Introduction to Systems Analysis and Design

A. What is a System Analysis?

System analysis is a process in which the procedures and methods of a system are examined to identify their objectives and efficiencies. The goal is to improve system performance by finding solutions to problems or inefficiencies and ensuring that the system meets the intended requirements.

Analogy:

Think of system analysis as being similar to a doctor examining a patient. Just as a doctor gathers information through various tests and observations to diagnose a problem, a systems analyst studies an organization's current procedure and information systems. The doctor, like the analyst, seeks to understand the underlying issues that cause symptoms (problems or inefficiencies) and prescribes treatment (system improvements or new systems) that will help the patient (the organization) function more effectively. In both cases, thorough investigation and proper diagnosis are critical before effective solutions can be implemented.

Primary Purpose of Systems Analysis

- To improve the efficiency, effectiveness, and functionality of a system.
- It helps organizations optimize their processes, make informed decisions, and achieve their goals.

B. Roles of a Systems Analyst?

A **systems analyst** acts as a bridge between business problems and technology solutions. Here are the key roles they typically fulfill:

1. **Problem Identifier:** Identifies issues within existing systems or processes by gathering and analyzing data, and engaging with stakeholders to understand their needs and challenges.

2. **Requirements Gatherer**: Collects detailed information on what is needed from a system from various stakeholders, including technical staff, customers, and end-users. This role involves preparing and conducting interviews, surveys, and observation sessions to gather accurate requirements.

3. **Solution Architect**: Designs or redesigns computer systems, applications, and software configurations to solve identified problems, using analysis data to ensure the solution meets all stakeholder requirements. This involves creating design specifications, diagrams, and process maps.

4. **Project Manager**: Oversees the development and implementation of the new system, coordinating with the development team, vendors, and users, ensuring the project remains within scope, time, and budget constraints.

5. **Quality Assurance Tester**: Tests the system to ensure it meets the required standards and functionalities as intended. This involves coordinating testing phases, reporting issues, and verifying fixes.

6. **Trainer and Support Provider**: Trains end-users on the new system and provides ongoing support and enhancements. This role ensures users are comfortable and proficient in using the system and that the system continues to function as needed over time.

7. **Change Agent**: Acts as a catalyst for change within the organization, helping to optimize business processes and integrate new technologies into everyday business workflows.

Systems analysts are crucial for bridging the gap between IT and the business by understanding the needs of both and translating them into technical specifications.

C. Basic Terminologies

System: A collection of interrelated components that work together to achieve a common goal. In the context of SAD, it typically refers to an information system that processes data to produce information.

Systems Analysis: The process of studying and understanding current business processes and systems, identifying problems, and defining requirements for new or improved systems to address those problems.

Systems Design: The process of planning a new business system or replacing an existing system by specifying its components, architecture, modules, interfaces, and data for a system to satisfy specified requirements.

Requirement Gathering: The process of collecting the needs and expectations of stakeholders for a new or altered product. This typically involves interviews, surveys, observations, and reviewing existing documentation.

Use Case: A detailed description of how users will interact with a system to achieve a specific goal. It includes the steps taken by the user and the system's responses.

Entity-Relationship Diagram (ERD): A graphical representation of entities (data objects) and their relationships to each other, typically used in database design.

Data Flow Diagram (DFD): A graphical tool used to represent the flow of data through a system and how the data is processed by the system.

Unified Modeling Language (UML): A standardized modeling language consisting of an integrated set of diagrams, used to specify, visualize, construct, and document the artifacts of a software system. * Use Case Diagram , Class Diagram

Feasibility Study: An assessment of the practicality and potential success of a proposed project, considering factors like technical feasibility, economic feasibility, legal feasibility, operational feasibility, and schedule feasibility.

Prototyping: An iterative approach to system design where a preliminary version of the system is built, tested, and refined based on user feedback until an acceptable prototype is achieved.

Systems Development Life Cycle (SDLC): A framework that describes the phases involved in the development of an information system, which typically includes planning, analysis, design, implementation, and maintenance.

Waterfall Model: A linear and sequential approach to software development where each phase must be completed before the next phase begins, with little to no overlap between phases.

Agile Methodology: An iterative and incremental approach to software development that emphasizes flexibility, collaboration, rapid delivery, and continuous improvement.

Scrum: A subset of Agile methodology; a framework that facilitates team collaboration on complex products. It involves roles such as Scrum Master and Product Owner, and practices like sprints, daily stand-ups, and sprint reviews.

Use Case Diagram: A type of UML diagram that shows the interactions between users (actors) and the system to achieve a goal. It helps identify and clarify requirements.

Class Diagram: A type of UML diagram that shows the static structure of a system, including its classes, attributes, methods, and the relationships among objects.

Stakeholder: Any individual, group, or organization that can affect or be affected by the outcomes of a project. Stakeholders include end users, clients, project managers, developers, and sponsors.

Functional Requirements: Specific behaviors or functions of a system, detailing what the system should do. They define the operations, inputs, outputs, and processes of the system.

Non-Functional Requirements: The criteria that judge the operation of a system, rather than specific behaviors. These include performance, usability, reliability, security, and scalability.

Gap Analysis: The process of comparing the current state of a system to the desired future state and identifying the gaps that need to be addressed to meet the desired goals.

A. Importance of Systems Analysis and Design

Alignment with Business Goals:

- Systems analysis and design ensure that the developed system aligns with the strategic goals of the business. By understanding business objectives, analysts can design systems that support and enhance business processes.

User-Centric Approach:

- Effective systems are designed with the end-user in mind. Systems analysis involves gathering detailed user requirements to ensure the system is intuitive, functional, and meets user expectations.

Mitigating Risks:

- Thorough analysis and design help identify potential risks and challenges early in the development process, allowing for proactive risk management and mitigation strategies.

Efficiency and Productivity:

- Well-designed systems streamline business processes, reduce redundancy, and enhance productivity by providing accurate and timely information to users.

Cost Effectiveness:

- Identifying and addressing requirements accurately during the analysis phase helps avoid costly changes and rework later in the project, ensuring the project stays within budget.

B. Methodological Approach

- What are the key stages in the methodological approach to systems analysis and design?

Requirement Gathering:

- Techniques: Interviews, surveys, observations, and document analysis.
- Outputs: Requirement specifications, user stories, and use cases.

System Analysis:

- Techniques: Data flow diagrams, entity-relationship diagrams, and process modeling.
- Outputs: Logical models of the system, identifying how data flows through the system and how processes interact.

System Design:

- Techniques: Architectural design, interface design, and database design.
- Outputs: Physical models, system architecture diagrams, and detailed design specifications.

Implementation:

- Activities: Coding, system integration, and initial testing.
- Outputs: Developed system components, integration scripts, and initial test results.

Testing:

- Types: Unit testing, integration testing, system testing, and user acceptance testing (UAT).
- Outputs: Test plans, test cases, and test reports.

Deployment:

- Activities: System installation, data migration, and user training.
- Outputs: Deployed system, user manuals, and training materials.

Maintenance:

- Activities: Ongoing support, system updates, and performance monitoring.
- Outputs: Maintenance plans, update patches, and performance reports.

C. Influence of Project Scale and Technology on Methodologies

- How do the scale of the project and the technology involved dictate the methodologies used in systems analysis and design?

Project Scale

- Small-scale Projects:
  - Often benefit from Agile methodologies due to their need for flexibility and rapid iteration.
  - Emphasis on frequent communication, iterative development, and continuous feedback.
- Large-scale Projects:
  - May require more structured approaches like the Waterfall model to manage

complexity and ensure thorough documentation at each stage.

- Emphasis on detailed planning, phased development, and formal approvals at each stage.

Technology Involved

- Modern Technologies
  - Agile methodologies are often preferred to accommodate fast-paced changes and integration with other services.
  - Focus on modular design, continuous integration, and deployment.
- Legacy Systems
  - Waterfall or similar methodologies might be used to ensure careful integration and minimize disruption.
  - Emphasis on thorough analysis and planning to manage dependencies and compatibility issues.

Industry Standards

- Regulated Industries (e.g., Healthcare, Finance)
  - Often requires adherence to strict regulatory standards, making the Waterfall methodology more suitable.
  - Emphasis on compliance, detailed documentation, and rigorous testing.
- Innovative Sectors (e.g., Tech Startups)
  - Agile or Lean methodologies are favored to foster innovation and adaptability.
  - Emphasis on Minimum Viable Product (MVP) development, rapid prototyping, and user feedback.