Jenny Lopes
TE COMPS B
9618

SE ASSIGNMENT 2

Q1] Risk assessment in software projects is the process of idenfying, analyzing, and prioritizing potential risks or uncertainties that could affect the project's success. It involves evaluating the likelihood and impact of these risks and developing strategies to mitigate or manage them. This is crucial for several reasons:

a) Project success: Identifying and addressing risks early can prevent project failure or delays, ensuring that the project is completed on time and within budget.

b) Cost Control: Effective risk assessment helps allocate resources efficiently, reducing the likelihood of unexpected expenses or overruns.

c) Quality Assurance: It ensures the quality of the software by identifying and mitigating risks related to defects, security vulnerabilities, or performance issues.

d) Stakeholder Confidence: Transparent risk assessment and management build trust among project stakeholders, such as clients, by demonstrating a proactive approach to project challenges.

e) Schedule Adherence: Mitigating risks can help in adhering to project timelines, which is essential for meeting deadlines and market demands.

f) Resource Allocation

g) Legal and Regulatory Compliance

Q2] Software Configuration Management (SCM) is a set of processes and practices that help control and mange changes to software throughout its development lifecycle. Its primary role is to ensure project quality by maintaining the integrity, consistency and traceability of software components and their related documentation. Here's how SCM accomplishes this:

(1) Version Control: SCM tools, such as Git, enables developers to track changes to the source code.

(2) Configuration Identification: SCM defines and manages software components, including source code, documentation, and other artifacts.

(3) Change Control: SCM establishes a formal process for requesting, reviewing and approving changes to the software.

(4) Configuration Status Accounting: SCM keeps a record of the status of all software components and their versions.

(5) Configuration Auditing: Regular audits ensure that the software configuration aligns with the defined standards and requirements.

(6) Baseline management: SCM creates baselines, which are stable and well-tested configurations of the software.

(7) Traceability: SCM establishes traceability between various project artifacts, such as requirements, design, code and testing.

(8) Build and Release Management: SCM tools help automate the build and release processes, ensuring that correct components are assembled to create the final product.

(9) Collaboration: SCM facilitates collaboration among team members by providing a centralized repository for code and related assets.

Q3] Formal Technical Reviews (FTRs) play a crucial role in ensuring software quality and reliability by facilitating thorough examination and verification of software artifacts. Here's how FTR's contribute:

(1) Defect Identification: FTR's involve a systematic examination of software documentation, code and design.
(2) Knowledge sharing: FTRs encourage knowledge sharing among team members.
(3) Consistency and standards: FTRS ensure that the software conforms to established coding and design standards.
(4) Improved Documentation: Through FTRs, documentation quality is enchanced for understanding the software over time.
(5) Risk Mitigation: By identifying and addressing issues early, FTRs reduce the risk of major problems emerging later in the software development process, which can be costly and time-consuming to fix.
(6) Requirement Validation: FTR's help verify that software aligns with the specified requirements.
(7) Enhanced communication: FTRs promote communication and collaboration among team members.
(8) Quality Control: FTRs are a structured quality control process that ensures that software artifacts meet predefined quality standards.
(9) Time and cost savings.
(10) Continuous Improvement.

Q4] The process for conducting a formal software project walkthrough:

(i) Preparation:
- Schedule and define the purpose.
- Select artifacts and distribute them.

(ii) Participant Preparation:
- Participants review artifacts and make notes.

(iii) Conduct the walkthrough:
- Gather the team and appoint a mentor.
- Present, discuss and document issues.

(iv) Issue Tracking and Resolution:
- Assign issues for resolution.
- Follow up on issue status.

(v) Documentation:
- Create a summary report and share it.

(vi) Closure:
- Close the process when issues are resolved.

This streamlined process ensures a collaborative review, issue resolution, and improved software quality.

Q5] i] User Satisfaction: Unreliable software can lead to a poor user experience, causing frustration and dissatisfaction.

ii] Cost Implication: Unreliable software can lead to higher support and maintainance costs

iii] Project Delays: Development teams divert their attention from planned tasks to address defects and stability problems.

iv] Reputation Damage: Reliability problems can damage an organization's reputation.

v] Legal and Compliance Risks: Depending on the domain, unreliable software can result in legal and compliance risks.

vi] Security vulnerabilities: Unreliable software can be more susceptible to security vulnerabilities, putting sensitive data at risk.

vii] Operational Disruption: Software that's not reliable can disrupt business operations, causing downtime and affecting productivity.

viii] Maintainance Burden: Unreliable software often requires constant maintenance, diverting resources from new development initiatives and innovation.

ix] Stakeholder Confidence: Stakeholders, including investors and projects sponsors, may lose confidence in the project's success if reliability risks are not adequately addressed.

x] Competitive edge: In today's competitive landscape, reliable software can be a key differentiator.