Nash Dabre
9767

Fr. Conceicao Rodrigues College of Engineering,Mumbai
SOFTWARE ENGINEERING (CSC601)

Assignment -II

Date: 17-10-23

CO5: Identify risks, manage the change to assure quality in software projects.

Assignment 2

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

   Risk assessment in the context of software projects refers to the process of identifying, analyzing, and mitigating potential risks that can impact the successful completion of a software development project. These risks can take various forms, including technical, financial, organizational, and operational issues. Risk assessment is essential in software projects for several reasons:

   Project Success: Identifying and managing risks early in a project's lifecycle can significantly increase the likelihood of project success. By addressing potential issues proactively, you can avoid costly delays, budget overruns, and project failures.

   Cost Control: Risk assessment helps in budget planning and control. By identifying financial risks such as unexpected expenses or budget constraints, you can allocate resources more effectively and make informed decisions about how to mitigate or manage these risks.

   Resource Allocation: Identifying resource-related risks allows you to allocate the right people and assets to the project. You can avoid overloading teams, ensure that you have the necessary skills on hand, and avoid resource shortages.

   Quality Assurance: Risks related to the quality of the software, including technical challenges or issues with the development process, can be identified and addressed. This contributes to the delivery of a higher-quality product.

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

   Software Configuration Management SCM is crucial for ensuring project quality, as it helps maintain consistency, traceability, and reliability in a software project. Here's an explanation of the concept of SCM and its role in ensuring project quality:

   Concept of Software Configuration Management:

   Version Control: SCM provides version control mechanisms to keep track of different versions of source code, documentation, and other software assets.

   Configuration Items: In SCM, software artifacts are grouped into configuration items (CIs). These can include source code files, libraries, documentation, and other deliverables. Each CI is uniquely identified and managed as a separate entity.

   Change Management: SCM helps track and manage changes to CIs. It provides tools for recording, reviewing, approving, and implementing changes, whether they are bug fixes, new features, or enhancements.

   Baselining: Baselines are snapshots of the entire project or specific CIs at a particular point in time.

   Role in Ensuring Project Quality:

   Consistency and Reproducibility: SCM ensures that the project's code and artifacts are consistent and reproducible. This is vital for quality control because it helps identify issues caused by changes in the code or dependencies.

   Traceability: SCM enables traceability by maintaining a historical record of changes. This is critical for identifying the source of defects, understanding why certain changes were made, and ensuring that software meets requirements.

   Risk Management: SCM helps in identifying and managing risks related to changes in the software. It allows for controlled, well-documented changes, reducing the likelihood of introducing defects or breaking existing functionality.

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

   Formal Technical Reviews (FTR) are a systematic and structured approach to assessing and improving the quality and reliability of software during its development process. FTRs involve a group of individuals, often including developers, testers, and other stakeholders, coming together to review and discuss various aspects of the software, such as requirements, design, code, and documentation. FTRs contribute to ensuring software quality and reliability in several ways:

   Defect Detection and Correction:
   FTRs are effective in identifying defects, errors, and inconsistencies early in the development process. This includes issues in requirements, design, and code.

   Consistency and Compliance:
   FTRs ensure that the software adheres to standards, guidelines, and best practices established for the project. This helps in maintaining consistency and ensures that all team members follow agreed-upon processes.

   Risk Management:
   FTRs are instrumental in identifying and mitigating risks early in the project. By addressing potential issues during reviews, the project team can reduce the likelihood of critical problems arising later in the development process.

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

   Conducting a formal walkthrough for a software project is an essential part of the software development process. A formal walkthrough is a structured, collaborative review of a software artifact, such as requirements, design, code, or documentation. It involves key stakeholders coming together to evaluate the quality, correctness, and completeness of the artifact. Here's a step-by-step process for conducting a formal walkthrough:

   1. Planning:
   - Identify the specific artifact to be reviewed, such as requirements, design, or code.
   - Determine the scope and objectives of the review, including the goals and expected outcomes.

   2. Preparation:

   - Distribute the artifact to be reviewed to all team members well in advance of the scheduled walkthrough. This allows participants to familiarize themselves with the material.
   - Reviewers should prepare by reading the artifact, taking notes, and identifying potential issues or questions in advance.

3. Conducting the Walkthrough:

- Start the walkthrough meeting at the scheduled time with a clear agenda.
- The author of the artifact (e.g., the developer or author of requirements) typically presents it to the review team.
- The author provides context, explains design decisions, and addresses any known issues or areas of concern.

4. **Issue Identification and Documentation:**

- As issues are raised, they should be documented in a review report.
- Each issue should be clearly described, including its location in the artifact, a summary of the problem, and any relevant references.
- Issues may include defects, ambiguities, inconsistencies, missing information, or violations of standards or best practices.
- Reviewers and the author should work together to resolve issues and reach a consensus on necessary changes or clarifications.

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

Considering software reliability when analyzing potential risks in a project is crucial for several reasons:

User Trust and Satisfaction: Software reliability is directly tied to user trust and satisfaction. If the software is prone to frequent failures or crashes, users will lose confidence in it. Reliability issues can lead to dissatisfaction, loss of users, and damage to the software's reputation.

Cost Implications: Unreliable software can result in increased costs. Fixing defects and issues after the software is deployed can be significantly more expensive than addressing them during the development phase. Moreover, reliability issues may lead to costly downtime and maintenance.

Business Impact: Software is often a critical component of a business's operations. Unreliable software can disrupt business processes, cause delays, and result in financial losses. In some cases, it can even lead to legal or regulatory issues.

Safety and Security: In sectors such as healthcare, finance, and transportation, software reliability is essential for safety and security. Failures or vulnerabilities in software can have life-threatening or critical consequences. Analyzing reliability risks is vital to ensure safety and security.

User Experience: Unreliable software can result in a poor user experience, leading to frustration and reduced productivity for end-users. This can negatively impact the software's adoption and utilization.

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 detailsare 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)				