services, hardware, monitoring systems, data configuration, drivers, security measures, etc. It also involves training the users of the system and employing appropriate change over method to ensure that the new system functions as expected.

g) Maintenance: System maintenance starts during the effective use of the product by end users. Obtaining feedback from end-users guarantees that likely issues are highlighted and taken care of. Providing updates and modifications to the software system after release ensure that the system better matches the needs of users. It entails response to errors that are encountered when using the system. Features could be added or modified to make the system more effective and efficient.

System maintenance may include tracking and monitoring of the system's security, eliminating potential risks and threats, assembling a list of functionalities that need to be updated, adapting the system to environmental and technological changes as well as new business requirements. Automated monitoring tools, which continuously evaluate system performance and detect errors can assist developers with regular quality assurance check, this is also known as instrumentation.

PROJECT MANAGEMENT

A project is an important and planned piece of work that is intended to build or produce something new (a solution) or deal with a problem. It is usually a temporary or one-time endeavor undertaken to create a unique product or service. Project management is the application of processes, methods, skills, knowledge and experience to achieve specific project objectives according to the project acceptance criteria within agreed parameters. Project management has final deliverables (a product of a development process) that are constrained to a finite timescale and budget.

Project management is aimed at producing an end product that will significantly improve the performance of an organization. It is the initiation, planning and control of a range of tasks required to deliver the end product. Projects that require formal management are those that:

- produce something new or altered, tangible or intangible;
- have a finite time frame: a definite start and end;
- are likely to be complex in terms of work or groups involved;
- require the management of change;
- require the management of risks.

Building Blocks of Project

- **Objectives:** A must have objectives. It is one of the important tasks of project managers to see that the projects are manage meet their objectives. Let us now discuss the objectives of projects.
- **Time:** scheduling is a collection of techniques used to develop and present schedules that show when work will be performed. Failure to operate within a time frame may delay the project completion time, which could increase the cost.

- **Cost**: This is concerned with funds required to manage a project till completion. Financial expenditures attached to a project are usually controlled by budget. The budget sets a limit as to the amount of funds a project can consume.
- **Quality**: The main goal of software development is to deliver high quality products. The resultant software products should match the user requirement and organizational goals after completion.

Project Team

A software project team is made up of people that are usually involved in system development projects, they include:

- System Analyst
- Programmer
- Project Team Leader
- Database Administrator (DBA)
- System Manager
- End User

System Analyst: System Analyst is the top-level employee in the System Development Life Cycle. It is the role of a system analyst to analyze an existing system in order to develop a new system required by an organization for better performance as well as carry our feasibility study to determine the viability of a new system. So, a system analyst plays a vital role in adopting effective and efficient system for an organization.

Programmer: Programmer works under the system analyst during system development. It is the duty of the programmer to implement the system based on the new system specification and design. This involves writing the code for the system using specific programming language. The programmer is also involved in maintaining, debugging and troubleshooting systems to ensure the smooth running of the system. A programmer should be familiar with the programming language selected for development of the new system.

Project Team Leader: Project team leader coordinates all the activities during system development and relates with all the persons involved in the system development. Project team leader should have all the information and knowledge about the development of the system. The ability to direct the entire project team like programmers, managers, database administrator and as well as the system analyst is imperative.

Database Administrator: Database administrator is responsible for creating, managing, maintaining, securing and operating databases. A DBA should have the ability to understand and manage the overall database environment. Database administrator plays vital role in the system development process. Database is very important for storage, management and retrieval of data.

System Manager: Systems manager is an IT professional who oversees the activities that involve the computer systems. A systems manager is responsible for monitoring the operations of the information technology department, evaluating staff performance, developing strategic procedures to maximize productivity, and identifying business opportunities that would generate more revenues and profitability for the company. It is the responsibility of the systems manager to inspect networks, conduct regular diagnostic tests (troubleshooting), and install updates and utilities to ensure efficiency. A systems manager must have excellent knowledge of Information

Technology, as well as mastery of programming to be able to maintain the security of database and network in organizations.

End User: End user is a person who uses the developed system or deliverables on a regular or daily basis to accomplish specific tasks. End users typically do not possess the technical understanding or skill of the product designers; they could be employees of organizations or the general public, who use the software to perform tasks. End users provide feedback that helps to determine if the goal of the developed system was met. They also help in capturing the requirements of the new system during the analysis of the existing system.

PROJECT INITIATION

Project initiation is the first step in starting a new project. During the project initiation phase, you establish why you're doing the project and what business value it will deliver—then use that information to secure buy-in from key stakeholders.

During the project's initiation phase, the project manager will lay the groundwork to establish every aspect of the work that will be part of the project; this could includes things such as:

- What the project
- Why the project is important
- Specifying the objectives
- What constitutes failure or success
- What business value the project will aim to deliver

During this stage, the organization identifies the objectives, scope, purpose and deliverables to be produced.

- Preliminary Project Acceptance
- System request is reviewed by approval committee
- Based on information provided, project merits are assessed.
- Worthy projects are accepted and undergo additional investigation – the feasibility analysis.

FACTS GATHERING TECHNIQUES

Different methods could be employed to gather the required information during analysis. There are various methods of data collection; choice of the method depends on the type of system involved; the most common methods include:

- Interview
- Observation
- Questionnaire
- Online Survey

1. Interview

Interview is a planned meeting during which information can be obtained from another person. It is a structured conversation where one participant asks questions, and the other provides answers. The analyst should have the basic skills needed to plan, conduct and document interviews

successfully. It is the most common method for data collections /facts gathering. Interviewing the prospective users can provide the valuable accurate information about a system. The system analyst can interview any personnel in an organization and should create the environment for interview setting time for whom, when, and about what.

Tips for a Successful Interview

i. Be punctual. This often means 10-15 minutes early. Interviewers are often ready before the scheduled time.
ii. Have the questions prepared in advance.
iii. Have a reliable instrument for recording.
iv. Be formal and stay focused.

The Procedures for Interview

The interview process consists of several distinct steps that help to obtain required information from respondents. The exact content of the steps of the interview process may vary based on the specific objectives of the interview.

i. **Determine the People to Interview:** To get an accurate picture of the system, it is important to select the right people to interview and ask them the right questions. People who relate with the system are usually selected for the interview. There are two sub categories of interview; structured and unstructured. Structured interview is a formal interview where fixed questions are asked and specific information is obtained while unstructured interview is more or less like a casual conversation where in-depth areas topics are covered and other information apart from the topic may also be obtained. Informal structures are usually based on interpersonal relationships and can be developed from the previous assignments, physical proximity, unofficial procedures, or personal relationships. In an informal structure, some people may have more knowledge than others; this could help determine the people to interview.

ii. **Establish Objectives for the Interview:** It is imperative to determine the objectives of the interview, the general areas to be discussed are determined and then the facts to be gathered are listed. The person to be interviewed is important; for example, upper-level managers can provide the better insight and can help in understanding the system better. Specific details about the processes are best learned from people who actually work with the system on a daily basis (the end users). In the early stages of system analysis, interviews usually are general. As the fact-finding processes continues, however, the interviews should begin to focus on topics that are more specific. Interview objectives also vary at different stages of investigation. By setting specific objectives, a framework can be created to help decide what questions to ask and how to phrase the questions.

iii. **Prepare for the Interview:** Careful preparation is essential because it is an important meeting and not just a casual chat. Scheduling a specific day, time and location for the interview and placing a reminder call to confirm the meeting should be a good practice. It is also important to note that the interview is an interruption of the other person's routine. Business pressures might force a postponement of the meeting; when this occurs, the interview can be rescheduled as soon as it is convenient for both parties. The department managers should be informed of meetings with their staff members. Creating a list of interview questions will help keep on track and avoid unnecessary digressions.

iv. **Conduct the Interview:** Conducting an interview should begin with introduction, describing the project and explaining objectives of the interview should also be considered. During the interview, questions are presented in the order in which they are prepared and sufficient time is provided for the respondent to give thoughtful answers. Establishing a good rapport with the respondent is important, especially if this is a first-time meeting. If the respondent feels comfortable and at-ease, a more complete and honest answers is likely to be obtained. The primary responsibility of the interviewer during the interview is to listen carefully to the response. It is ideal to concentrate on what is being said and to perceive any non-verbal communication that may take place; this is known as engaged listening.

v. **Document the Interview:** Although there are pros and cons to taking notes during an interview, the accepted view is that note taking should be kept to a minimum (in case of writing). Writing down a few notes to jog memory after the interview is recommended. Extensive writing should be avoided as too much writing could distract the respondent and make it harder to establish a good connection. A recorder may be employed, which could be played back to help in the documentation. After conducting the interview, time should be set aside time to record the facts and evaluate the information. Studies have shown that some percentage of a conversation is forgotten within a specific time, so facts should be summarized by preparing a narrative describing what took place or by recording the response beside each question on the prepared question list.

vi. **Evaluate the Response:** In addition to recording facts obtained in an interview, it is good to also evaluate the answers obtained to identify any possible bias. For example, a respondent who tries to protect his or her own area or function might withhold valuable information. A respondent with different opinion about the current or future system might distort the actual facts. Some may answer the questions in an attempt to be helpful even though they do not have the necessary experience to provide accurate information on a particular issue. The accuracy of the information obtained can be determined as the interviewer can cross-check and clear the doubts.

2. Observation

With this method, the analyst visits the organization to personally observe and understand the flow of documents, working of the existing system, the users of the system etc. Sometimes a system can be better understood by just observing its operation. Seeing the system in action could provide additional perspective and a better understanding of the system and its procedures. Personal observation also allows an analyst to verify responds obtain using questionnaires and interviews and determine whether the procedures really correspond to what was obtained; so, it could serve as an additional measure to obtain more insight. Through observation, the accuracy of information obtained from questionnaires and interview could be validated.

Personal observation also can provide important advantages in later SDLC phases. For example, recommendations often are better accepted when they are based on personal observation of actual operations. Observations also can provide the knowledge needed to test or install future changes and can help build relationships with the prospective users of the system. Observing the employees working in the organization with the present situation of the system and its environment is very useful in data collection. It should be planned in advance by preparing a

checklist of specific tasks to be observed and likely questions that could be asked. The following issues should be considered when preparing a list:

i. Ask sufficient questions to ensure a complete understanding of the present system operation is established. A primary goal is to identify the methods of handling situations that are not covered by standard operating procedures.
ii. Observe all the steps in a processing cycle and note the output from each step.
iii. Examine each pertinent form, record, and report. Determine the purpose each item of information serves.
iv. Consider each person who works with the system and ask the following questions: what information is received from other people? What information is generated by this person's work? What tools are used in the process?
v. Talk to the people who receive current reports to ensure that the reports are complete, timely, accurate, and in a useful form.

3. Questionnaires

Questionnaire is a document containing a number of standard questions that could be administered to many individuals. It can be a valuable tool in a system development project; it is usually employed to obtain information from a large number of people. A typical questionnaire starts with a heading, which includes a title, a brief statement of purpose, the name and the telephone number of the contact person, the deadline date for completion, and how and where to return the form, and then the questions and possible options to select from.
The heading is usually followed by general instructions that provide clear guidance on how to answer the questions. When designing a questionnaire, the most important rule is to make sure that the questions collect the right data in a form that it can be use to advance fact-finding. Listed are some additional ideas to keep in mind when designing questionnaires:

a) Keep the questionnaire brief and user-friendly.
b) Provide clear instructions that will help answer all anticipated questions.
c) Arrange the questions in a logical order, going from easy to more complex topics.
d) Phrase questions to avoid misunderstandings; use simple terms and words.
e) Try not to lead the response or use questions that give clues to expected answers.
f) Limit the use of open-ended questions that will be difficult to tabulate.
g) Limit the use of questions that can raise concerns about job security or other negative issues.
h) Include a section at the end of the questionnaire for general comments.
i) Test the questionnaire whenever possible on a small group before finalizing it.

4. Online survey

Online survey is one of the most popular data-collection sources, where a set of survey questions are sent out to a target sample, and the members of this sample can respond to the right questions through digital platforms. It is sometimes also referred to as online poll. Respondents can receive surveys via various mediums such as email, embedded in websites, social media, application etc. It enables gathering information from a wider audience or respondents. It also allows data analysis to be performed easily and quickly.

Respondents have more anonymity with online surveys so they provide more valid and candid answers. They are also more likely to give honest and open answers which provide more

accurate data. The same approaches used in the aforementioned methods can still be applied in online survey; the only difference is that online survey is carried out on the internet platform using digital devices and network.

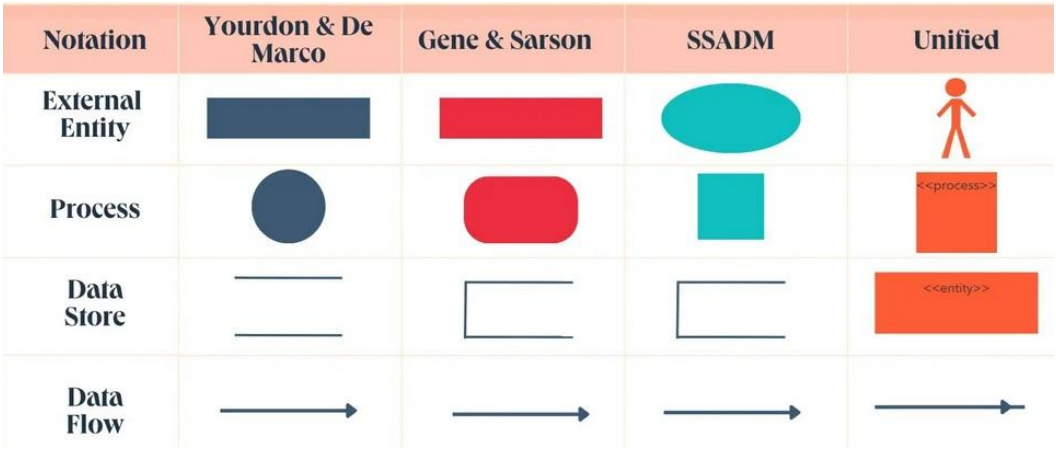
DATA FLOW DIAGRAM (DFD)

Data Flow Diagram (DFD) is used to represented the flow of data through a process in a system. It also gives insight into the inputs and outputs of each entity and the process itself.It is a graphical tool, useful for communicating with users, managers and other personnel. It is useful for analyzing existing and proposed systems.

The DFD provides information about the inputs and outputs of each entity and the process itself. A data flow diagram has no control flow — there are no decision rules and no loops. Specific operations based on the data can be represented by a flowchart. There are several notations for displaying data flow diagrams. The data flow diagram is a graphical tool that is part of structured analysis and data modeling. When using object-oriented design, the activity diagram typically takes over the role of the data flow diagram.

Components of DFD

There are four main standard notations that are used in DFD, these symbols represent entities, data flow directions, processes (inputs/outputs) and storage locations. There are different types of notations that are used for representations in DFD, which include Yourdon & De Marco, Gene & Sarson, SSADM, and Unified. They all use the same labels (names) and similar shapes to represent the four main elements of DFD.



a) External Entity: External entities are also known as actors or terminators. These are components that exist outside the system and interact with the system; they are responsible for sending or receiving data. In other words, they are the sources and destinations of the inputs and outputs of the system. They might be an outside organization or person, a computer system or a business system which does not belong to the system being modeled. They are represented on the DFD diagram as ovals drawn outside of the system boundary, containing the entity name and an identifier.

b) Process: A process is part of a system that transforms inputs to outputs. The symbol of a process is a circle, an oval, a rectangle or a rectangle with rounded corners (according to

the type of notation). The process is named in one word, a short sentence, or a phrase that clearly to expresses its essence.

c) Data Store: This is used to store the data in the system such as a file or database. A data store is represented in the data flow diagram by a rectangle, containing two locations. The small left-hand box is used for the identifier, which comprises a numerical reference prefixed by a letter. The main area of the rectangle is labeled with the name of the data store. Brief names that reflect the content of the data store are recommended.

d) Data flow: Data flow shows the transfer of information from one part of the system to another. It is depicted on the diagram as a directed line drawn between the source and recipient of the data, with the arrow depicting the direction of flow. Data-flows between external entities are depicted by dashed, rather than unbroken, lines. It links processes, data stores and entities.

Rules for Creating DFD

i. Each process should have at least one input and an output.
ii. Each data store should have at least one data flow in and one data flow out.
iii. Data stored in a system must go through a process.
iv. All processes in a DFD go to another process or a data store.
v. Entity names should be comprehensible without further comments
vi. Processes should be numbered for easier mapping and referral to specific processes.
vii. DFD should be clear, as the maximum number of processes in one DFD is recommended to be from 6 to 9.

DATA MODELING

Data modeling is a process used to define and analyze data requirements needed to support business processes within the scope of corresponding information systems in organizations. The process of data modeling involves professional/experts working closely with business stakeholders, as well as potential users of the information system. Thus, a data model is an abstract model that allows the building of conceptual model and setting relationships between data items.

Types of Data Models

There are three different types of data models that are used to implement database in the life-cycle of a software development project. They include:

- Conceptual Data Model
- Logical Data Model
- Physical Data Model

Conceptual Data Model: A conceptual data model is a high-level description of information needs underlying the design of a database. It typically includes only the main concepts and the main relationships among them. It maps out the kinds of data that are needed, how different business entities interrelate and associated business rules. It describes the structure of the whole database for a group of users but hides the internal details of physical storage. It targets the description of entities, data types, relationships and constraints. The conceptual model is

translated into a logical data model, which documents structures of the data that can be implemented in databases. Implementation of one conceptual data model may require multiple logical data models.

Logical Data Model: A Logical data model shows how data entities are related and describes the data from a technical perspective. It deals with a specific problem domain expressed independently of a particular database management product or storage technology but in terms of data structures such as relational tables and columns, object-oriented classes, or XML tags. It represents the abstract structure of a domain of information. It is basically used in business processes that seek to capture things of importance to an organization and how they relate to one another.

Physical Data Model: A physical data model (or database design) is a representation of a data design as intended to be implemented in a database management system. Physical models are specific to the database management system (DBMS) or application software that will be implemented. It defines the structures that the database or a file system will use to store and manage the data. These include tables, columns, fields, indexes, constraints, triggers and other DBMS elements. Database designers use physical data models to create designs and generate schema for databases.

ARCHITECTURE DESIGN

Institute of Electrical and Electronics Engineers (IEEE) defined architectural design as the process of specifying a collection of hardware and software components and their interactions to establish the framework for the development of a computer system. The architectural design of a system outlines its major elements, their structures (organization) and how they interact with each other. The primary goal of the architecture is to identify requirements that affect the structure of the application. A well-laid architecture reduces the risks associated with building a technical solution and builds a bridge between business and technical requirements.

It determines the structure and features of the components, data flow patterns, and how these components communicate with each other to share data. It specifies the components along with their inputs, outputs, functions, and the interaction between them necessary for developing a computer-based system.

UML Diagram

Unified Modeling Language (UML) is a standardized general-purpose modeling language in software engineering that is used across different programming languages and software development processes. It is used to represent the architecture, design, and implementation of complex software systems. It uses a standardized method for creating a system model and capturing conceptual ideas. UML diagrams divide a software system into components and sub-components and helps to keep track of the relationships and hierarchies within the software system. UML diagrams give a quick overview of the structure of the systems and illustrate patterns and relationships that help in the architectural design of the software. UML provides a more efficient design process which helps to detect potential issues early in the software architecture.

Common Types of UML Diagrams

a) Activity diagram: An activity diagram is a graphical representation that outlines all of a system's activities. It shows everything from start to finish, defining the various decision paths and steps that need to happen to move from one activity to the next. The steps can be chronological, branched, or simultaneous. This type of UML diagram is used to show the dynamic behavior of a system, but it can also be useful in business process modeling.

b) Sequence diagram: UML sequence diagram which is sometimes known as event diagram shows the order in which your objects interact. This includes the lifelines of your objects, the processes that interact with your objects, and the messages exchanged between the objects to perform a function. Software developers and business professionals often use sequence diagram to understand how to structure a new system or improve an existing process.

c) Class diagram: A UML class diagram is a fundamental building block of any object-oriented solution. It is a type of static structure diagram that describes the structure of a system by showing a system classes, attributes, operations, and the relationships among objects. Both software engineers and business managers use this interactive diagram to model different connections involved within a process. In the diagram, the class is represented by a rectangle. Each rectangle is split vertically into three sections. The top section has the name of the class, while the second and third sections provide details about class operations, behaviors, and attributes.

d) Use-case diagram: Use case diagram is a graphical depiction of user's possible interactions with a system. It is used to represent the interaction between actors (users or external systems) and a system under consideration to accomplish specific goals. A use case diagram shows various use cases and different types of users in a system. By illustrating a system's functionalities and outlining the expected behavior, they help developers analyze the relationships between use cases and users. The use cases are often represented by ellipses (oval) while the actors are often shown as stick figures.

Data Storage Design

An important activity of the system design is designing the data storage component of the system. There are two main types of data storage formats: files and databases. Files are electronic lists of data that have been optimized to perform a particular purpose while a database is a collection of groups of related information.

In system design, file and database storage systems play pivotal role in managing and organizing data efficiently. These systems provide the foundation for storing, retrieving, and organizing information in applications, ensuring data integrity and accessibility. File systems handle structured and unstructured data, while database systems offer structured data management with advanced querying capabilities. Effective integration of these storage systems is essential for designing scalable and reliable software applications, making them integral components of modern system architecture. The commonest storage tool used in system design is the database.

- Database design is a process that helps to create, implement, and maintain a data management system. It is a framework that the database uses for planning, storing and managing data in organizations. Data and database design are the backbone of every

company. Consistency of a data is achieved when the database is designed in such a way that enables it store only useful and often most required data.

Interface Design

A user interface is that portion of the computer system that communicates with the user. Design of the user interface includes any aspect of the system that is visible to the user. In the past, computer users were mainly experts in computing, and interfaces consisted of jumper wires in patch boards, punched cards prepared offline, and batch printouts. Today, a wide range of non-experts use computers, and keyboards, mouse and graphical displays are the most common interface hardware. The user interface is becoming a larger portion of the software in a computer system—and a more important portion, as broader groups of people use computers. As computers become more powerful, the critical bottleneck in applying computer-based systems to solve problems is now more often in the user interface, rather than the computer hardware or software. The major goal of user interface design is to make the user's interaction as simple and efficient as possible, in terms of accomplishing user tasks. A good user interface cannot be applied to a system after it is built but must be part of the design process from the beginning. Proper design of a user interface can make a substantial difference in the following ways;

- Training time
- Performance speed
- Error rates
- User satisfaction

It defines a set of interface objects, actions, and their screen representations that enable a user to perform all defined tasks in a manner that meets every usability objective defined for the system. It starts with task analysis which understands the user's primary tasks and problem domain. It should be designed in terms of user's terminology and outset of user's job rather than programmers. To perform user interface analysis, the practitioner needs to study and understand four elements:

i. The users who will interact with the system through the interface
ii. The tasks that end users must perform to do their work
iii. The content that is presented as part of the interface
iv. The work environment in which these tasks will be conducted

User Interface (UI) is the first impression of a software system from the user's point of view; it is the point where interactions between humans and machines occur. Therefore, any software system must satisfy the requirement of user. UI mainly performs two functions:

- Accepting the user's input
- Displaying the output

It plays a crucial role in any software system. It is possibly the only visible aspect of a software system as:

i. Users will initially see the architecture of software system's external user interface without considering its internal architecture.

ii. A good user interface must attract the user to use the software system without mistakes. It should help the user to understand the software system easily without misleading information. A bad UI may cause market failure against the competition of software system.
iii. UI has its syntax and semantics. The syntax comprises component types such as textual, icon, button etc and usability summarizes the semantics of UI. The quality of UI is characterized by its look and feel (syntax) and its usability (semantics).

Software in different domains may require different style of user interface; for example, calculator needs only a small area for displaying numeric numbers, but a big area for commands, a web page needs forms, links, tabs, etc. Proper or good UI design works for the user's capabilities and limitations and not the machines. In designing suitable UI for any system, a good knowledge of the nature of the user's task and system environment are important.

PROCESS DESCRIPTION

A software process is the set of activities and associated outcome that produce a software product; these activities are mostly carried out by software engineers. There are four key process activities, which are common to all software processes. These activities are:

i. Software Specifications: The functionality of the software and constraints on its operation must be defined.
ii. Software Development: This involves series of activities that are employed to design, build, deploy and support any software system.
iii. Software Validation: The software must be validated to ensure that it meets specified objectives.
iv. Software Evolution: The software must evolve to meet ever changing user's needs.

Software Process Model

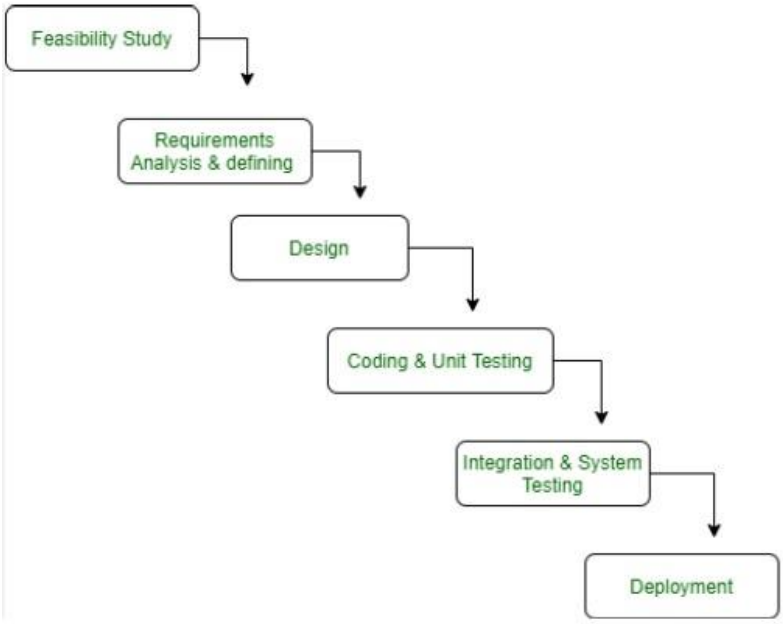
A software process model refers to the methods and techniques used to develop and maintain software products. It is a specified definition of a software process, which is presented from a particular perspective. Models are simple in nature, so a software process model is a simple idea or concept of the actual process. A software process model may contain activities, which are part of the software process, software product and the roles of people involved in software engineering.

Software processes involve the activities for designing, implementing, and testing a software system. A software model specifies the stages and order of a process. So, it can be seen as a representation of the order of activities of the processes and the sequence. A model defines the following:

- The tasks to be performed
- The input and output of each task
- The pre and post-conditions for each task
- The flow and sequence of each task

Types of Software Process Models

1. Waterfall Model



The waterfall model is a linear and sequential approach to software development. It is a breakdown of project activities into linear sequential phases, where each phase depends on the deliverables of the previous one and corresponds to a specialization of tasks. It has several phases: requirements gathering, design, implementation, testing, and maintenance. It requires that every stage of software development must be completed before moving to the next. It makes this model relatively rigid since all plans must be scheduled beforehand. On the other hand, the waterfall is easy to manage if the project requirements are clear. The approach is typical for certain areas of engineering design.

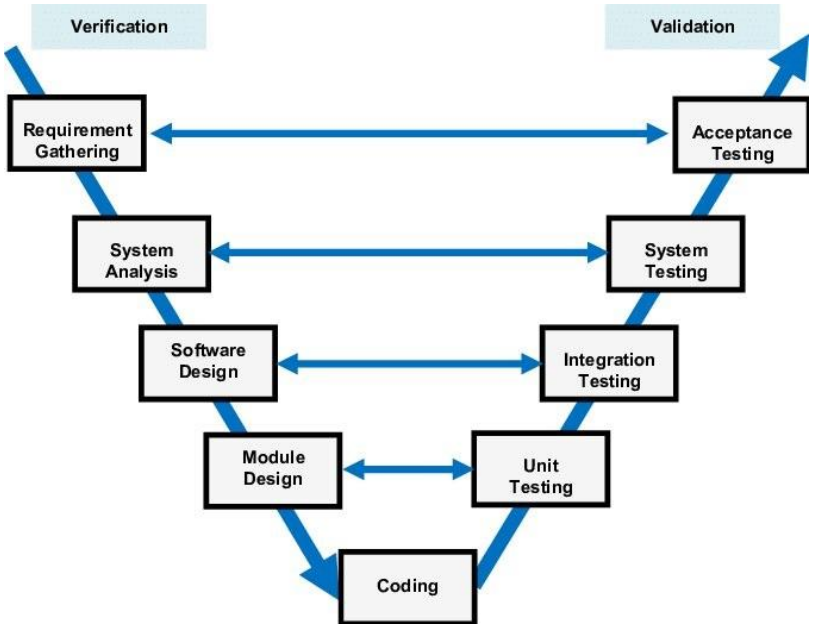
Advantages of Waterfall

i. Linear structure is easy to understand.
ii. It has well-defined stages and deliverables, so development progress is easily estimated and documented.
iii. It is scalable.

Disadvantages of Waterfall

i. It is only suitable for small, straightforward projects with well-defined requirements
ii. It is not flexible; corrections are limited to current phase.
iii. Working version of system is only available near the end.

2. V Model



The V-model represents a development process that may be considered an extension of the waterfall model. But instead of moving down in a linear path, the process steps are bent upwards after the coding phase to form the typical V shape. The V-Model demonstrates the relationships between each phase of the development life cycle and its associated phase of testing. The horizontal and vertical axes represent time or project completeness (left-to-right) and level of abstraction respectively.
It is also referred to as verification/validation model because it is associated with a testing phase for every corresponding phase and the next phase starts only at the completion of the previous state.

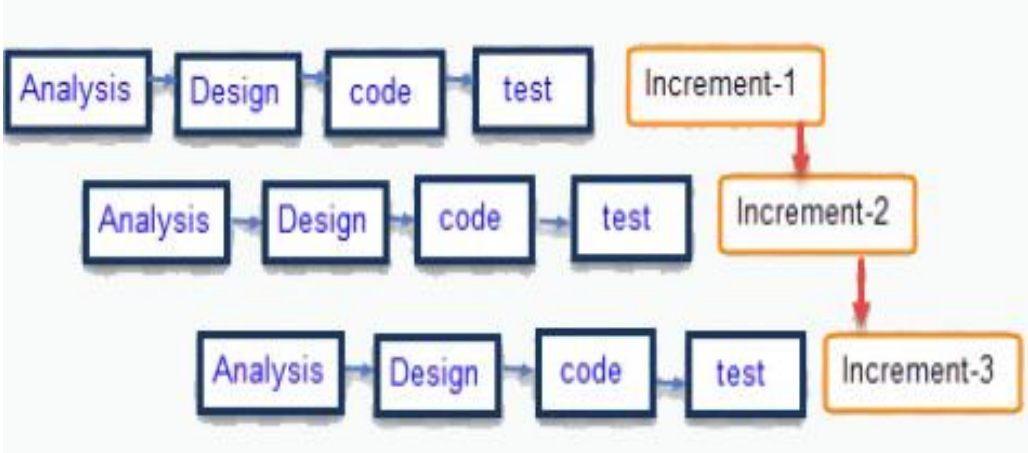
Advantages of V-model:

1. It is simple to understand and easy to use.
2. Testing activities happens before coding, which saves a lot of time and guarantees higher chance of success.
3. Defects are discovered at early stage of system development.
4. It works well for small projects where requirements are easily understood.

Disadvantages of V-model:

1. It is not flexible.
2. It is not suitable for complex projects.
3. It is not suitable for projects with unclear or changing requirements since making changes is difficult.

3. Incremental/Iterative Model



An incremental/ iterative software development process is where requirements are broken down into multiple independent modules of software development cycle. It is a method of software development where the model is designed, implemented and tested incrementally with a functionality added each time until the product is completed. Each iteration passes through the requirements, design, coding and testing phases. And each subsequent release of the system adds function to the previous release until all designed functionality is fully implemented. This model combines the elements of the waterfall model with the iterative idea of prototyping. It is suitable for projects with changing requirements, where feedback and improvement throughout the development process are desirable and flexibility is critical.

Advantages

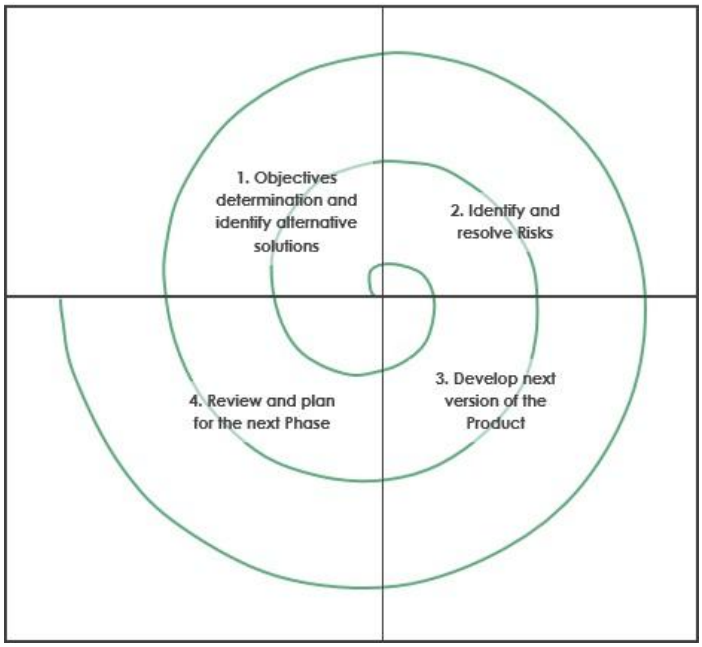
i. The delivery of working software is possible at the end of each iteration.
ii. It provides opportunities for feedback and improvement throughout the development process.

Disadvantages

i. It can be time-consuming.
ii. It requires a high level of involvement of the teams.
iii. It may require frequent updates to documentation and project plans.

4. Spiral Model: The spiral model, first described by Barry Boehm in 1986, is a risk-driven software development process model which was introduced for dealing with the shortcomings in the traditional waterfall model. The model looks like a spiral with many loops; the exact number of loops of the spiral is unknown and can vary from project to project.

This model supports risk handling approach to software engineering, highlighting the iterative nature of the development process. It comprises four phases: planning, risk analysis, engineering, and evaluation. Each iteration of the spiral model involves completing these four phases, with an increased emphasis on risk management as the project progresses.



Each loop of the spiral is called a phase of the software development process. The exact number of phases needed to develop the product can be varied by the project manager depending upon the project risks. As the project manager dynamically determines the number of phases, so the project manager has an important role to develop a product using a spiral model. It is suitable for projects with high levels of risk and uncertainty.

Advantages

i. It focuses on risk management and evaluation throughout the project
ii. It is flexible and adaptable to changing requirements

Disadvantages

i. It can be time-consuming and expensive
ii. It may not be suitable for small projects and projects with well-defined requirements
iii. It requires a high level of expertise in risk management

5. Agile Model: The agile model is an iterative and incremental software development approach emphasizing flexibility and collaboration between cross-functional teams. It consists of short time-boxed iterations called sprints, where the team develops and tests small project increments. Core values of this methodology include incremental and continuous delivery, customer involvement, and change responsiveness.

Agile is an umbrella term for a set of methods and practices based on the values and principles expressed in the Agile Manifesto that is a way of thinking that enables teams and businesses to innovate, quickly respond to changing demand, while mitigating risk. The agile solution can be implemented using many of the available frameworks such as Scrum, Kanban, Lean, Extreme Programming (XP) and etc.

The primary goal of agile model is giving the development team the ability to create and respond to change in order to succeed in an uncertain and turbulent environment. Agile software development approach is typically operated in rapid and small cycles. This results in more frequent incremental releases with each release building on previous functionality. Thorough testing is done to ensure that software quality is maintained.

Advantages

i. It is flexible and adaptable to changing requirements.
ii. It focuses on collaboration between teams and stakeholders.
iii. It enables early and continuous delivery of working software.

Disadvantages

i. It requires a high level of collaboration and communication.
ii. It may not be suitable for projects with rigid timelines or fixed budgets.
iii. It can be challenging to estimate the effort required for each iteration accurately.

6. Rational Unified Process

The Rational Unified Process (RUP) is a software process framework that provides guidelines, templates, and best practices for software development. It is based on iterative and incremental delivery of working solutions after each iteration.

Advantages

i. It provides a comprehensive framework for software development.
ii. It highlights the delivery of working software after each iteration.

Disadvantages

i. It may be time-consuming.
ii. It is expensive to implement.
iii. It might be too rigid for small projects.

FEASIBILITY STUDY

Feasibility study is a preliminary study undertaken to determine the viability of a project. It aims at objectively and rationally uncovering the strength and weaknesses of a system, opportunities and threats present in the natural environment, the resources required to carry out the project and ultimately the prospects for success after development. It is usually part of the analysis stage of the Software Development Life Cycle.

It provides an independent assessment that examines all aspects of a proposed project, including technical, economic, financial, legal, and environmental considerations. This information then helps decision-makers determine whether or not to proceed with the project. A software project must succeed several feasibility studies before it can continue. Most software project are carried out within defined budget and time constraints; this means that software project feasibility is a required activity in all information system development to ascertain that the project is worth investing in; i.e., the project will generate enough profit to justify the investment.

Types of Feasibility Study

1. Operational Feasibility: This is concerned with the impact of the new system on the organization's routine operations. The main purpose of operational feasibility is to gain an understanding of the extent the proposed system will solve the identified problems. It assesses how well the system will be used to provide services based on the requirements. It also assesses how easy it will be to operate and maintain the system after deployment. Basically, it helps to ascertain effectiveness and efficiency of the proposed system after it has been developed. If users have difficulty with a new system, it will not produce the expected benefits to its organization operations.

2. Technical Feasibility: The purpose of technical feasibility is to gain understanding of the ability of an organization to develop and implement the proposed system. Technical feasibility assesses the availability of required technology and infrastructure in an organization that will enable meeting the requirements. It addresses the available knowledge to analyze, design, implement, deploy and maintain the proposed system. It determines the availability of technical resources that are appropriate to use in developing the system. It helps to ascertains if the technical skills of the project team is capable of converting the ideas into working systems. It also evaluates available hardware and software resources required for the proposed system.

3. Economic Feasibility: This is concerned with determining the viability, cost and benefits associated with the project. It identifies the favorable economic advantages that the proposed system would bring to an organization. It is sometimes referred to as cost-benefit analysis. A new system should have the ability to increase revenue and generate more profit to be economically feasible. In addition, an organization should be able to recover all funds invested in a project from inception to conclusion and as well guarantee revenue generation from the system after deployment.

4. Schedule (Time) Feasibility: This assesses the probable completion period of a software project. The study should establish a start date, estimated duration, and completion date for each phase of the project. Schedule feasibility guarantees the likelihood that all potential time frames and completion date schedules can be met and that meeting the deadline will take care of the needs of the organization. It takes into account the time the project is going to take from start to completion. A project will fail if it takes too long to be completed before it becomes useful; thus, some projects are initiated with specific completion deadlines. It is also necessary to determine whether the deadlines are mandatory or desirable.

5. Legal Feasibility: This involves examining legal requirements such as contracts, data protection acts, social media laws, project certification, licensing, copyright issues, etc. consequently, a business entity could face expensive litigation if these are not checked and anticipated. Legal feasibility helps to determine whether the proposed system conforms to legal and ethical requirements.

Importance of Feasibility Study

i. It prevents committing resources and capital to an impractical project.
ii. It helps to discover new ideas that might completely change the scope of a software project.

iii. It improves the confidence of the project team to concentrate on the project with the awareness that likely issue have been taken care of.
iv. It helps validate the need for developing any software.
v. It helps to estimate the risk involved in developing any software.
vi. It helps to evaluate the success rate of the software.

SOFTWARE TESTING

Software testing is a systematic approach that aims to identify and rectify defects, errors, and potential issues in software during its development life-cycle. It is the process of executing a program or system with the intent of evaluating the capability of a program or system to determine if it meets the required objectives. Software testing entails verification and validation of the developed system. It involves checking whether the actual software product matches expected requirements and ensures that software product is defect free. A **software bug** is a failure or flaw in a program that produces undesired or incorrect results. It is an error that prevents a system functioning as it should. The following are likely causes of error in programs:

- Software complexity
- Programming errors
- Time constraint
- Software development tools
- Changing requirements

Reason for Software Testing

The purpose of software testing is to ensure the delivery of high-quality, reliable, and efficient software products. The primary goal of software testing is to validate the software's functionality, enhance its performance and improve the overall user experience. Listed are few reasons for software testing:

i. Validating the functionality of the software to ensure it meets the specified requirements and works as intended.
ii. Identifying and eliminating defects and errors that could negatively impact the software's performance and stability.
iii. Improving the reliability and robustness of the software to ensure consistent and accurate results.
iv. Enhancing the software's performance, making it more efficient and responsive even under heavy user loads.
v. Ensure the software is user-friendly and intuitive, offering a satisfactory user experience.
vi. Minimizing the risk of software failures and security vulnerabilities, protecting user data and sensitive information.
vii. Certifying that software products comply with industry standards, regulations and best practices.
viii. Reducing the overall development and maintenance costs by detecting and fixing issues early in the development process.

Testing Methods

Software testing methods are broadly classified into two:

- White box or Structural testing
- Black box or Functional testing

1. White Box or Structural Testing

White box testing is a form of testing that concentrates on the procedural details. It is concerned with checking the source code and the internal design of software. It examines the program logic. With this testing method, the internal structure is disclosed to detect faults. White box testing relies on the in-depth knowledge of the code and procedural design to drive the test case. It is widely utilized in unit testing to determine all possible paths within a module to execute all loops and to test all logical expressions. White box testing helps the software engineer to:

- Guarantee that all independent paths within a module have been executed at least once.
- Examine all logical statements on their true and false values.
- Execute all loops and test their operations at their limits.
- Check data structures to ensure their validity.

2. Black Box Testing or Functional Testing

Black box testing focuses on the overall functionality of the software. This testing method enables detection of flaws like incorrect or missing functions, errors in any of the interfaces, errors in data structures or databases and errors related to performance and program initialization or termination. It ensures that software works as intended for end-users. Black box testing aims at testing a given program's behaviour against its specification without making any reference to the internal structures of the program or the algorithms used.

Different Types of Testing

- Unit testing: This is a test of particular function or software modules. Unit testing requires detailed knowledge of the internal program design and code, is typically done by the programmer. Developers often use test automation tools such as NUnit, Xunit, JUnit for the test execution.

- Incremental integration testing: It is a continuous testing of an application as new functionalities are added; it requires that various aspects of an application’s functionality be independent enough to work separately before all parts of the program are completed.

- System testing: System testing is a form of testing based on overall requirements specifications. It embraces all combined parts of a system. This type of test involves examination of the whole computer system. All the software components, all hardware components and the interfaces are tested together. During the result analysis, it may be discovered that the outputs are do not match the expected output of the system.

SYSTEM CHANGEOVER

System changeover is the process of transitioning from one system to another. This could be a major or minor change, and may involve changes to personnel, processes, or technology. Changeover can be rapid or slow, depending on the method. The reason for applying changeover methods is to guarantee safe transition from old to new system. The four changeover approaches are direct changeover, parallel operation, pilot operation, and phased.

- **Direct Changeover**

  It involves fully implementing the new system while the old one is totally discarded immediately. With the direct changeover approach, the changeover from the old system to the new system occurs immediately the new system becomes fully operational. This strategy demands thorough testing and well-planned file creation and training strategies. All operations of the system must be understood at the moment of trying because the opportunity for gradual training and testing does not exist.

  Direct changeover involves more risk than other changeover methods. Regardless of how thoroughly and carefully testing and training can be conducted, some difficulties might arise when the system goes into operation. Problems can result from situations that not anticipated or from errors caused by users. A system also can encounter difficulties because real data typically occurs in much larger volumes than test data. Organizations often choose the direct changeover method for implementing commercial software packages because these applications involve less risk of total system failure. However, it is a less expensive changeover method because the IT group has to operate and maintain only one system at a time but basically, it is the riskiest method.

  Advantages of Direct Change Over

  i. It is cheaper to implement.
  ii. Implementation is immediate and fast.

  Disadvantages of Direct Change Over

  i. There is nothing to fall back on if the new system fails.
  ii. Staff training has to be completed before system deployment.
  iii. It can be time consuming and difficult.

- **Parallel Changeover**

  With the parallel operation changeover method, both the old and the new information systems operate fully for a specified period. Data input is in into both systems and output generated by the new system is compared to the equivalent output from the old system. When users, management, and the IT group are satisfied that the new system operates correctly, the old system is then eliminated.

  Advantages of Parallel Change Over

  i. The most obvious advantage of parallel operation is lower risk. If the new system does not work correctly, the old system can be used as a backup until appropriate modifications are made.

ii. It is much easier to verify that the new system is working properly under parallel operation than with direct changeover because the output from both systems are compared and verified.
iii. It also gives users time to get familiar with the new system and gain the confidence to use it.

Disadvantages of Parallel Change Over

i. It is the most expensive changeover method expensive since it demands twice the resources for running one system; example, electricity, data, etc.
ii. Users must work in both systems and temporary employees might be needed to handle the extra workload.
iii. Running both systems might place burden on the operating environment and cause processing delays.
iv. Parallel changeover is not practical if the old and new systems are technically incompatible or if the operating environment cannot support both systems.
v. It is also inappropriate if the two systems perform different functions or if the new system involves a new method of operations.

- **Pilot Changeover**

With the pilot changeover method, there is a complete implementation of the new system at a selected area of the organization. A new sales reporting system, for instance, might be installed in just one branch office, or a new payroll system might be installed in only one department. The group that uses the new system first is called the pilot site. During the pilot operation, the old system continues to operate in the entire organization, including the pilot site. After the system proves successful at the pilot site, it is implemented in the rest of the organization, usually using the direct changeover method. Therefore, pilot operation is a kind of semi-parallel operation that combines the parallel and direct changeover methods.

Advantages of Pilot Change Over

i. Restricting the implementation to a pilot site reduces the risk of system failure, compared to a direct changeover method.
ii. Operating both systems for only the pilot site is less expensive than a parallel operation for the entire organization.
iii. If a parallel approach is used to complete the implementation, the changeover period can be much shorter if the system proves successful at the pilot site.

Disadvantages of Pilot Change Over

i. The main disadvantage is that the process takes a long time to complete because phases need to be implemented separately. With a pilot changeover, the new system is tried out at a test site before launching it company-wide.
ii. It involves more risks of total system failure; issues like data lose may arise if the new system fails.

- **Phased Changeover**

  With the phased changeover method, the new system is implemented in stages or modules at a time until the whole system is implemented. It combines parallel and direct change-over strategies. The module can be a functional part of the system or a specific subsystem. Each sub-system is implemented until it succeeds that is when the next one is implemented. When a particular phase of the new system has been fully tested and approved, the next phase is introduced; gradually the entire old system will be completely replaced. This is sometimes referred to as phasing out.

  Advantages of Phased Change Over

  i. One advantage of phased changeover is that the risk of errors or failures is limited to only the implemented module.
  ii. Staff training can be done gradually.
  iii. Error can easily be detected.
  iv. Risk is minimized because if any part fails, the old system can still be accessed.
  v. It is also less expensive than full parallel operation.

  Disadvantages of Phased Change Over

  i. Phased changeover is not possible if the system cannot be separated easily into independent modules or parts.
  ii. If a large number of phases are involved, it will take a longer time to fully implement the system.

TRAINING

Training is essential in system deployment because it plays a key role in the adoption of the new product. It is critical that people adapt and change in line with the deployment, and that acceptance of change does not become a bottleneck for the deployment. This can be assured by incorporating appropriate training in the system deployment phase.

No system can be successful without proper training of users, whether it involves software or hardware. The success of a system development project can depend on whether people understand the system and know how to use it effectively. A training plan should be considered early in the system development process. As the documentation is created, the subsequent use of the material by users to understand the system should be considered. During the system deployment, it is essential to provide the right training to the right people at the right time. The first step is to identify who should receive training and what training is needed for proper utilization of the system.

Considerations for Training

The system analysts must consider the following with regards staff training:

- Who needs to be trained
- Who will carry out the training
- Method of instruction
- Location for the instruction
- Materials for the instruction

Possible Sources of training for users of information systems include:

- Vendors
- Systems analysts
- External paid trainers
- In-house trainers
- Other system users

DOCUMENTATION

Software documentation refers to all the technical and written documentation related to a software product. It contains information on the software development process and equally helps end-users make effective use of software. Basically, it provides information that describes the software product to the people who develop, deploy and use it. Documentation is one of the most important activities in the system development life cycle. It helps to ensure the continuity of a system. There are generally two types of documentation prepared for any system. These are: User or Operator Documentation and System Documentation.

1. User Documentation

This type of documentation includes the resources provided to end users and system administrators to guide them on how to use the product. It may also include information about special features of software, tips for use and troubleshooting guide. Common types of user documentation include Frequently Asked Questions (FAQs), tutorials, and support information. It makes it easier for users to navigate and familiarize themselves with the software. The user documentation is a complete description of the system from the user's point of view, detailing how to use or operate the system. It could also include major error messages likely to be encountered by the users. Examples are:

- Installation/Setup Guide: Getting started with certain products, like enterprise software applications, gadgets, fixtures, etc., requires some level of technical expertise. To this end, manufacturers provide such products with comprehensive installation or setup guides, including detailed step-by-step instructions to make the overall process user-friendly. With the help of these tutorials, businesses ensure that the end-users do not mess up when starting their journeys with their products.

- User Manual: A manual provides more in-depth information and instructions about software. It includes everything, from an instruction manual on how to install a product, to troubleshooting steps, and a breakdown of the user interface and/or the various features. Only a complete manual or set of manuals can help users to have a complete understanding of a software, going into all necessary details. User manual can be provided in hardcopy form and softcopy form on CD or online.

- Reference Guides: This is another common type of user documentation. This type of user documentation comes with most software products and is intended for more experienced end-users. A reference document provides information on the functionalities of different aspects or features of a software product such that any user with good knowledge of the software can quickly get information about certain features without having to skim through the entire user manual.

2. System Documentation

System documentation records all the core information about a product development process. It explains the product's functionality for future maintenance and upgrades from the developers' perspective. It tells the development team the technology used for creating this product, making it easy for them to upgrades the product in the future. It contains the details of system design, programs, coding, system flow, data dictionary, process description, etc. This helps to understand the system and permit changes to be made in the existing system to satisfy new user needs. Various types of system documentation are:

- Requirements documentation: This is the primary document of the software product. It is produced at the initiation of the project, and it states the functions of the software. It explains the purpose of the software, so it guides the development team through the creation of the software product. It includes hardware specifications, functionality requirements and compatibility. It identifies attributes, capabilities, characteristics or qualities of a system. It is the foundation for what will be or has been implemented.

- Administrative documentation: This type of documentation helps those responsible for managing computer systems and servers to maintain the software. It usually includes information on software installation, updates and functionality. It may also provide a description of software behaviour with different systems and what to do if it malfunctions. It basically provides administrative guidelines and product requirements for the software development team and project managers working on the software. It also may include status reports and meeting notes.

- Technical documentation: This describes how the software uses or builds upon existing technological concepts. As its name suggests, it is geared toward information technology (IT) specialists and software engineers who use it to ensure a quality user experience and earn the trust and loyalty of customers. Technical documentation includes elements such as application protocol interface (API) routes, which allow the software to communicate with the user's device, and software development kits, or a set of tools use by developers to create software. It involves documentation of code, algorithms, interfaces, and Application Programming Interface (API).

POST-IMPLEMENTATION EVALUATION

Once the new system is operational, two additional tasks need to be carried out: preparing a post-implementation evaluation and delivering a final report to management. A post-implementation evaluation assesses the overall quality of the information system. The evaluation verifies that the new system meets specified requirements, complies with user objectives, and achieves the anticipated benefits. In addition, by providing feedback to the development team, the evaluation also helps improve information system development practices for future projects. This task requires one-to-one communication with users so the information system staff can learn as much as possible about the strengths and weaknesses of the new system. A post-implementation evaluation should examine all aspects of the development effort and the end product. A typical evaluation includes feedback for the following areas:

- Accuracy, completeness, and timeliness of information of system output
- User satisfaction
- System reliability and maintainability

- Adequacy of system controls and security measures
- Hardware efficiency and platform performance
- Effectiveness of database implementation
- Completeness and quality of documentation
- Quality and effectiveness of training
- Accuracy of cost-benefit estimates and development schedules

The same fact-finding techniques that are used to determine the system requirements during the systems analysis phase could also be applied in a post-implementation evaluation. Evaluating a system could involve the following:

- Interviewing members of management and key users of the developed system.
- Observing users and computer operations personnel actually working with the new information system.
- Online survey to get feedback from actual users.
- Reading all documentation and training materials.
- Examining all source documents, output reports, and screen displays.
- Use questionnaire to gather information and opinions from a large number of users
- Analyzing maintenance and help desk logs.

Whenever possible, the post-implementation evaluation should be conducted by people who were not directly involved in developing the system. The evaluation usually is done by information system staff and users, although some firms use an internal audit group or independent auditors to ensure that the accuracy and completeness of the evaluation. Ideally, conducting a post-implementation evaluation should be standard practice for all information system projects. Sometimes, however, evaluations are skipped for various reasons, which include:

i. Users are eager to use the new system to complete their work.
ii. Information system personnel are assigned to other projects or qualified people are not available to perform the evaluation.
iii. In some organizations, management might not recognize the importance and benefits of a post-implementation evaluation.

Final Report to Management

At the end of software development process, a report is produced and sent to management, and this marks the end of systems development process. A report should include the following items:

1. Final versions of all system documentation.
2. Planned modifications and enhancements of the system that have been identified.
3. A recap of all systems development costs and schedules.
4. A comparison of actual costs and schedules with the original estimates.
5. The post-implementation evaluation result, if it has been performed.

AUTOMATED TOOLS IN SOFTWARE DESIGN

Automation tools play significant role in custom software development services, offering functionalities to streamline processes and improve efficiency. Automation software employs

various applications and features that can be used across industries with minimal human intervention required, they are used to turn repetitive tasks into automated actions.

Every type of automation tool has a distinct function within the software development life-cycle, optimizing operations and boosting output. Development teams may speed up delivery, increase quality, and collaborate effectively by utilizing the correct set of automation technologies.

Types of Automated Tools

1. Testing Tools: These tools automate the execution of test cases resulting in thorough and effective testing of software applications. Testing automation tools allow developers to design automated test scenarios that can repeatedly run to evaluate software functionality and find flaws. These scenarios can be created using visual interfaces or scripting languages. Popular testing tools for software development solutions automation include Selenium, Appium, and JUnit.

2. Deployment Tools: Packaging, setting, and deploying software applications across diverse environments is automated by deployment automation tools. These tools replace manual deployment, decreasing the possibility of human error and assuring reliable and consistent deployments. For instance, tools for deployment automation include Ansible, Bamboo, and Jenkins.

3. Continuous Integration (CI) Tools: CI tools facilitate the integration of code changes from multiple developers into a shared repository. They automate the build, test, and integration processes, enabling teams to detect and resolve integration issues early in the development cycle. CI automation tools in software development, such as Jenkins, Travis CI, and CircleCI, ensure that the software remains in a continuously deployable state.

4. Monitoring and Logging Tools: Software performance, behavior, and availability are carefully monitored and recorded using monitoring and logging tools. These automated data collection and analysis technologies let developers detect problems early and take proactive measures to fix them. Monitoring tools like Nagios, New Relic and Datadog provide real-time insights into application performance while logging tools like the ELK Stack (Elasticsearch, Logstash, and Kibana) assist in gathering and analyzing logs.

5. Task and Project Management Tools: Task and project management tools automate software development projects' tracking, scheduling, and collaboration aspects. These tools enable teams to manage and prioritize tasks, assign responsibilities, and monitor progress efficiently. Examples of task and project management tools include Jira, Trello, and Asana.