



UMARU MUSA YAR'ADUA UNIVERSITY, KATSINA

COMPUTING SCIENCE DEPARTMENT

INS 204: SYSTEM ANALYSIS AND DESIGN



LECTURE NOTE 1

Course contents:

LECTURE 1: Introduction to System Analysis and Design

- Overview of Systems and their Components
- Introduction to System Development Life Cycle (SDLC)
- System Analysts

LECTURE 2: Requirements Gathering

- Understanding Stakeholder Needs
- Requirement Elicitation Techniques: Interviews, Questionnaires
- Requirements Analysis and Documentation

LECTURE 3: Modeling Techniques

- Data Flow Diagrams (DFDs)
- Entity-Relationship Diagrams (ERDs)
- Use Case Diagrams
- Activity Diagrams

LECTURE 4: Introduction to Database Design

- Database Concepts and Normalization
- Introduction to SQL (Structured Query Language)

LECTURE 5: System Design Principles

- Interface Design
- Usability Principles

LECTURE 6: System Implementation

- Software Development Methodologies
- Coding Techniques

LECTURE 7: Testing and Quality Assurance

- Unit Testing
- Integration Testing

LECTURE 8: System Deployment

- Deployment Strategies
- User Training and Documentation

LECTURE 9: System Maintenance

- Maintenance Strategies
- Software Updates and Version Control

LECTURE ONE

INTRODUCTION TO SYSTEM ANALYSIS AND DESIGN

Overview of Systems and Their Components

A system is an organized collection of components that work together to achieve a specific goal or set of goals. Systems can be found in various domains, including natural, social, and technological environments. In the context of information systems, a system typically refers to a set of interrelated components that collect, process, store, and distribute information to support decision making, coordination, control, analysis, and visualization within an organization.

Key Components of a System:

1. Inputs:

- **Definition:** Inputs are the resources, data, or materials that are put into a system to be processed.
- **Examples:** Raw data, user commands, energy, materials.

2. Processes:

- **Definition:** Processes are the activities or operations that transform inputs into outputs.
- **Examples:** Data processing, computation, manufacturing steps, transformation of raw materials.

3. Outputs:

- **Definition:** Outputs are the results produced by the system after processing the inputs.
- **Examples:** Processed information, finished products, reports, decisions.

4. Feedback:

- **Definition:** Feedback is information about the output of a system that is used to make adjustments or improvements to the inputs or processes.
- **Examples:** Performance reports, customer feedback, error messages.

5. Control:

- **Definition:** Control involves the mechanisms that monitor and

regulate the operation of a system to ensure it achieves its goals.

- **Examples:** Quality control processes, management oversight, automated control systems.

6. Environment:

- **Definition:** The environment encompasses everything outside the system that can influence its operation and performance.
- **Examples:** External data sources, regulatory constraints, economic conditions.

7. Boundaries:

- **Definition:** Boundaries define the limits of the system and differentiate it from its environment.
- **Examples:** Organizational boundaries, scope of a project, the perimeter of a manufacturing plant.

Types of Systems:

1. Open Systems:

- **Characteristics:** Interact with their environment by receiving inputs and producing outputs.
- **Examples:** Businesses, ecosystems, computer systems connected to the internet.

2. Closed Systems:

- **Characteristics:** Do not interact with their environment; all inputs and outputs are contained within the system.
- **Examples:** A clock, a sealed laboratory experiment.

Information Systems:

Information systems specifically refer to systems designed to manage and process data to provide useful information for decision-making within an organization. They consist of several critical components:

1. Hardware:

- **Definition:** Physical devices and equipment that perform input, processing, storage, and output activities.
- **Examples:** Computers, servers, peripherals.

2. Software:

- **Definition:** Programs and applications that control hardware and process data.
- **Examples:** Operating systems, database management systems, enterprise applications.

3. Data:

- **Definition:** Raw facts and figures that are processed into meaningful information.
- **Examples:** Customer records, sales transactions, inventory levels.

4. People:

- **Definition:** Individuals who use and manage information systems.
- **Examples:** IT professionals, end-users, system analysts.

5. Processes:

- **Definition:** Procedures and rules that define how data is collected, processed, and distributed.
- **Examples:** Business processes, data entry protocols, reporting standards.

6. Networks:

- **Definition:** Communication systems that connect hardware components and allow data sharing.
- **Examples:** Local Area Networks (LAN), Wide Area Networks (WAN), the internet.

Introduction to System Development Life Cycle (SDLC)

The System Development Life Cycle (SDLC) is a structured approach used for developing information systems. It provides a systematic process for planning, creating, testing, and deploying information systems, ensuring that high-quality systems are delivered that meet or exceed customer expectations. The SDLC consists of distinct phases, each with specific tasks and deliverables. Understanding these phases helps in managing the complexity of system development projects and improving project success rates.

Phases of the SDLC

1. Planning:

- **Purpose:** To define the scope, objectives, and feasibility of the project.

- **Activities:** Project initiation, feasibility analysis, project scheduling, resource allocation.
- **Deliverables:** Project charter, feasibility study report, project plan.

2. Analysis:

- **Purpose:** To gather detailed requirements and analyze business needs.
- **Activities:** Requirement gathering (interviews, surveys, document analysis), requirement analysis, requirement documentation.
- **Deliverables:** System requirements specification (SRS), use cases, process diagrams.

3. Design:

- **Purpose:** To create detailed system designs based on requirements gathered.
- **Activities:** System architecture design, database design, interface design, specification of system components.
- **Deliverables:** Design documents, data models, UI/UX prototypes.

4. Implementation (Development):

- **Purpose:** To build and develop the system based on design specifications.
- **Activities:** Coding, integration of system components, development of databases, creation of system interfaces.
- **Deliverables:** Source code, database schema, developed system modules.

5. Testing:

- **Purpose:** To ensure the system works as intended and is free of defects.
- **Activities:** Unit testing, integration testing, system testing, user acceptance testing (UAT).
- **Deliverables:** Test plans, test cases, test scripts, defect reports.

6. Deployment:

- **Purpose:** To deliver the system to the users and make it operational.
- **Activities:** System installation, data migration, user training, deployment planning.
- **Deliverables:** Deployed system, user manuals, training materials.

7. Maintenance:

- **Purpose:** To monitor, support, and enhance the system post-deployment.

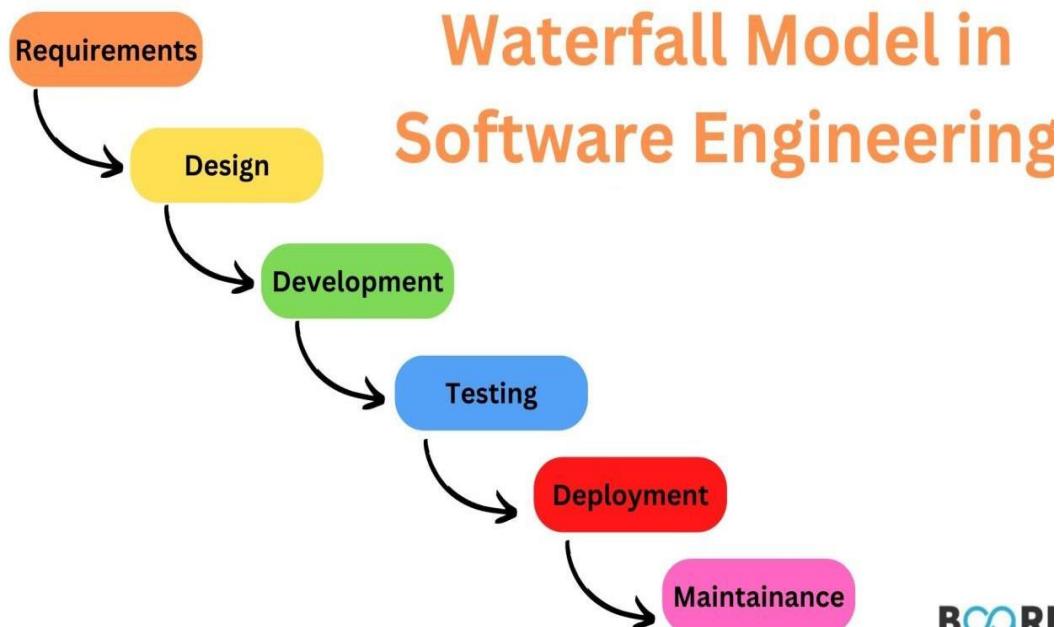
- **Activities:** Bug fixing, system updates, performance tuning, new feature integration.
- **Deliverables:** Maintenance reports, system updates, enhancement specifications.

Types of SDLC Models

Different SDLC models have been developed to address various project requirements and constraints. Each model has its strengths and weaknesses, making them suitable for different types of projects.

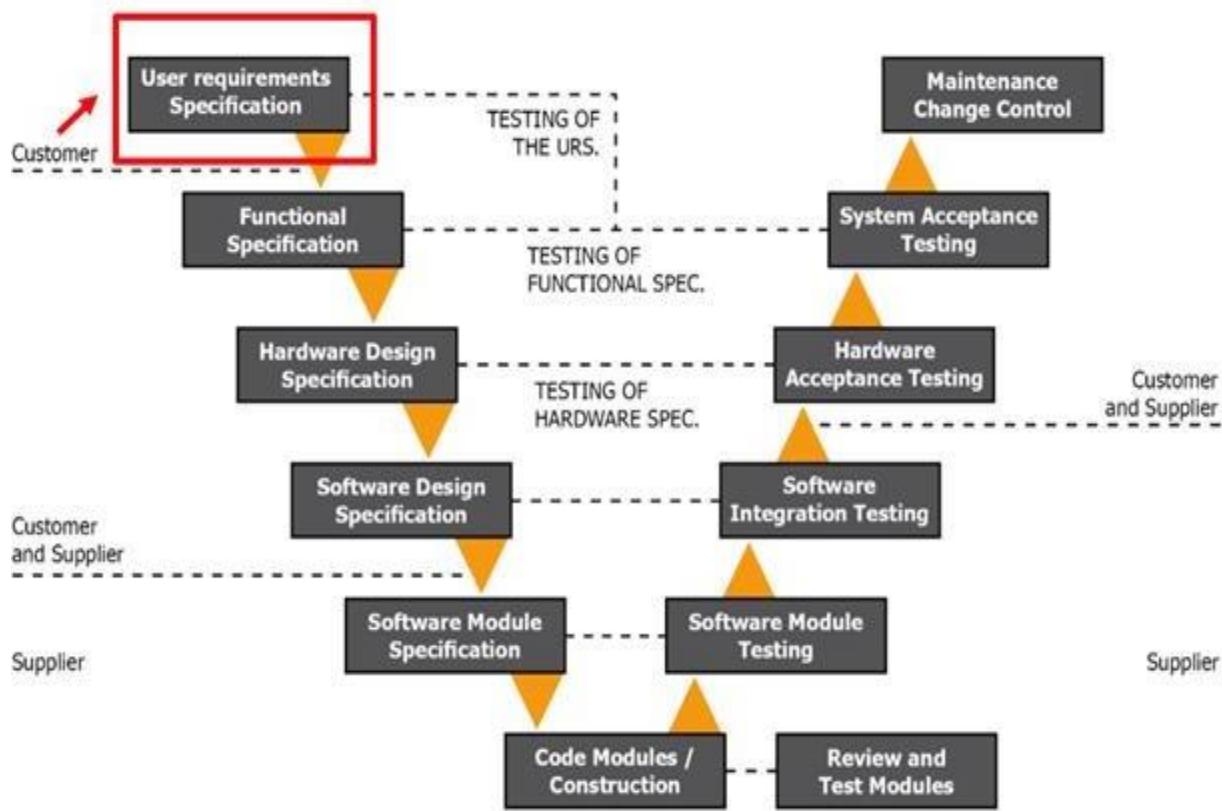
1. Waterfall Model:

- **Description:** A linear and sequential approach where each phase must be completed before the next one begins.
- **Strengths:** Simple, easy to understand, well-suited for projects with clear requirements.
- **Weaknesses:** Inflexible, difficult to accommodate changes, high risk if initial requirements are not well-understood.
- **Use Case:** Suitable for projects with well-defined requirements and low uncertainty.



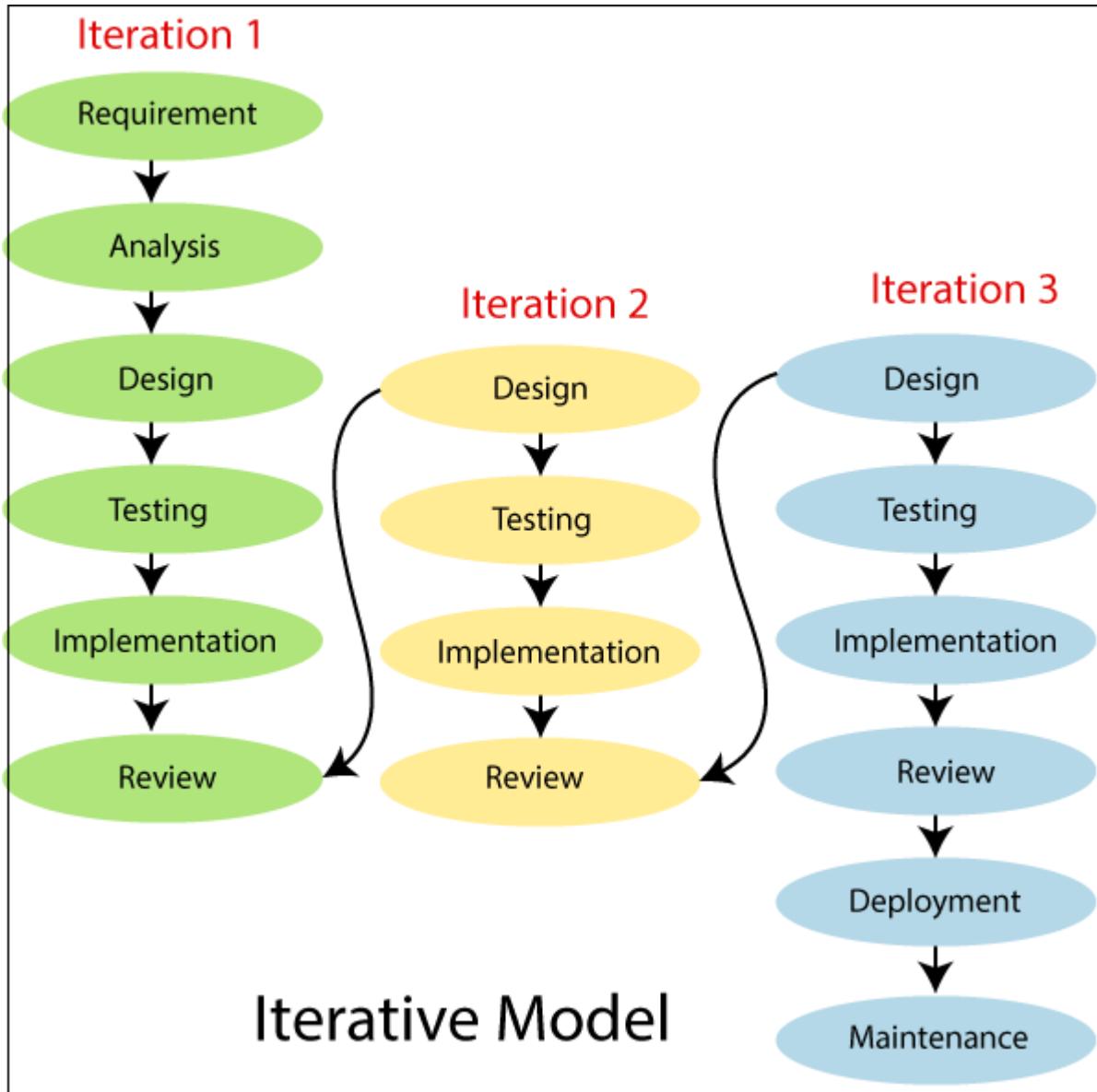
2. V-Model (Validation and Verification Model):

- **Description:** An extension of the Waterfall model that includes corresponding testing phases for each development stage.
- **Strengths:** Emphasizes verification and validation, ensures early detection of defects.
- **Weaknesses:** Similar to Waterfall, it can be rigid and challenging to handle changes.
- **Use Case:** Suitable for projects where quality and reliability are critical.



3. Iterative Model:

- **Description:** Develops the system through repeated cycles (iterations), allowing refinement through each iteration.
- **Strengths:** Flexible, allows for changes and refinements, reduces risk through early iterations.
- **Weaknesses:** Can lead to scope creep, requires careful project management.
- **Use Case:** Suitable for complex projects where requirements may evolve over time.



4. Spiral Model:

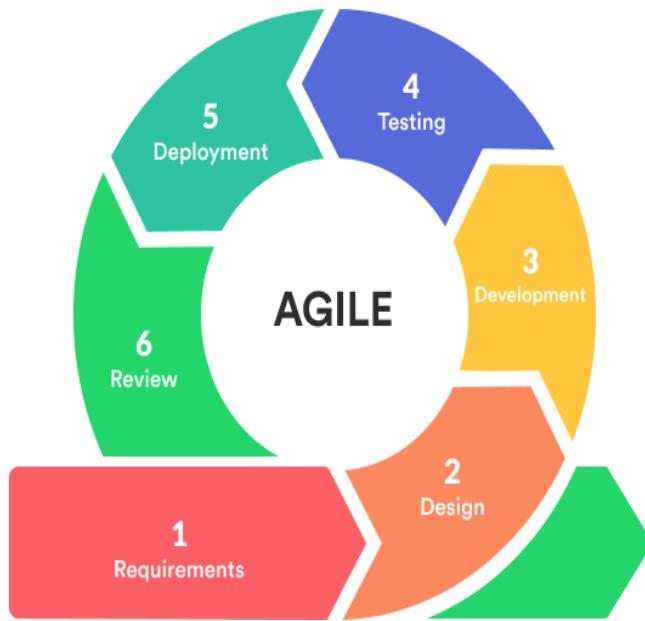
- **Description:** Combines iterative development with risk assessment. Each iteration involves planning, risk analysis, engineering, and evaluation.
- **Strengths:** Focuses on risk management, iterative refinement, and client feedback.
- **Weaknesses:** Can be complex to manage, requires significant risk assessment expertise.
- **Use Case:** Suitable for large, high-risk projects where risk management is a priority.

Spiral Model



5. Agile Model:

- **Description:** Emphasizes flexibility, collaboration, and customer feedback. Uses iterative cycles called sprints to develop system increments.
- **Strengths:** Highly flexible, adaptive to changes, focuses on customer satisfaction.
- **Weaknesses:** Requires strong team collaboration, can be challenging in fixed-budget projects.
- **Use Case:** Suitable for dynamic projects where requirements are expected to change frequently.



6. DevOps Model:

- **Description:** Integrates development (Dev) and operations (Ops) to improve collaboration, automate processes, and enhance the delivery pipeline.
- **Strengths:** Enhances collaboration, continuous delivery, and deployment, quick feedback loops.
- **Weaknesses:** Requires cultural shift, high initial setup and integration costs.
- **Use Case:** Suitable for projects aiming for continuous integration and delivery.

System Analyst

A system analyst is a professional who specializes in analyzing, designing, and implementing information systems. They act as a bridge between business problems and technology solutions, ensuring that business needs are met with appropriate technical solutions. System analysts work to improve system efficiency, integrate new technologies, and enhance business processes through effective use of information systems.

Roles of a System Analyst

1. Requirement Gathering and Analysis:

- **Description:** Collecting and documenting the requirements of the business or project.
- **Activities:** Conducting interviews, surveys, and workshops with stakeholders to understand their needs.

2. System Design:

- **Description:** Creating detailed specifications and design plans for system components.
- **Activities:** Developing data flow diagrams, system models, and technical specifications.

3. Feasibility Analysis:

- **Description:** Assessing the technical, financial, and operational feasibility of proposed systems.
- **Activities:** Conducting cost-benefit analysis, risk assessment, and feasibility studies.

4. System Integration:

- **Description:** Ensuring that new systems integrate seamlessly with existing systems.
- **Activities:** Designing interfaces, data migration plans, and integration testing.

5. Project Management:

- **Description:** Planning, coordinating, and managing system development projects.
- **Activities:** Developing project plans, timelines, resource allocation, and monitoring progress.

6. Quality Assurance and Testing:

- **Description:** Ensuring the system meets the required standards and functions correctly.
- **Activities:** Developing test plans, conducting tests, and managing defect tracking.

7. Training and Support:

- **Description:** Providing training and support to end-users and technical staff.
- **Activities:** Creating user manuals, conducting training sessions, and offering ongoing support.

Skills of a System Analyst

1. Technical Skills:

- **Proficiency in Programming Languages:** Understanding of languages such as Java, C#, Python.
- **Database Management:** Knowledge of SQL, Oracle, and database design principles.
- **System Design Tools:** Familiarity with tools like UML (Unified Modeling Language), ERD (Entity-Relationship Diagrams).

2. Analytical Skills:

- **Problem-Solving:** Ability to analyze complex problems and develop effective solutions.
- **Critical Thinking:** Evaluating various options and making sound decisions based on analysis.

3. Communication Skills:

- **Interpersonal Communication:** Effectively communicating with stakeholders at all levels.
- **Technical Writing:** Documenting requirements, designs, and test plans clearly and concisely.

4. Project Management Skills:

- **Planning and Scheduling:** Developing and managing project plans and schedules.
- **Resource Management:** Allocating and managing resources efficiently.

5. Business Skills:

- **Domain Knowledge:** Understanding the specific industry and

business processes.

- **Cost-Benefit Analysis:** Evaluating the financial impact of system solutions.

6. Interpersonal Skills:

- **Collaboration:** Working effectively with cross-functional teams.
- **Negotiation:** Mediating between stakeholders with different viewpoints.

Types of System Analysts

1. Business System Analysts:

- **Focus:** Bridging the gap between business needs and IT solutions.
- **Responsibilities:** Understanding business processes, identifying opportunities for improvement, and designing systems that enhance business efficiency.

2. Technical System Analysts:

- **Focus:** Providing technical expertise and support for system development.
- **Responsibilities:** Developing technical specifications, coding, and ensuring systems are technically sound.

3. Functional System Analysts:

- **Focus:** Specializing in specific functions within an organization, such as finance or HR.
- **Responsibilities:** Understanding functional requirements and ensuring the system meets these needs.

4. Infrastructure System Analysts:

- **Focus:** Ensuring the underlying IT infrastructure supports business operations.
- **Responsibilities:** Analyzing network, hardware, and software needs, and ensuring system scalability and performance.

5. Data System Analysts:

- **Focus:** Managing and analyzing data to support decision-making.
- **Responsibilities:** Designing data models, ensuring data quality, and implementing data management solutions.

Recommended Text books:

- Tilley, S., & Rosenblatt, H. J. (2016). *Systems analysis and design* (11th Ed.). Cengage Learning.
- Satzinger, J. W., Jackson, R. B., & Burd, S. D. (2016). *Systems analysis and design in a changing world* (7th Ed.). Cengage Learning.
- Hoffer, J. A., George, J. F., & Valacich, J. S. (2016). *Modern systems analysis and design* (8th Ed.). Pearson.
- Dennis, A., Wixom, B. H., & Roth, R. M. (2018). *Systems analysis and design* (7th Ed.). Wiley.