System Analysis and Design BCA 3rd

Topic: System Development Fundamentals

System Development Fundamentals (9 hrs)

This section provides a comprehensive overview of the fundamentals of system development, focusing on the development environment, the origins of software, and managing information systems projects. The concepts covered in this topic are essential for understanding how modern software applications are developed, managed, and improved through various methodologies and frameworks.

---

a. The Systems Development Environment

The **Systems Development Environment (SDE)** refers to the set of tools, processes, methodologies, and technologies used in the development of an information system. This environment enables teams to collaborate, design, implement, test, and maintain software systems. A structured development process ensures the success and efficiency of creating software systems.

1. Introduction to Systems Development Environment

In the context of information systems, the development environment includes hardware, software, methodologies, and standards used to develop, manage, and maintain systems. This environment ensures consistency, quality, and traceability throughout the system development lifecycle.

2. Modern Approach to System Analysis and Design

System analysis and design is the process of studying the needs of an organization and creating software solutions that meet those needs. Modern approaches focus on flexibility, user-centered design, and iterative feedback. The main goal is to produce functional systems efficiently while managing risks and ensuring stakeholder satisfaction.

3. Information System and Its Types

An **Information System (IS)** is a set of interrelated components designed to collect, process, store, and distribute information. There are several types of information systems, including:

- **Transaction Processing Systems (TPS)**: Focus on automating routine transactions.
- **Management Information Systems (MIS)**: Provide management with reports and data to support decision-making.
- **Decision Support Systems (DSS)**: Help with more complex decision-making processes.
- **Expert Systems**: Mimic human expertise to solve specific problems.
- **Enterprise Resource Planning (ERP)**: Integrate core business processes into one system.

4. Developing Information Systems and Its Types

There are multiple types of systems development approaches. These approaches vary based on their flexibility, delivery speed, and the complexity of the system being developed. The development environment will define how these approaches are utilized:

- **Waterfall Development**: A traditional approach that is sequential and rigid.
- **Rapid Application Development (RAD)**: Focuses on quick prototyping and user feedback.
- **Agile Development**: Prioritizes flexibility and iterative cycles of development.
- **Service-Oriented Architecture (SOA)**: Focuses on building systems as a set of services that can be reused across various platforms.

5. The Systems Development Life Cycle (SDLC)

The **Systems Development Life Cycle (SDLC)** is a structured methodology for developing information systems. The SDLC consists of several stages:

- **Planning**: Identifying the project scope, resources, and objectives.
- **Analysis**: Gathering and analyzing business requirements.
- **Design**: Creating detailed system and software designs.
- **Development**: Writing the actual code and developing the system.
- **Testing**: Evaluating the system for defects and ensuring it meets requirements.
- **Implementation**: Deploying the system to users.
- **Maintenance**: Continuously updating and improving the system after deployment.

The **Waterfall SDLC** is a linear and sequential approach, where each phase must be completed before moving to the next one. However, the waterfall model has limitations, especially when changes are required in later phases.

6. Approaches for Improving Development

To overcome limitations in traditional approaches like Waterfall, various techniques and methodologies have been developed to improve system development.

a) CASE Tools (Computer-Aided Software Engineering)

CASE tools are software tools that support the development of systems through various stages, such as planning, design, coding, and testing. These tools help automate repetitive tasks and improve system quality, documentation, and collaboration.

b) Rapid Application Development (RAD)

RAD is an iterative development approach that emphasizes quick development cycles and the use of prototypes. It allows users to give feedback on early versions of the system, helping ensure the system meets their needs before final delivery.

c) Service-Oriented Architecture (SOA)

SOA is a design style in which applications are built as a series of reusable and loosely coupled services. Each service is independent, and different services can communicate with each other over a network. This approach promotes flexibility and scalability.

d) Agile Methodologies

Agile focuses on flexibility and iterative development. Agile methodologies promote collaboration with stakeholders, continuous improvement, and adaptive planning. Some well-known agile methodologies include **Scrum** and **Kanban**.

e) Extreme Programming (XP)

XP is an agile development methodology that emphasizes technical excellence, rapid feedback, and close collaboration between developers and customers. Key practices include pair programming, test-driven development, and continuous integration.

f) Object-Oriented Analysis and Design (OOAD)

OOAD is a methodology for designing systems using object-oriented principles. It focuses on creating reusable and modular software components based on real-world objects and their interactions. Key concepts in OOAD include classes, objects, inheritance, polymorphism, and encapsulation.

---

b. The Origins of Software

The origin of software development is deeply rooted in the need for efficient processing and automation of tasks. Over time, various practices, technologies, and models have evolved to handle software development more effectively.

1. Introduction

The process of developing software can be traced back to the early days of computing, where computers were used to perform specific tasks. Early software development was manual and highly complex, with a focus on solving specific problems rather than general-purpose applications.

2. System Acquisition

System acquisition refers to the process of acquiring software and systems for an organization. This can involve purchasing off-the-shelf software, developing custom software, or integrating existing systems. Acquiring the right system involves assessing the needs, evaluating vendors, and ensuring the selected system is scalable and cost-effective.

3. Reuse

Software reuse involves leveraging existing code or system components in new applications to reduce development time and cost. Reuse can take several forms:

- **Code Reuse**: Reusing previously written code components or libraries.
- **Component Reuse**: Reusing pre-built software components, such as APIs or services.

- **Design Reuse**: Leveraging established design patterns or architectural solutions.

Reusing software accelerates development, improves consistency, and reduces the risk of errors.

---

c. Managing the Information Systems Project

Managing information systems projects is a key aspect of software development. Effective management ensures that the project stays on track, within scope, and on budget.

1. Introduction to Project Management

Project management involves planning, organizing, and overseeing the resources and activities involved in the development of an information system. It helps ensure that the project meets its objectives, delivers value to stakeholders, and adheres to timelines and budgets.

2. Managing Information Systems Projects

Successful management of information systems projects requires careful planning, monitoring, and control. Project managers must coordinate teams, resources, and tasks, while also managing risks, issues, and stakeholder expectations.

Key elements of managing IS projects include:

- **Project Scope**: Defining the boundaries of the project, including what is and isn't included.
- **Resources**: Allocating human resources, budget, and tools effectively.
- **Risk Management**: Identifying and mitigating potential risks to project success.
- **Communication**: Maintaining open communication with stakeholders to keep them informed of progress and issues.

3. Representing and Scheduling Project Plans

Project plans should include detailed timelines, task lists, and resource allocations. Scheduling involves breaking the project down into smaller, manageable tasks and determining the timeframes for each.

Common techniques for project scheduling:

- **Gantt Charts**: Visual representation of tasks and their timelines.
- **Critical Path Method (CPM)**: Identifies the longest path of tasks that must be completed on time to ensure the project is finished as scheduled.
- **PERT (Program Evaluation Review Technique)**: Helps estimate the time required to complete tasks when there is uncertainty in task duration.

4. Using Project Plans

Project plans are used to ensure that the project stays on track. These plans should be regularly updated and serve as a guide for the team. The project manager needs to monitor progress, adjust schedules, and keep stakeholders informed.

5. Using Project Management Software

Project management software is used to streamline the planning, execution, and monitoring of information systems projects. These tools help with task assignments, communication, tracking, and reporting.

Popular project management tools include:

- **Microsoft Project**: Provides scheduling, resource allocation, and reporting features.
- **Trello**: A simple, flexible tool for organizing tasks and collaboration.
- **JIRA**: A popular tool for managing agile development projects, tracking issues, and assigning tasks.

Conclusion

The **System Development Fundamentals** covered in this note highlight the core elements involved in building and managing information systems. Understanding different methodologies (such as Agile, Waterfall, RAD), the importance of managing resources effectively, and the use of modern tools can help organizations successfully develop and maintain software systems. Properly managing software development projects, from acquisition to execution, ensures that the systems created meet business needs, are delivered on time, and are aligned with the organization's strategic goals.

Topic: Planning

Planning (7 hrs)

Effective planning is crucial to the success of system development projects. It involves the identification, selection, initiation, and careful planning of projects, ensuring that resources, time, and efforts are aligned with organizational goals. This section will delve into the identification and selection of system development projects and the process of initiating and planning these projects.

---

a. System Development Projects: Identification and Selection

1. Introduction

System development projects are undertaken by organizations to meet specific business goals, improve operational efficiency, or develop new systems that provide competitive advantages. The first critical step in managing these projects is identifying the right projects to undertake and selecting those that will most benefit the organization.

2. Identifying and Selecting Systems Development Projects

The identification and selection of system development projects involve recognizing potential projects that align with the organization's strategic objectives. This requires thorough analysis, input from stakeholders, and an understanding of the broader business goals.

a) Identifying System Development Projects

- **Problem Identification**: Projects are often born from a business problem or an opportunity. For example, an inefficient inventory system may require a new inventory management system.
- **Needs Assessment**: A comprehensive needs analysis is conducted to understand the problem, define user requirements, and identify gaps in current systems.
- **Technology Advancements**: Emerging technologies can also drive the need for system development projects. Adopting newer technology or upgrading outdated systems could be critical for staying competitive.
- **Regulatory Requirements**: Changes in laws or industry standards may require systems to be updated to maintain compliance.

b) Selecting System Development Projects

Once projects are identified, organizations must prioritize which projects to undertake. This is done by evaluating the project's:

- **Strategic Alignment**: Does the project align with the organization's long-term goals and vision?
- **ROI (Return on Investment)**: What is the potential return for the resources invested?
- **Risk Assessment**: What are the risks involved? Can the project be realistically completed within time and budget constraints?
- **Resource Availability**: Does the organization have the necessary resources (people, technology, finances) to support the project?

Tools such as **SWOT Analysis** (Strengths, Weaknesses, Opportunities, Threats) and **Cost-Benefit Analysis** are commonly used to assess the value and feasibility of the projects.

c) Corporate and Information Systems Planning

- **Corporate Planning**: This involves the long-term planning of the organization's strategy. Corporate goals and objectives should drive the identification and selection of system development projects.
- **Information Systems Planning**: Information Systems (IS) planning focuses on aligning IT projects with business strategies. The goal is to ensure that IT investments support overall business objectives and are managed effectively.

Example of Project Selection:

Imagine a company that has been receiving complaints about its customer support system. Upon identifying the need for improvement, the company conducts a feasibility study. They assess the cost of updating the existing system versus building a new one, and after reviewing the financial and operational impact, they decide to proceed with building a new system.

---

b. System Development Projects: Initiation and Planning

1. Introduction

Once a system development project has been selected, it moves into the initiation and planning phase. This stage involves the formalization of the project by establishing objectives, defining scope, and ensuring that the necessary resources are in place. Proper initiation and planning help set realistic expectations and provide a clear path forward for the project team.

2. Initiating and Planning System Development Projects

System development project initiation and planning involve setting up the project's foundational structures. During this phase, a project manager is appointed, and the project's objectives, scope, timeline, resources, and budget are defined.

Key elements of the initiation and planning phase include:

- **Project Charter**: A formal document that authorizes the project, provides its goals, and identifies stakeholders.
- **Project Team**: Identification of the core project team, which typically includes project managers, analysts, developers, and testers.
- **Stakeholder Analysis**: Identifying and understanding the interests and expectations of the stakeholders, including internal teams, customers, and external parties.

a) Process of Initiating and Planning IS Development Projects

The initiation and planning process consists of several steps:

1. **Establishing the Project Goals**: Clear and measurable goals are set for the project to guide its execution and evaluate its success. For example, if the goal is to develop a new customer relationship management (CRM) system, the objectives might include improving customer satisfaction by 25% in the first year.
2. **Defining the Project Scope**: The project scope defines the boundaries of the project, specifying what is included and excluded. For example, the scope may specify that only certain customer interaction channels (e.g., email, website) will be covered, while others (e.g., phone support) will not.
3. **Identifying Stakeholders**: Stakeholders are individuals or groups who have a vested interest in the project's success. This includes both internal stakeholders, like project team members and department heads, and external stakeholders, such as clients and third-party vendors.
4. **Developing a Project Plan**: The project plan outlines the timeline, resources, budget, and responsibilities. It provides a roadmap for the project and ensures that all activities are well-

organized. This includes detailed scheduling of tasks and milestones using tools like **Gantt charts** or **Work Breakdown Structure (WBS)**.

5. **Risk Management Plan**: Risk management is an essential component of project planning. A risk management plan is developed to identify potential risks, assess their impact, and develop mitigation strategies. Common risks include technological challenges, budget overruns, and delays.

---

3. Assessing Project Feasibility

Before proceeding further, it is essential to assess the feasibility of the project. Feasibility studies help evaluate the practicality and potential success of a project. This can be done through various types of feasibility analysis:

- **Technical Feasibility**: Can the project be built with the existing technology? Do we have the necessary infrastructure?
- **Operational Feasibility**: Will the system be easy to operate and maintain? Does the organization have the skills to manage it?
- **Economic Feasibility**: Does the project make financial sense? Will it provide a return on investment (ROI)?
- **Legal Feasibility**: Does the project comply with laws, regulations, and industry standards?
- **Schedule Feasibility**: Can the project be completed within the allocated time?

Feasibility studies reduce the chances of project failure by providing early insight into any potential obstacles.

---

4. Building and Reviewing the Baseline Project Plan

The **Baseline Project Plan** is a formalized version of the initial project plan that outlines the expected outcomes, resources, timelines, and cost estimates. The baseline is critical because it serves as the standard against which project progress is measured.

a) Building the Baseline Plan:

The baseline plan is created after initial discussions and planning. It includes:

- **Detailed Project Schedule**: A timeline that includes task dependencies and milestones.
- **Cost Estimates**: A comprehensive budget that accounts for resources, equipment, software, and personnel.
- **Resource Allocation**: A clear indication of the project team members, their roles, and responsibilities.

b) Reviewing the Baseline Plan:

Once the baseline plan is created, it is reviewed by the project team and key stakeholders to ensure that it is feasible and aligns with the organization's goals. Adjustments may be made to address any concerns before final approval. During the project, progress is continually compared to the baseline to track deviations and take corrective actions if needed.

Conclusion

Planning is the foundation of successful system development projects. Identifying and selecting the right projects ensures alignment with business objectives, while initiating and carefully planning those projects sets the stage for efficient and effective development. By conducting thorough feasibility studies and establishing a baseline project plan, organizations can mitigate risks and ensure that projects stay on track. Proper planning not only defines the direction of the project but also helps manage time, resources, and expectations, leading to successful outcomes.

Topic: Analysis

Analysis (13 hrs)

The **Analysis** phase is a crucial part of the Systems Development Life Cycle (SDLC), where the requirements for the system are gathered, understood, and analyzed. This phase includes defining both functional and non-functional requirements, designing models for processes, and ensuring that the system's data needs are well understood. It establishes the foundation for later stages, including system design and development. This section covers system requirements, system process requirements, and system data requirements.

---

a. System Requirements

1. Introduction

System requirements define what a system must accomplish, both from a business and technical perspective. The analysis phase focuses on understanding the needs of stakeholders and translating them into detailed, clear, and unambiguous system requirements. These requirements guide the design, implementation, and evaluation of the system.

2. Performing Requirements Determination

Requirements determination is the process of identifying and defining the needs of the users and stakeholders. This is typically the first step in the analysis phase, where the main objective is to gather, analyze, and prioritize requirements for the system.

The process involves:

- **Stakeholder Interviews**: Talking to individuals or groups who will use or be affected by the system.

- **Surveys and Questionnaires**: Collecting broad input from a large number of users.
- **Document Reviews**: Analyzing existing documentation, such as system reports, user manuals, and policy documents, to understand the existing systems.
- **Observation**: Observing current system usage to gather insights about user behavior and system inefficiencies.

3. Traditional Methods for Determining System Requirements

Traditional methods involve structured, step-by-step approaches to gathering and analyzing requirements. Some common traditional methods include:

- **Interviews and Focus Groups**: Direct discussions with users, stakeholders, and experts to gather detailed insights.
- **Use Cases and Scenarios**: Describing how users will interact with the system and detailing the system's functionality in various situations.
- **Joint Application Development (JAD)**: A facilitated workshop where users and developers collaborate to determine system requirements.

These traditional methods are often formal and documented in detailed reports.

4. Contemporary Methods for Determining System Requirements

Contemporary methods emphasize collaboration, rapid feedback, and iterative development. Some of the modern approaches include:

- **Prototyping**: Building a prototype early in the development process and refining it based on user feedback. This approach is particularly useful for understanding complex requirements that are difficult to define upfront.
- **User Stories and Use Case Modeling**: Agile development often uses user stories—short, simple descriptions of features from the user's perspective.
- **Storyboarding**: A visual tool for depicting the flow of user interactions with the system, often used in UI/UX design.

These contemporary approaches enable flexibility and adaptation as the system's requirements become clearer throughout the development process.

5. Requirements Management Tools

These are software tools that help manage, track, and organize the requirements throughout the project lifecycle. Tools like **JIRA**, **IBM Rational DOORS**, and **Microsoft TFS** allow teams to capture, prioritize, and monitor requirements to ensure that they are met during development.

6. Requirements Determination Using Agile Methodologies

In Agile methodologies, requirements are not fully defined upfront. Instead, they evolve iteratively through constant interaction between developers and stakeholders. Common tools and techniques in Agile include:

- **User Stories**: Brief descriptions of functionality as seen from the perspective of the end-user.
- **Backlog**: A prioritized list of user stories or features that need to be developed.
- **Sprints**: Short development cycles, typically lasting 1-2 weeks, where a subset of the backlog is implemented, tested, and reviewed.

---

b. System Process Requirements

1. Introduction

Once the system requirements have been gathered, the next step is to understand the processes that the system will support. This includes how the system will perform its tasks and handle various inputs and outputs. This is done through process modeling.

2. Process Modeling

Process modeling refers to the graphical representation of processes within the system. It helps analyze workflows, identify inefficiencies, and understand how different parts of the system interact.

Key techniques for process modeling include:

- **Data Flow Diagrams (DFDs)**: A graphical way to represent how data flows through a system and how it is transformed.
- **Flowcharts**: Diagrams that represent step-by-step logic for a process.
- **Business Process Modeling Notation (BPMN)**: A method for modeling business processes with standard symbols to depict process workflows.

3. Data Flow Diagramming Mechanics

Data Flow Diagrams (DFDs) are one of the most widely used methods for modeling system processes. A DFD visually represents the flow of data in and out of various processes, systems, or storage locations.

Components of a DFD:

- **External Entities**: Represent the sources or destinations of data (e.g., users, other systems).
- **Processes**: Represent the activities or operations that transform the data (e.g., calculations, updates).
- **Data Stores**: Represent data repositories where information is stored.
- **Data Flows**: Arrows that show the direction of data movement between entities, processes, and stores.

DFDs can be created at different levels of abstraction, from high-level context diagrams to detailed data flow diagrams.

4. Using Data Flow Diagramming in the Analysis Process

Data Flow Diagrams are useful in the analysis phase because they help stakeholders understand how the current system works and identify areas for improvement. For example, a DFD can show how customer orders flow through a system, from being placed by the user to being processed by the back-end system and ultimately fulfilling the order.

DFDs are typically used in:

- **Current System Analysis**: Understanding the existing system to identify gaps or inefficiencies.
- **Designing New Systems**: Modeling the processes of the new system.

5. Modeling Logic with Decision Tables

Decision tables are used to model complex decision-making logic in systems. They consist of a matrix where conditions (rules) are listed in columns, and possible actions are shown in rows. Each combination of conditions determines the actions to be taken.

Decision tables are useful when there are multiple rules and outcomes that need to be considered, such as pricing logic, eligibility criteria, or error handling.

---

c. System Data Requirements

1. Introduction

Understanding the data requirements of a system is as crucial as understanding its processes. The data model defines how the data will be stored, accessed, and related to other pieces of data. In this section, we focus on conceptual data modeling, which involves designing the data structures needed for the system.

2. Conceptual Data Modeling

Conceptual Data Modeling is the process of abstractly defining the data requirements of a system without focusing on how the data will be physically stored or implemented. The goal is to understand what data is needed, how it is related, and how it will be used.

- **Entity Relationship (ER) Model**: One of the most widely used techniques for conceptual data modeling.
- **ER Diagram**: A visual representation that shows entities (e.g., customers, orders) and their relationships (e.g., a customer places an order).

3. Gathering Information for Conceptual Data Modeling

Information for conceptual data modeling is gathered through:

- **Interviews**: Engaging with stakeholders to understand their data needs.

- **Document Analysis**: Reviewing existing system documents to extract relevant data requirements.
- **Observation**: Observing the processes to understand what data is generated and consumed.

4. Introduction to E-R Modeling

Entity-Relationship (ER) Modeling is a popular technique for designing databases. It focuses on defining:

- **Entities**: Objects or concepts that hold data (e.g., a product, an employee).
- **Relationships**: How entities are related (e.g., an employee works in a department).
- **Attributes**: Characteristics or properties of entities (e.g., a product has a name and price).

ER diagrams use standard symbols, such as rectangles for entities, diamonds for relationships, and ovals for attributes.

5. Conceptual Data Modeling and the E-R Model

The ER model helps define the structure of the system's database at a conceptual level, focusing on the relationships and data flows rather than the technical implementation. It is especially useful for database designers and developers in the early stages of database development.

6. Representing Super-types and Sub-types

In data modeling, a **super-type** is a general entity that can have specialized sub-types. For example:

- A **Person** super-type might have sub-types like **Employee** and **Customer**.
- This concept helps model complex systems where entities share common attributes but also have distinct features.

7. Business Rules

Business rules define the constraints and conditions under which the system operates. They ensure the accuracy, consistency, and integrity of the data. Examples include:

- A **customer** must have a unique ID.
- An **order** cannot exceed the available stock quantity.

8. Role of Packaged Conceptual Data Models-Database Patterns

Packaged data models, often referred to as **database patterns**, are pre-designed templates or frameworks that can be customized for specific use cases. These models provide a starting point for data designers and help streamline the modeling process by providing common data structures and relationships used in a variety of applications.

Conclusion

The **Analysis** phase plays a crucial role in the overall system development process, as it defines what the system should do, how it will interact with data, and how users will engage with it. This phase involves understanding the system's requirements through both traditional and contemporary methods, modeling the system's processes, and defining its data needs. It sets the stage for the system's design and ensures that the developed system will meet the business objectives and user needs effectively.

Topic: Design

Design (12 hrs)

The **Design** phase in system development is where all the gathered requirements, process models, and data models are transformed into a blueprint that will guide the system's construction. This phase involves designing the architecture, databases, user interfaces, and other critical components of the system. The design is crucial as it impacts the system's performance, usability, and maintainability.

---

a. Designing Databases

1. Introduction

A well-designed database is essential for efficient data management and system performance. Database design focuses on structuring data in a way that ensures consistency, integrity, and scalability. In this part of the design phase, the system's data storage needs are defined, taking into account how data will be used and manipulated within the system.

2. Database Design

Database design involves creating a detailed blueprint for how data will be stored, organized, and accessed. The goal is to ensure that data is logically structured and optimized for performance. The process includes determining:

- The data entities and their relationships.
- The attributes of each entity.
- The keys (primary and foreign) that uniquely identify records.

3. Relational Database Model

The **relational database model** is the most widely used database model. In this model, data is stored in tables (relations) with rows (records) and columns (attributes). The relational model allows for easy querying, updating, and manipulation of data using Structured Query Language (SQL).

- **Tables (Relations)**: A table represents a single entity (e.g., customer, order).
- **Rows (Records)**: Each row in a table represents a unique instance of the entity (e.g., a specific customer).

- **Columns (Attributes)**: Each column represents an attribute of the entity (e.g., customer name, order date).

4. Normalization

Normalization is the process of organizing data to reduce redundancy and improve data integrity. The goal is to divide large tables into smaller, related tables and ensure that the relationships between them are logically consistent. Normalization involves the following normal forms:

- **First Normal Form (1NF)**: Ensures that each column contains atomic values and that there are no repeating groups of columns.
- **Second Normal Form (2NF)**: Ensures that the database is in 1NF and that all non-key attributes are fully functionally dependent on the primary key.
- **Third Normal Form (3NF)**: Ensures that the database is in 2NF and that there are no transitive dependencies between non-key attributes.
- **Boyce-Codd Normal Form (BCNF)**: A stricter version of 3NF, ensuring that all functional dependencies are handled properly.

5. Transforming E-R Diagrams into Relations

Once the **Entity-Relationship (E-R) diagram** is created in the analysis phase, the next step is to convert the E-R diagram into a relational database schema. This transformation involves:

- **Entities** becoming **tables**.
- **Attributes** of entities becoming **columns** of the respective tables.
- **Relationships** between entities becoming **foreign keys** in the tables that reference the related tables.

6. Merging Relations

After the normalization process, sometimes it becomes necessary to merge related tables that were split during the normalization process. This is done to improve the performance of the database or reduce the complexity of queries.

7. Physical File and Database Design

In the **physical design** phase, decisions are made regarding how the data will be stored on physical media, such as disk drives. Key considerations include:

- **Indexing**: Creating indexes to speed up data retrieval.
- **Partitioning**: Dividing large tables into smaller parts to optimize performance.
- **File organization**: Determining the file structure that will store the data.
- **Storage efficiency**: Ensuring the database is optimized for storage space.

8. Designing Fields

In this step, each database field is defined with its data type, size, and constraints. Common data types include:

- **Integer**, **Float**, **Decimal** for numbers.
- **Char**, **Varchar**, **Text** for strings.
- **Date**, **Datetime** for date and time.
- **Boolean** for true/false values.

9. Designing Physical Tables

The physical design of tables includes defining how data is stored in the database. This includes:

- Choosing appropriate **primary keys** and **foreign keys**.
- **Partitioning** large tables into smaller, manageable chunks.
- **Indexing** frequently queried fields to optimize performance.
- Considering **redundancy** in tables to ensure efficient data retrieval.

---

b. Designing Forms and Reports

1. Introduction

Forms and reports are essential user interface components that allow users to interact with the system's data. Forms are used for entering and editing data, while reports display information in a readable format. Designing them effectively ensures that the system is user-friendly and meets the needs of the users.

2. Designing Forms and Reports

Form design refers to creating screens that users interact with to input and modify data. **Report design** involves creating output formats that present data in a meaningful way.

Good design involves:

- **User-centered design**: Designing with the user in mind to ensure the interface is intuitive and easy to use.
- **Consistency**: Using consistent layouts, labels, and color schemes to improve usability.
- **Feedback**: Providing users with clear feedback (e.g., confirmation messages, error notifications).

Forms:

- Fields should be appropriately labeled, with clear instructions for users.
- Fields should be arranged logically to avoid confusion and improve efficiency.

- Consider using **validation rules** to ensure correct data is entered (e.g., requiring a valid email format).

Reports:

- Reports should display data in an easily readable format, typically in tables, charts, or graphs.
- Grouping related data together helps improve readability.
- Sorting and filtering options can improve the usefulness of the report.

3. Formatting Forms and Reports

Formatting involves adjusting the visual presentation of forms and reports, such as:

- **Fonts**: Choosing readable fonts for users.
- **Color schemes**: Using contrasting colors to highlight important sections or actions.
- **Spacing and alignment**: Ensuring that form fields and report data are spaced out properly for readability.

4. Assessing Usability

Usability refers to how easy and efficient it is for users to interact with the forms and reports. Testing the design with real users and collecting feedback is essential for improving usability. Common assessments include:

- **User testing**: Observing how users interact with the forms and reports.
- **Task analysis**: Understanding the tasks users perform to ensure the design meets their needs.
- **Heuristic evaluation**: Reviewing the design based on established usability principles (e.g., Nielsen's heuristics).

---

c. Designing Interfaces and Dialogues

1. Introduction

User interfaces (UIs) and dialogues are critical components of the system's design. They define how users interact with the system, navigate through it, and complete tasks. Designing effective interfaces is essential to ensure a positive user experience.

2. Designing Interfaces and Dialogues

Interface design involves creating the layout and elements that users will interact with. This includes:

- **Menus**: Organizing options in a clear and easy-to-use structure.
- **Buttons**: Placing action buttons in intuitive positions to guide users through tasks.
- **Navigation**: Ensuring that users can easily move between sections of the application.

- **Input Fields**: Creating forms and input fields that are easy to fill out, with clear labels and validation.

Dialogues are interactions that occur between the system and the user, often in the form of pop-up windows or message boxes. Good dialogue design ensures users know what action to take next and provides them with helpful feedback.

3. Interaction Methods and Devices

Designing for different **interaction methods** and **devices** is critical for creating interfaces that work across multiple platforms. This includes:

- **Mouse, Keyboard, and Touchscreen**: Designing interfaces that support various input methods.
- **Voice Interaction**: Designing voice-controlled systems where users can interact with the system using speech commands.
- **Responsive Design**: Ensuring that interfaces adjust to different screen sizes and orientations (desktop, tablet, mobile).

4. Designing Interfaces and Dialogues in Graphical Environments

In modern systems, **graphical user interfaces (GUIs)** are commonly used. Designing these interfaces involves:

- Using **icons** to represent actions and objects clearly.
- Ensuring that graphical elements like buttons, sliders, and text fields are placed consistently and logically.
- Ensuring that the interface is visually appealing, easy to navigate, and accessible to users with disabilities (e.g., using color contrast, text size adjustments, and screen reader compatibility).

Conclusion

The **Design** phase of the SDLC is critical for creating the blueprint of the system that will guide its development. It covers database design, form and report design, and user interface design. Effective database design ensures data integrity and optimal storage, while well-designed forms, reports, and interfaces ensure that users can interact with the system efficiently and effectively. By focusing on user needs and system requirements, designers can ensure that the final product will meet business objectives and user expectations.

Topic: Implementation and Maintenance

Implementation and Maintenance (4 hrs)

The **Implementation and Maintenance** phase of the system development life cycle (SDLC) is where the system is transitioned from design to operation. During this phase, the system is built, tested, installed, and then supported and maintained throughout its life cycle. The success of this phase relies heavily on effective planning, thorough testing, and continuous support for the system and its users.

---

a. System Implementation

1. Introduction

System Implementation is the process of building, testing, deploying, and transitioning the system from development to production. This phase involves turning the design into a fully functional system that is accessible and usable by the end-users. System implementation requires careful planning to ensure minimal disruption and successful deployment.

2. System Implementation

In the **system implementation** phase, the actual work of building and deploying the system occurs. It typically involves:

- **Coding**: Writing the program code based on the designs.
- **Integration**: Ensuring that various components of the system work together seamlessly.
- **Configuration**: Setting up the system environment (servers, networks, etc.) and ensuring all configurations are correct.
- **Deployment**: Installing the system on the target infrastructure or making it available to users.

3. Software Application Testing

Testing is a critical part of the system implementation process. It ensures that the system meets the specifications outlined during the analysis and design phases. Common types of testing include:

- **Unit Testing**: Testing individual components or modules for correctness.
- **Integration Testing**: Testing the interaction between different system components to ensure they work together properly.
- **System Testing**: Testing the entire system as a whole to ensure it meets all requirements.
- **User Acceptance Testing (UAT)**: Testing conducted by end-users to ensure the system works as expected in real-world scenarios.

Testing Process:

- **Develop test plans** that outline the testing strategy and specific scenarios to be tested.
- **Run tests** and document the results.
- **Fix defects** or issues that arise during testing and re-test as necessary.
- **Ensure system reliability** and performance standards are met.

4. Installation

System installation refers to the process of moving the completed and tested system from the development environment to the production environment. Installation can occur in various ways:

- **Direct Installation**: Installing the new system directly, replacing the old one.
- **Parallel Installation**: Running the new and old systems simultaneously for a while to ensure the new system is functioning correctly before the old one is retired.
- **Phased Installation**: Implementing the system in stages, with certain features or modules introduced gradually.
- **Pilot Installation**: Installing the system on a limited scale (e.g., with a small user group) before a full rollout.

The installation process should include:

- Ensuring hardware and software requirements are met.
- Configuring the system to fit the organization’s environment.
- Installing the application and ensuring all components are working.

5. Documenting the System

Documentation is an essential part of system implementation, providing a record of the system’s design, installation, and configuration. It includes:

- **Technical Documentation**: Detailed information about the system architecture, code, and configuration settings.
- **User Documentation**: Manuals and guides for end-users to help them understand how to interact with the system.
- **Maintenance Documentation**: Information on how to perform updates and troubleshoot common issues.

Effective documentation serves as a reference for both users and IT support staff.

6. Training and Supporting Users

Once the system is installed, users need to be trained to effectively use it. **User training** ensures that staff understand how to navigate the system and perform necessary tasks. Training can include:

- **Classroom Training**: In-person or online sessions that teach the use of the system.
- **Hands-on Training**: Providing users with the opportunity to interact with the system and practice using it.
- **User Guides and Tutorials**: Providing written materials or video tutorials for users to reference.

Additionally, users need ongoing support to resolve issues, answer questions, and ensure proper system usage. This support may involve:

- **Help Desk**: A team available to answer user queries and resolve issues.

- **Technical Support**: Addressing more complex issues related to the system's performance or functionality.

7. Organizational Issues in Systems Implementation

Implementing a new system within an organization often involves addressing several organizational challenges, such as:

- **Resistance to Change**: Employees may resist the new system due to fear of change or unfamiliarity with the new processes.
- **Training Needs**: Ensuring that employees are trained and comfortable with the new system.
- **Communication**: Effectively communicating with stakeholders about the status and impact of the system implementation.
- **Data Migration**: Ensuring that data from previous systems is transferred accurately to the new system.

---

b. System Maintenance

1. Introduction

System Maintenance refers to the activities involved in keeping a system operational, efficient, and up-to-date after it has been implemented and deployed. It is an ongoing phase of the SDLC that begins once the system is live and continues until the system is retired. Maintenance ensures that the system remains functional and can adapt to changing requirements over time.

2. Maintaining Information Systems

System maintenance includes several activities aimed at ensuring that the system operates as intended and remains relevant over time:

- **Bug Fixes**: Addressing any issues or defects that arise after deployment.
- **System Updates**: Applying updates to software components, security patches, and other relevant system upgrades.
- **Performance Optimization**: Monitoring and improving system performance, ensuring that it continues to meet the expected load and response times.
- **Enhancements**: Adding new features or capabilities based on user feedback or evolving business requirements.
- **Security Maintenance**: Regularly reviewing and updating security measures to protect against vulnerabilities and data breaches.

There are typically three types of system maintenance:

1. **Corrective Maintenance**: Fixing issues that cause the system to malfunction or not meet specifications.

2. **Adaptive Maintenance**: Making changes to the system to keep it compatible with evolving technologies or business processes.
3. **Perfective Maintenance**: Enhancing the system to improve performance, functionality, or usability.

3. Conducting Systems Maintenance

Maintenance activities can be broadly categorized into:

- **Routine Maintenance**: Regular tasks such as monitoring system performance, applying security patches, and updating documentation.
- **Emergency Maintenance**: Quick fixes for critical system failures, usually caused by unforeseen issues or bugs.
- **Scheduled Maintenance**: Planned upgrades or changes to the system that are typically announced in advance to users.

To effectively manage system maintenance:

- **Develop a Maintenance Plan**: Define a structured approach for performing maintenance activities.
- **Set Up a Maintenance Team**: Ensure there is a dedicated team responsible for ongoing maintenance tasks.
- **Monitor System Performance**: Use monitoring tools to detect issues early and address them before they impact users.

Conclusion

The **Implementation and Maintenance** phase is essential for ensuring that the system works as expected and remains functional throughout its life cycle. Successful implementation involves proper planning, testing, training, and user support. Once the system is live, maintenance ensures it continues to meet the organization's needs, remains secure, and adapts to any changing requirements or issues. By focusing on both immediate deployment and long-term support, organizations can ensure their systems deliver consistent value.

Topic: Syllabus

Course Description

Course Description

This course mainly focuses on different aspect of system analysis and design such as foundation, planning, analysis, design, implementation and maintenance.

Course Objectives

The general objective of this course is to provide concepts related to information systems development in a systematic approach including foundations, planning, analysis, design, implementation and maintenance.

Unit Contents

1. System Development Fundamentals : 9 hrs

a. The Systems Development Environment

Introduction, Modern Approach of System Analysis and Design, Information System and its Type, Developing Information System and its Type, Developing Information Systems and the Systems Development Life Cycle, The Heart of the Systems Development Process, The Traditional Waterfall SDLC, Approaches for Improving Development, CASE Tools, Rapid Application Development, Service-Oriented Architecture, Agile Methodologies, extreme Programming, Object-Oriented Analysis and Design.

b. The Origins of Software

Introduction, System Acquistion, Reuse

c. Managing the Information Systems Project

Introduction, Managing Information Systems Project, Representing and Scheduling Project Plans, Using Project Plans, Using Project Management Software

2. Planning : 7 hrs

a. System Development Projects: Identification and Selection

Introduction, Identifying and Selecting Systems Development Projects, Corporate and Information Systems Planning.

b. System Development Projects: Initiation and Planning

Introduction, Initiating and Planning Syatems Development Projects, Process of Initiating and Planning IS Development Projects, Assessing Project Feasibility, Building and Reviewing the Baseline Project Plan

3. Analysis : 13 hrs

a. System Requirements

Introduction, Performing Requirements Determination, Traditional Methods for Determining System Requirements, Contemporary Methods for Determining System Requirements, Requirements Management Tools, Requirements Determination Using Agile Methodologies.

b. System Process Requirements

Introduction, Process Modeling, Data Flow Diagramming Mechanics, Using Data Flow Diagramming in the Analysis Process, Modeling Logic witth Decision Tables

c. System Data Requirements

Introduction, Conceptual Data Modeling, Gathering Information for Conceptual Data Modeling, Introduction to E-R Modeling, Conceptual Data Modeling and the E-R Model, Representating Super-types and Sub-types, Business Rules, Role of Packaged Conceptual Data Models-Database Patterns

4. Design : 12 hrs

a. Designing Databases

Introduction, Database Design, Relational Database Model, Normalization, Transforming E-R Diagrams into Relations, Merging Relations, Physical File and Database Design, Designing Fields, Designing Physical Tables

b. Designing Forms and Reports

Introduction, Designing Forms and Reports, Formatting Forms and Reports, Assessing Usability

c. Designing Interfaces and Dialogues

Introduction, Designing Interfaces and Dialogues, Interaction Methods and Devices, Designing Interfaces and Dialogues in Graphical Environments

5. Implementation and Maintenance : 4 hrs

a. System Implementation

Introduction, System Implementation, Software Application Testing, Intallation, Documenting the System, Traninig and Supporting Users, Organizational Issues in Systems Implementation

b. System Maintenance

Introduction, Maintaining Information Systems, Conducting Systems Maintenance

Text and Reference Books

Text Book

1. Jeffrey A.Hoffer, Joey George, Joe Valacich, “Modern Systems Analysis and Design”, 6/E, Prentice Hall India.

Reference Book

1. Jeffery Whitten, Lonnie Bentley, “Systems Analysis 7/E, McGraw-Hill