Victoria University
of Bangladesh

MID Term Assessment

Md Bakhtiar Chowdhury

ID: 2121210061

Department: CSE

Semester: Summer 2023

Batch: 21th

Course Title: System Analysis and Design

Course Code: CSI 311

Submitted To:

Umme Khadiza Tithi

Lecturer, Department of Computer Science & Engineering

Victoria University of Bangladesh

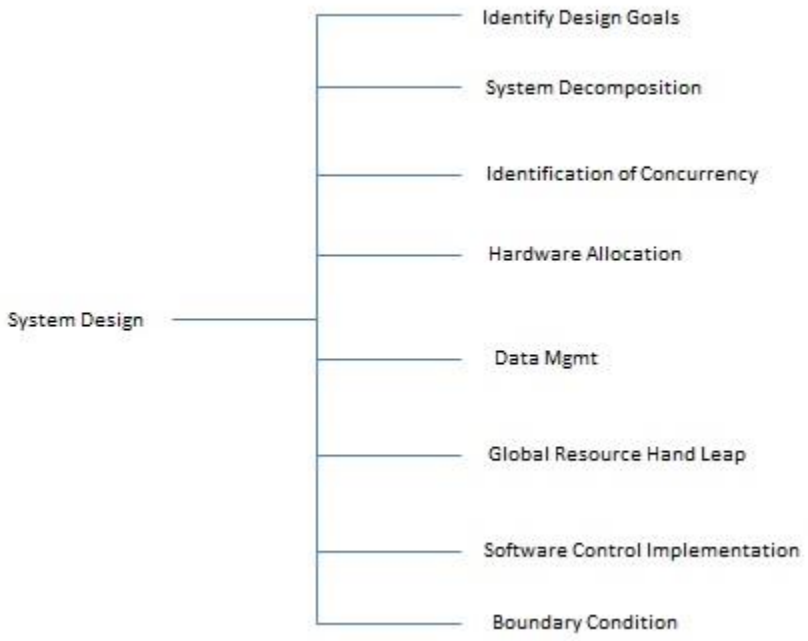
Submission Date: 18 August, 2023

Answer to the question no 1

 # What is System Design? Define Input to system design and outputs for System design?

System Design is a crucial phase in the software development lifecycle that follows system analysis. It involves creating a detailed blueprint or specification for the proposed system based on the requirements identified during the analysis phase. The main goal of system design is to define how the system will be structured, how its components will interact, and how it will fulfill the identified user needs and business objectives.

Example: Online Shopping System



Input to System Design:

1: User Requirements: During the system analysis phase, various user requirements were identified for the online shopping system, such as user registration, product search, shopping cart management, order placement, and payment processing.

2. System Analysis Artifacts: Use case diagrams were created to visualize the interactions between users (buyers) and the system, showing scenarios like "searching for products," "adding items to the cart," and "completing an order."

3. Functional Requirements: Detailed functional requirements were documented, specifying the behavior of each system component, such as the search algorithm, the checkout process, and the account management features.

4. Non-Functional Requirements: Performance expectations, security measures (e.g., user authentication, encryption), and scalability requirements were defined as part of the non-functional requirements.

5. Technology and Platform: The system will be built using a web-based architecture, utilizing technologies such as a front-end framework (e.g., React), a back-end server (e.g., Node.js), and a database system (e.g., MySQL).

Outputs of System Design:

1. **System Architecture:** The system design outlines the high-level structure of the online shopping system. It defines components like user authentication, product catalog, shopping cart, order processing, and payment gateway. It describes how these components interact and depend on each other.

2. Data Design: The system design specifies the database structure. For example, it defines tables for users, products, orders, and their relationships. It also outlines how data will be stored, retrieved, and updated in the database.

3. Interface Design: The system design includes the user interface (UI) design. This might consist of screen layouts showing how product listings, product details, shopping carts, and checkout screens will look. It defines the user interaction flow.

4. Security and Access Control Design: The system design outlines security measures such as user authentication (username/password or social logins), encryption of sensitive data (e.g., credit card information), and access control to ensure that only authorized users can perform certain actions.

5. Algorithm and Logic Design: If there are specific algorithms used in the system (e.g., for search functionality), the design might specify the details of these algorithms, how they work, and how they are implemented.

6. Prototypes and Mockups: The design might include interactive prototypes or mockups of key UI screens, allowing stakeholders to visualize the user experience.

7. System Specifications: The design documents provide detailed specifications for the development team, including APIs, data structures, and interaction protocols.

8. Documentation: Comprehensive documentation explaining the architecture, data flow, security measures, and other technical aspects of the system.

In this example, the System Design phase takes the initial requirements and analysis findings for the online shopping system and transforms them into a detailed plan that guides the development and implementation of the system.

Answer to the question no 2

 # Advantages of Structured analysis and structured design?

Structured analysis and structured design are methodologies used in the field of software engineering to develop and design complex systems. They offer several advantages that contribute to the successful development of high-quality and maintainable software. Here are some key advantages of structured analysis and structured design:

1. Clarity and Understandability: Structured analysis and design promote the use of standardized notations and techniques. This enhances the clarity and understandability of system requirements and design among team members, stakeholders, and developers. Everyone involved in the project can more easily comprehend the system's structure and functionality.

2. Modularity: Structured design encourages breaking down the system into smaller, manageable modules or components. This modular approach simplifies development, testing, and maintenance. Each module can be developed and tested independently, making the system more scalable and easier to modify.

3. Maintenance: With a clear modular structure, maintenance becomes more straightforward. When a change is needed, it's often isolated to a specific module, reducing the risk of unintended side effects in other parts of the system. This reduces maintenance costs and efforts.

4. Reusability: By designing components with clear interfaces and well-defined functions, structured design promotes reusability. These modular components can be reused in other projects, saving time and effort in future software development endeavors.

5. Error Detection: Structured analysis and design methodologies emphasize systematic analysis and thorough design documentation. This helps identify potential errors,

inconsistencies, or ambiguities early in the development process, reducing the likelihood of major issues in the final product.

6. Scalability: The modular nature of structured design allows for better scalability. As the system requirements change or expand, new modules can be added, and existing modules can be modified without disrupting the entire system.

7. Collaboration: Structured analysis and design provide a common framework and language for communication among team members. This facilitates collaboration by ensuring that everyone understands the system's structure and functionality, leading to more effective teamwork.

8. Reduced Development Time: While it may seem that structured analysis and design could increase development time due to documentation and planning, in the long run, they often lead to reduced development time by preventing misunderstandings, reducing rework, and streamlining the development process.

9. Documentation: Structured analysis and design methodologies require comprehensive documentation, which is beneficial for future reference, knowledge transfer, and maintaining system continuity.

10. Reduced Complexity: By breaking down the system into smaller, well-defined modules, the overall complexity of the system is reduced. This makes it easier to manage, understand, and troubleshoot.

Overall, structured analysis and structured design help create more organized, maintainable, and efficient software systems, leading to higher-quality outcomes and a better experience for both developers and end-users.

Answer to the question no 3

 # Describe Objectives and Elements of analysis modeling?

Analysis modeling is a crucial phase in software engineering, especially in the context of System Analysis and Design. It involves creating models and representations that help understand, visualize, and document the requirements and specifications of a system. The objectives of analysis modeling are to capture and describe the essential aspects of the system in a structured manner, ensuring a clear understanding of user needs and system functionality. The elements of analysis modeling are the building blocks or components used to construct these models. Let's explore the objectives and elements in more detail:

Objectives of Analysis Modeling:

1. Understanding User Needs: The primary goal of analysis modeling is to gain a deep understanding of user requirements, expectations, and the problem the system is intended to solve. By understanding these needs, the development team can design a system that truly meets user expectations.

2. Visualization: Analysis models provide a visual representation of the system, making it easier for stakeholders, including users and developers, to comprehend the system's structure, behavior, and interactions.

3. Requirements Elicitation: Analysis modeling techniques facilitate the systematic gathering of requirements from users and stakeholders. It helps identify both functional and non-functional requirements, ensuring that no critical aspects are overlooked.

4. Documentation: Analysis models serve as documentation for the system's design during the development process. This documentation helps in communication, knowledge transfer, and future reference, especially during maintenance or upgrades.

5. Communication: Analysis models act as a common language that bridges the gap between technical and non-technical stakeholders. They facilitate effective communication by providing a tangible representation of the system.

6. Verification and Validation: Analysis models allow for early validation of requirements. By visualizing the system's behavior and structure, potential issues or inconsistencies can be identified and addressed before moving on to the design and implementation stages.

Elements of Analysis Modeling:

1. Use Cases: Use cases describe the interactions between actors (users, external systems) and the system. They capture specific scenarios that demonstrate how the system will be used to achieve certain goals.

2. Requirements: Requirements capture the desired functionalities and qualities (non-functional requirements) of the system. They form the foundation for the system's design and development.

3. Data Flow Diagrams (DFD): DFDs illustrate how data flows within the system. They depict processes, data stores, data sources/destinations, and the data flow paths.

4. Entity-Relationship Diagrams (ERD): ERDs model the relationships between entities (data objects) in the system. They are particularly useful for database design and understanding data structures.

5. Class Diagrams: Class diagrams depict the static structure of the system, showing classes, attributes, associations, and inheritance relationships.

6. Activity Diagrams: Activity diagrams illustrate the flow of activities or processes within the system. They are especially useful for modeling business processes and workflows.

7. State Diagrams: State diagrams show the different states that an object can be in and the transitions between those states. They are used to model the behavior of objects over time.

8. Sequence Diagrams: Sequence diagrams depict the interactions and messages exchanged between objects or components in a specific scenario, showing the order of events.

These elements, when used appropriately and collectively, form a comprehensive analysis model that helps achieve the objectives of understanding, visualizing, and documenting the system's requirements and specifications.

Answer to the question no 4

 # Write down Bottom-Up strategys advantages and Disadvantage?

Bottom-Up Strategy

The Bottom-Up strategy is an approach used in various fields, including software development, problem-solving, and system design. It involves starting with the smallest components or details and gradually building up to create a larger, more complex system. This strategy is often associated with incremental development and can be contrasted with the Top-Down approach, which starts with the overall system and breaks it down into smaller components.

Advantages of Bottom-Up Strategy:

1. Modularity: One of the key advantages of the Bottom-Up strategy is its focus on modularity. By developing and testing small, independent components first, it becomes easier to manage, reuse, and maintain these modules. This modularity enhances system flexibility and scalability.

2. Early Validation: Since each small component is developed and tested individually, validation of these components can occur early in the development process. This helps in identifying and fixing issues at a granular level, reducing the risk of major problems in the final system.

3. Incremental Progress: Bottom-Up development allows for incremental progress. Developers can see tangible results as individual components are completed and integrated, providing a sense of accomplishment and ensuring that progress is visible even before the entire system is complete.

4. Faster Iterations: Bottom-Up allows for faster iterations and frequent updates. As each component is developed, tested, and integrated, it can be refined and improved in subsequent iterations, leading to faster overall development.

5. Flexibility and Adaptability: Bottom-Up strategy lends itself well to adaptability. Changes in requirements or the addition of new features can be integrated more easily into the existing modular structure.

Disadvantages of Bottom-Up Strategy:

1. Integration Complexity: The Bottom-Up strategy, while focused on modularity, can lead to integration complexities. Ensuring that all components work seamlessly together can be challenging, especially if there are unexpected dependencies or interactions.

2. Lack of Early System View: Since the Bottom-Up approach starts with small components, there might be a lack of a comprehensive system view in the early stages of development. This can make it difficult to ensure that the overall system design aligns with the intended goals.

3. Potential Redundancy: Without careful planning and coordination, there's a risk of redundant efforts in developing similar functionalities within different components. This can lead to inefficiencies and increased development time.

4. Difficulty in System-level Decisions: Some system-level decisions may need to be deferred until the components are integrated, making it challenging to make informed architectural choices early in the development process.

5. Testing Complexity: While individual components can be tested effectively, testing the interactions and integration of all components can be complex and time-consuming, especially if integration testing is not well-structured.

In summary, the Bottom-Up strategy offers advantages such as modularity, early validation, incremental progress, and flexibility. However, it can also present challenges in terms of integration complexity, the lack of early system view, and potential redundancy. The choice of strategy depends on the specific project, its requirements, and the development team's preferences and expertise.

>>>>>>END<<<<<<