Sanika Patankar
9563
TE COMPS B

SE ASSIGNMENT 2

Rubrics :

Indicator	Average	Good	Excellent	Marks
Organization (2)	Readable with some mistakes and structured (1)	Readable with some mistakes and structured (1)	Very well written and structured (2)	
Level of content(4)	Minimal topics are covered with limited information (2)	Limited major topics with minor details are presented(3)	All major topics with minor details are covered (4)	
Depth and breadth of discussion(4)	Minimal points with missing information (1)	Relatively more points with information (2)	All points with in depth information(4)	
Total Marks(10)				

1. What is risk assessment in the context of software projects, and why is it essential?

- **Definition**: Risk assessment in software projects is the process of identifying, analysing, and managing potential risks and uncertainties that could affect project success.
- **Importance**:
    1. Early problem identification.
    2. Efficient resource allocation.
    3. Cost control.
    4. Schedule management.
    5. Quality assurance.
    6. Stakeholder communication.
    7. Informed decision-making.
    8. Project success.
- **Steps in Risk Assessment**:
    1. Risk Identification: Identify potential risks.
    2. Risk Analysis: Assess probability and impact.
    3. Risk Prioritization: Focus on critical risks.
    4. Risk Mitigation or Management: Develop strategies.
    5. Monitoring and Review: Continuously track and adapt risk management strategies.

In summary, risk assessment is vital for effective project management by identifying and addressing potential issues and uncertainties in software development projects. It contributes to successful project outcomes.

2. Explain the concept of software configuration management and its role in ensuring project quality.

Software Configuration Management (SCM) is a set of processes and tools used to systematically manage, control, and track changes in software development projects. It involves managing various software artefacts, including source code, documentation, libraries, and configurations, throughout the project's lifecycle. SCM plays a crucial role in ensuring project quality for several reasons:

- **Version Control** keeps track of different software versions to prevent conflicts and mistakes.
- **Change Tracking** maintains a history of changes, aiding debugging and accountability.
- **Baseline Management** allows for creating snapshots of software at specific points for testing and deployment.
- **Parallel Development** supports multiple teams working simultaneously without integration issues.
- **Change Control** regulates and documents software changes to maintain integrity.
- **Release Management** ensures the correct versions are included in each release.
- **Auditing and Compliance** aids in regulatory compliance and quality assurance.
- **Quality Assurance and Testing** provides controlled testing environments and defect reproduction.
- **Documentation Management** includes features for managing project documentation.
- **Efficiency and Productivity** streamlines processes, reducing errors and improving productivity.
- **Risk Mitigation** identifies and mitigates risks associated with code changes.

SCM is crucial for maintaining software quality, integrity, and compliance with standards throughout the development lifecycle.

3. How do formal technical reviews (FTR) contribute to ensuring software quality and reliability?

Formal Technical Reviews (FTRs) are a systematic and structured approach to evaluating software artefacts, such as code, design documents, and requirements, to ensure quality and reliability in software development. FTRs contribute to quality and reliability in several ways:

- **Defect Detection**: FTRs identify and address defects early in the development process, preventing issues from propagating.
- **Knowledge Sharing**: FTRs promote collaboration, leading to a shared understanding of the project and quality expectations.
- **Consistency and Compliance**: Ensure that software artifacts adhere to coding standards, design guidelines, and project requirements.

- **Requirement Verification**: Verify that the software aligns with specified requirements, ensuring it functions correctly.
- **Traceability**: Establish traceability between project artifacts, reducing integration issues.
- **Improvement of Software Design**: Identify and address design flaws and inefficiencies for a more robust architecture.
- **Risk Mitigation**: Identify and address project risks early, ensuring proactive risk management.
- **Verification of Coding Practices**: Ensure that coding practices follow best practices for reliability and maintainability.
- **Documentation Review**: Review and maintain project documentation for better understanding and reliability.
- **Continuous Improvement**: FTRs are part of an iterative process that fosters ongoing enhancement of software quality and reliability.

4. Describe the process of conducting a formal walkthrough for a software project.

- **Preparation**:
    - Select the artefacts for review.
    - Schedule a meeting with relevant participants.
    - Distribute artefacts to participants in advance.
- **Introduction**:
    - Start with an opening meeting to outline the purpose and objectives.
    - Clarify roles and responsibilities.
- **Review Session**:
    - Presentation by the author/presenter.
    - Discussion, questions, and issue identification.
    - Decision-making and issue prioritisation.
- **Documentation**:
    - Record all identified issues and action items.
- **Closure**:
    - Summarise key findings, action items, and decisions.
    - Conclude the formal walkthrough session.
- **Post-Review Activities**:
    - Resolve identified issues and address action items.
    - Update the reviewed artefact(s).
- **Documentation and Reporting**:
    - Prepare a formal review report.
    - Distribute the report to stakeholders.

The formal walkthrough process is vital for quality assurance, promoting collaboration, and ensuring software artefacts meet quality standards and requirements.

5. Why is it important to consider software reliability when analysing potential risks in a project?

- **User Satisfaction**: Reliable software ensures a positive user experience.
- **Financial Impact**: Unreliable software leads to increased costs and potential financial losses.
- **Project Delays**: Reliability issues can cause project timeline delays.
- **Productivity Loss**: Unreliable software disrupts productivity for users and the development team.
- **Competitive Advantage**: Reliable software can be a competitive differentiator.
- **Legal and Regulatory Compliance**: Some industries require reliable software to meet legal and regulatory standards.
- **Data Integrity**: Reliability is crucial to protect data integrity.
- **Customer Trust**: Unreliable software erodes customer trust, which is challenging to regain.
- **Maintenance Burden**: Maintenance to fix reliability issues consumes resources.
- **Reputation and Brand Image**: Software reliability affects an organization's reputation and brand image.
- **Market Acceptance**: Some markets require specific reliability standards for software acceptance.

Consideration of software reliability in risk analysis helps mitigate these potential negative consequences.