Software Engineering
Assignment - 3

Harshit Chaurasia
2301010210

Sol 1) The unified approach supports a layered, object oriented design by focusing on modeling and iterative development using UML, ensuring a clear seperation of concerns a clear sepration of concerns and high reusability.

1. Layered Design Support:
   - UML: Used to explicity model the system as distinct packages (layers) with clear interfaces, ensuring loose coupling between them.
   - Abstraction: Promotes identifying high level classes that belong to specific layers, facilating traceability and architectural integrity.

2. LMS layer Breakdown

| Layer | → | Responsibility |
|---|---|---|
| • User Responsibility (UI) | → | Handles user interaction, input capture, and data display. |
| • Business Logic | → | Contains core business rules, processing, and workflows. |
| • Data Access | → | Manages communication with the database (CRUD) operations. |

3. Justification of OOP for Scalability

| Principle | → | Justification | → | Scalability |
|---|---|---|---|---|
| • Inheritance | → | create class heirarchies (eg student inherits User) | → | New roles are added quickly by inheriting common features, reducing code duplication. |

- Encapsulation → Hides internal class data exposing only public interface → Enforces layer independence allowing changes in the data layer (eg database) without breaking the Business logic layer.
- Reuse (Polymorphism) → Allow different objects to share a common interface (eg. display Profile) → Components like a Notification Service can be reused across all system modules (assignments, announcements), simplifying extensions.

## 01|2) UML Structural Modeling for Hotel Booking System

### 1) UML Diagrams:

- Class Diagram: Shows core entities (User, Room, Booking, Payement) and their relationships (eg. Aggregation: Hotel → Room; Composition: User → Booking)
- Component Diagram: Shows the system's physical structure (eg: Booking Service, Payement Gateway) and their interactions via provided/required interfaces.

### 2) Key Class Details

| Classes | → Attributes | → Operations |
|---|---|---|
| User | → userID, name, Email | → login(), updateProfile() |
| Room | → roomNumber, type, pricePerNight | → checkAvailability(), reserveRoom() |
| Booking | → bookingID, StartDate, totalCost, Status | → confirmBooking(), cancelBooking(), processPayement() |

3. Benefits to System Maintainability

Structural modeling improves the system by:

- Understanding: Provides a visual blueprint (Class diagram) of the domain, allowing new team members to quickly grasp the system's logical design.
- Maintainability: Establishes clear boundaries and loose coupling (Component Diagram). Changes in one area (eg. updating the payement Gateaway) are isolated, reducing the risk of bugs in other components.
- Scalability: Identifies independent, reusable components, which guides development toward a robust, scalable architecture.

Q13) Software Reliability and fault analysis for smart home System

In the context of smart home security system, hardware reliability refers to the physical components (like sensors, cameras and batteries) functioning correctly without physical failure due to wear, tear or manufacturing correctly without over time. In contrast, software reliability is the probability that the system's code (the alarm logic, network handlers, and firmware) will execute correctly without exhibiting defects (bugs) over a specified period under defined conditions. Critical software faults include a logic error in Alert Threshold (eg. using $\geq$ instead of $>$ in comparison), which could lead to a false Alarm, or worse, a missed detection of the

logic is Inverted. Another fault is a Race condition in Multi-threading where concurrent processes improperly access shared data, potentially causing a System freeze/ crash and halting security monitoring. finally, Missing Exception handling for non-ideal conditions, such as a sudden loss of wi-fi connection, can lead to a system crash and a loss of Remote Access /data, preventing critical push notifications. To enhance system reliability, three key strategies are vital: Redundancy (eg. dual power sources and N-version programming for critical logic), Robust Testing (including stress testing and boundary analysis to uncover bugs before deployement), and exception handling (using mechanism like try-catch blocks and watchdog timers to manage running errors and ensure graceful recovery instead of failure).

Sol 4) Reliability Modeling and standars for health Monitoring device

1) Logarithmic Poisson Reliability ~~growth~~ Model (LPRM)

The LRPM models Reliability growth, showing failure intensity decreases exponentially as faults are fixed

$$\lambda(\mu) = \lambda_0 e^{-\theta\mu}$$

. use : It estimates the system's current failure rate based on the initial rate ($\lambda_0$) and the number of failures removed ($\mu$)

- Example : (Hypothetical $\theta = 0.01$): If $\lambda_0 = 0.06$ and $\mu = 20$, the estimated current failure intensity is $\lambda(20) \approx 0.0491$ failures / unit time, indicating improvement.

2. CMM levels for quality

CMM (Capability Maturity Model) provides a roadmap for processes improvement to achieve reliable software :

- Level 3 : (Defined) : Requires standardized development and testing processes across the organisation, leading to predictable quality.
- Level 4 (Quantitively Managed) : Requires statistical measurement and control of process and product quality (failure rate), which is vital for a health device to meet strict limits.

3. ISO 9001 vs CMM:

ISO 9001 is more appropriate for certification while CMM is better for the process of achieving high reliability.

| Standard | → Focus | → Health critical Appropriateness |
|---|---|---|
| ISO 9001 | → Quality Management System (QMS) for all processes (what to do) | → Essential for regulatory compliance and demonstrating a foundation of consistent quality management to external bodies |
| CMM | → Software development Process Maturity (How to do it) | → Crucial for achieving the high reliability needed; Level 4 provides the necessary quantative control over failure rates. |