OOAD Assignment 2

Group-1

Name Enrollment No

Anant Jain 22114005

Dhas Aryan Satish 22117046

Divij Rawal 22114031

Parit Gupta 22117100

Pratyaksh Bhalla 22115119

Roopam Taneja 22115030

Q1: Write a short note on different techniques for software design.

A: A large number of software design techniques are available. We can roughly classify these techniques into procedural and object-oriented approaches. Though these two design approaches are radically different, they are complementary rather than competing techniques.

Function-oriented Design

Function-oriented design is a technique that focuses on the functions or services that the software system provides to the users or other systems. Its main features are:

- **Top-down decomposition**: The system is viewed as a black box that provides certain high-level functions. Each high-level function is successively refined into more detailed subfunctions until they are simple enough to be implemented. For example, consider a function create-new-library-member. This high-level function may be refined into the following subfunctions:
  - assign-membership-number
  - create-member-record
  - print-bill
- Each of these subfunctions may be split into more detailed subfunctions and so on.
- **Centralized system state**: The system state is the values of certain data items that determine the response of the system to a user action or external event. It is centralized. Such data in procedural programs usually has a global scope is are shared by many modules.

Some advantages of function-oriented design are:

- It is easy to understand and follow the logic of the system.
- It allows reuse of existing functions or libraries.

Some disadvantages of function-oriented design are:

- It may lead to poor modularity and coupling, as different functions may depend on the same data or affect each other's behavior.
- It may not capture the real-world entities and their relationships well.

Some common function-oriented design methods are:

- Structured design by Constantine and Yourdon
- Jackson's structured design by Jackson
- Warnier-Orr methodology

Object-oriented Design

Object-oriented design is a technique that focuses on the objects or entities that make up the software system. It extensively uses abstraction and decomposition. The main features of object-oriented design are:

- **Objects**: An object is an entity that has its own data and functions that operate on the data. The data internal to an object cannot be accessed directly by other objects and only through invocation of the methods of the object. For example, in a library automation software, each library member may be a separate object with its own data (such as name, membership number, borrowed books, etc.) and functions (such as borrow book, return book, renew membership, etc.).
- **Classes**: A class is a collection of similar objects that share the same attributes and behaviors. A class defines the structure and behavior of its objects. For example, there may be a class called LibraryMember that defines the attributes and methods for all library members.
- **Message passing**: Objects communicate with each other by sending and receiving messages.
- **Decentralized system state**: The system state is distributed among the objects of the system. Each object maintains its own state and is responsible for managing it. There is no globally shared data in the system.

Some advantages of object-oriented design are:

- It allows better modularity and encapsulation, as each object has its own data and behavior.
- It captures the real-world entities and their relationships well.
- It supports inheritance and polymorphism, which enable reuse and extensibility of code.

Some disadvantages of object-oriented design are:

- It may be difficult to identify the appropriate objects and classes for a given problem domain.
- It may require more memory and processing time than procedural programs.

Some common object-oriented design methods are:

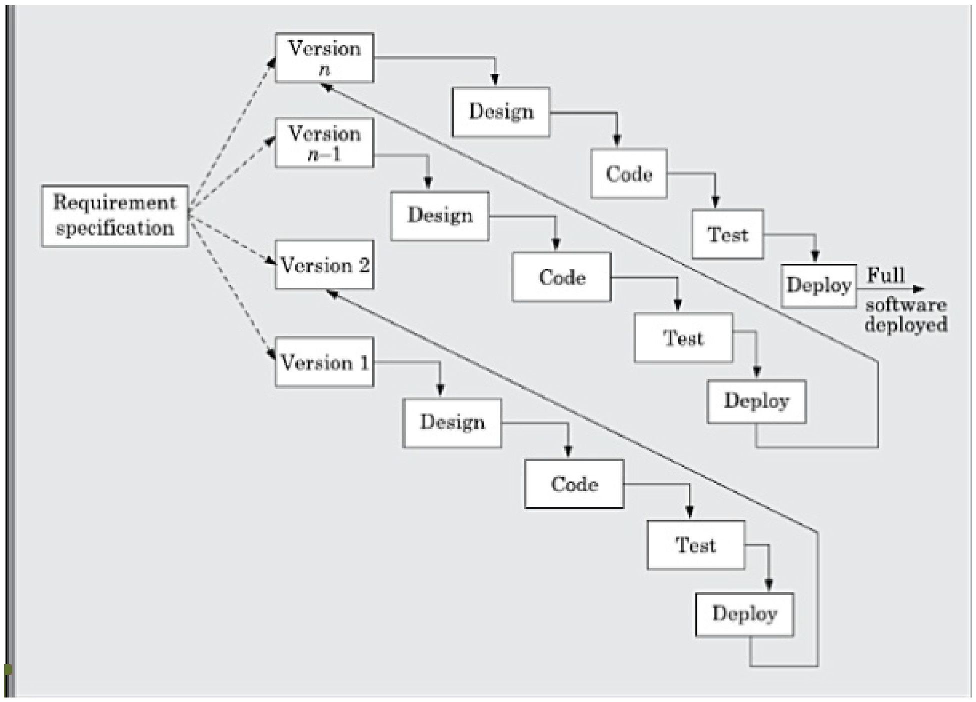
- Unified Modeling Language (UML)
- Object Modeling Technique (OMT) by Rumbaugh et al.
- Object-Oriented Software Engineering (OOSE) by Jacobson et al.

Q2: Write a short note on unified development process.

A: In the incremental life cycle model, the software requirements are initially divided into various modules or features, which can be incrementally built and delivered.

- Initially, the development team focuses on creating the essential features of the system, referred to as the core features. These core features operate independently without requiring support from other features. In contrast, non-core features rely on services from the core functionalities.
- After establishing the core features, subsequent versions of the software are refined to include additional functionalities and capabilities.
- With each version delivered to the customer, feedback is gathered and incorporated into the subsequent versions, ensuring that the evolving product aligns with the customer's expectations and requirements.
- Following the gathering and specification of requirements, the software requirements are segmented into multiple versions. Starting with the core version (version 1), each subsequent increment involves the iterative waterfall model for development, with each version deployed at the customer site. Once the final

version has been developed and deployed at the client site, the complete software is released.



Advantages:

Low Error Rate: Since the software gets used by the user from the beginning the software has grater reliability.

Use Case Driven: In the Unified Process, use cases are used to capture the functional requirements and to define the contents of the iterations. Each iteration takes a set of use cases or scenarios from requirements all the way through implementation, test and deployment.

Q3. Assume you are an expert working for a small-scale organisation to build up an accounting framework. Would you choose the unified process to build up the framework, or would you lean toward one of the other methodologies? Why?

A] The Unified Process is one approach, but there are several others to consider, such as Waterfall, Agile, or a hybrid approach. Here are some factors to consider:

1. **Project Complexity:** The Unified Process proves to be highly effective for managing extensive and intricate projects. However, if our accounting project is relatively straightforward, it might

be wise to opt for a more streamlined method like Waterfall or Agile, as they offer greater adaptability.

2. **Project Size:** Considering the limited resources of small organizations in terms of time and finances, it's crucial to align our chosen methodology with our size. The Unified Process tends to involve more paperwork, which could be overwhelming for a smaller project.

3. **Flexibility:** If our accounting project experiences frequent changes, or if our organization is still in the process of defining its requirements, Agile is an excellent choice due to its inherent adaptability and responsiveness to alterations.

4. **Documentation and Rules:** Accounting projects often require extensive documentation and strict adherence to rules and regulations. Waterfall and the Unified Process emphasize these aspects more than Agile. Therefore, if strict compliance and extensive documentation are paramount, we should consider a more traditional approach.

5. **Project Timeline:** It's worth noting that the Unified Process may necessitate a more extended planning and design phase, which can extend the project's overall duration. If our priority is to expedite the implementation of our accounting system, Agile could be a more suitable choice to ensure a quicker turnaround.

I would choose the Unified Process to build the accounting framework for our small-scale organization because:

1. **Thorough Planning**: The Unified Process is a good choice because it focuses on careful planning and documentation. This helps ensure that we think through all aspects of our accounting framework.

2. **Flexibility**: It allows us to adapt and make changes as needed, which is important for refining our accounting framework based on feedback and changing needs.

3. **Risk Management**: The Unified Process helps us identify and reduce financial and compliance risks, making our accounting framework more reliable.

4. **Phases and Milestones**: It breaks the project into clear phases and milestones, making it easier to track progress. This is important for financial reporting and compliance.

5. **Documentation**: The formal documentation at each stage is vital for auditing and maintaining the accounting framework in the long run, ensuring transparency in financial processes.

Q4. Assume you are an expert building up a new information framework to robotize the business exchanges and oversee stock for each retail store in an enormous chain. The framework would be introduced at each store and exchange information with a centralized server PC at the organization's administrative center. Would you choose the unified process to build up the framework, or would you lean toward one of the other methodologies? Why?

A] The development process used to design an information framework to automate business transactions and manage inventories for a big retail shop chain is determined by a variety of variables. The choice should be chosen after taking into account the project's particular requirements, limitations, and goals. Using the Unified Process (UP) in comparison to various other techniques is compared as follows:

Unified Process (UP):

The paradigm for software development known as "Unified Process" is iterative and incremental and focuses on requirements management, system design, implementation, testing, and maintenance. Its versatility and adaptability make it appropriate for big and complicated projects.

Benefits of utilising UP for this retail store system:

1. **Risk Management:** UP places a premium on risk management and early testing, which is vital for a mission-critical system such as retail inventory management. It makes it possible to identify and address any problems early on.
2. **Iterative Development:** UP allows iterative development, allowing you to progressively modify the system. In a retail setting, needs might change often, and an iterative strategy can better adapt to these changes.
3. **Traceability:** UP promotes thorough documentation and requirement traceability, both of which are necessary for regulatory compliance and audits in the retail industry.
4. **Cooperation:** UP encourages cross-functional team cooperation, which is advantageous when dealing with both business and technological elements of retail management.

Other Approaches to Consider:

1. **Agile (such as Scrum):**

- Choose Agile if the chain of stores prioritises adaptability, quick answers to shifting market conditions, and regular releases.

- Agile project management techniques are great for tasks when needs are not completely understood up front, as is typical in the retail industry where consumer preferences change.
- Agile encourages cooperation and communication both within the development team and with stakeholders.

2. **Waterfall:**

- If the needs of the project are clear from the beginning and unlikely to change dramatically, use waterfall.
- Waterfall may be appropriate for projects with a rigid, linear approach to development and where regulatory compliance is of the utmost importance.

3. **DevOps:**

- Consider combining DevOps behaviours with another approach if continuous integration, delivery, and operational efficiency are important.

As discussed in the above question , there are certain factors to be discussed before choosing the desired methodology. Without considering the factors, it is not possible to choose the actual desired methodology.