Mahavir Education Trust's

Shah & Anchor Kutchhi Engineering College

Chembur, Mumbai 400 088

(An Autonomous Institute Affiliated to University of Mumbai)

Name: Dwijesh Mistry
Class: TY09
Roll no: 39

Sub: Software Engineering

Assignment on Module 3

Q1. Consider the scenario, Module A has tightly coupled dependencies on other modules but high internal cohesion. Module B has low coupling but moderate cohesion. The modules are part of a large, distributed system.

Analyze the potential impact of the coupling and cohesion levels of these modules on system maintenance and scalability. Which module would you recommend for reuse in a new project? Justify your choice. [10 Marks]

Ans: Impact of Coupling and Cohesion

- Module A: Tightly Coupled, High Cohesion
    - Tightly coupled modules depend heavily on other parts of the system, making any change propagation complex and risky.
    - High internal cohesion means all functionalities within Module A support a single, well-defined purpose, leading to excellent reliability and error isolation.
    - Difficulties include:
        - Maintenance: Updates or bug fixes can affect many related modules, increasing risk and workload.
        - Scalability: Scaling Module A might require changes in other coupled modules, complicating expansion.
        - Testing: Isolating issues in tightly coupled modules is challenging because dependencies may mask or propagate errors.
- Module B: Low Coupling, Moderate Cohesion
    - Low coupling ensures that Module B operates mostly independently of other parts, making maintenance easier and enabling modular upgrades.
    - Moderate cohesion indicates the module performs related but not perfectly unified tasks, which may slightly reduce reliability but still keeps maintenance manageable.
    - Advantages include:
        - Maintenance: Changes to Module B seldom impact others, streamlining upgrades and bug fixes.
        - Scalability: The module can be moved, reused, or scaled with minimal system-wide impact.
        - Testing: Independent functionality allows for thorough, isolated testing.
- Recommendation for Reuse

- Module B is best suited for reuse in new projects due to its low coupling; it requires fewer changes and integrates smoothly into diverse systems.
- Moderate cohesion is acceptable in many scenarios, especially when the module is likely to need adaptation.

Q2. Consider the scenario,after deploying a customer relationship management (CRM) software, frequent changes are made to improve features and enhance user interface responsiveness. Simultaneously, there is a plan to prevent future security vulnerabilities.

Distinguish between perfective and preventive maintenance in this context. Discuss how the maintenance team should prioritize and implement these maintenance types to ensure software reliability and customer satisfaction. [10 Marks]

Ans: **Perfective Maintenance**

- **Definition**: Involves enhancing software features, performance, or user-friendliness based on user feedback or evolving business needs.
- **Typical Activities**:
  - Adding new modules or features (like reporting tools or dashboards).
  - Improving interface responsiveness and usability (faster load times, modernized UI).
  - Removing redundant or obsolete features.
- **Focus**: Meeting users' current needs and increasing customer satisfaction by evolving with market expectations.
- **CRM Context Example**: Upgrading the CRM interface so sales teams navigate faster or adding customer communication history filters.

Preventive Maintenance

- **Definition**: Proactively addresses potential software issues that could cause problems later, including security risks, maintainability, and performance bottlenecks.
- **Typical Activities**:
  - Refactoring legacy code to reduce vulnerabilities.
  - Updating third-party libraries and dependencies.
  - Regular security audits and patching latent bugs before they become issues.
- **Focus**: Ensuring long-term stability, security, and reliability, even if users don't immediately notice these changes.
- **CRM Context Example**: Running periodic penetration tests and updating encryption protocols to prevent future breaches.

Perfective vs Preventive Maintenance Table

Feature	Perfective Maintenance	Preventive Maintenance
Goal	Enhance functionality & usability	Prevent future problems (e.g., security, maintainability)
Triggers	User feedback, new business needs	Risk assessments, security advisories, code reviews
Typical Activities	Add features, improve UI, optimize performance	Patch vulnerabilities, refactor code, update components
Immediate Visibility	Yes, improvements noticed by end-users	Rarely, benefits realized by smoother operation
Example	New dashboard, better menu design	Database encryption update, removing deprecated API

Prioritization Strategy

- **Short-term Priority:**
  - Emphasize **perfective maintenance** when rapid user feedback or market pressure makes new features and performance enhancements urgent.
  - Schedule **preventive maintenance** right after major releases or at regular intervals to address covert risks without delaying visible improvements.
- **Long-term Reliability:**
  - Use risk analysis to identify security or stability tasks that cannot wait (e.g., urgent security patches) and escalate their priority.
  - Document all preventive actions and align them with compliance or industry regulations for continuous improvement.
- **Maintenance Implementation Best Practices:**
  - Develop a maintenance calendar alternating between perfective and preventive tasks.
  - Integrate preventive tasks into sprint cycles, pairing visible upgrades with backend cleanups.
  - Establish KPIs for both customer satisfaction (from perfective work) and security/reliability metrics (from preventive work).

Conclusion

To ensure customer satisfaction and robust software reliability, **perfective maintenance** keeps users engaged and pleased with improvements, while **preventive maintenance** protects the CRM from future setbacks or breaches. Strategic scheduling—balancing rapid feature releases with regular preventive updates—will provide the best long-term results for your CRM platform

Q3. Assume that you are the Test Lead for a mobile application development team working in an Agile environment. A new login and user profile module is being developed for the upcoming sprint. Key features include user registration, secure login, and updating profile information. [5 Marks]

Based on this scenario, apply the concept of Test plan and outline a detailed test plan for the sprint. Your plan should address the following components:

- Test objectives
- Scope of testing (in-scope and out-of-scope items)
- Key testing activities (e.g., types of testing)
- Entry and exit criteria for the testing phase.

- Ans: **Test Objectives:**
  - Validate the core functions of user registration, secure login, and profile updating.
  - Ensure the security of the login process, including password encryption and session management.
  - Confirm that profile information updates are saved correctly and data integrity is maintained.
  - Provide a smooth and error-free user experience across supported devices.
- **Scope of Testing:**
  - ***In-scope:***

        - Functional testing of registration, login, and profile update features.
        - Security testing focused on authentication and authorization processes.
        - Compatibility testing across different mobile devices and OS versions.
    - ***Out-of-scope:***
        - Payment gateway integrations.
        - Social media or third-party login methods.
        - Push notifications or unrelated app modules.
- **Key Testing Activities:**
    - Conduct functional tests with valid and invalid input scenarios for registration and login workflows.
    - Perform security tests to check password encryption, secure data transmission, and session expiry.
    - Execute usability tests to evaluate UI responsiveness and clarity of user messages.
    - Perform regression testing to ensure new changes do not break existing login and profile functionalities.
- **Entry Criteria:**
    - Completion and approval of all user stories and requirements related to the login/profile module.
    - Development finished with successful unit testing.
    - Test environment fully set up with necessary test data populated.
    - Test cases and scripts prepared and reviewed for completeness.
- **Exit Criteria:**
    - All planned functional, security, usability, and regression test cases have been executed.
    - No critical or high-severity defects remain unresolved.
    - User acceptance testing (UAT) has been completed successfully with stakeholder approval.
    - Test results and defect logs are documented and reviewed.
    - Test environment cleaned and ready for future sprints.