Software Quality Assurance (SQA)

CSC224

LECTURE 8

Learning Objectives:

- Understand the concept and importance of software quality assurance (SQA).
- Learn about quality attributes and how they are assessed in software.
- Explore SQA techniques and processes for ensuring software quality.
- Understand how to implement quality standards and metrics in software projects.

Intro

- **1. What is Software Quality Assurance?**
- **Definition:**
  Software Quality Assurance (SQA) is a set of activities designed to ensure that software products meet specified requirements and quality standards throughout the development lifecycle.
- **Key Goals of SQA:**
  - Prevent defects in software.
  - Ensure that the software meets customer expectations and requirements.
  - Reduce the cost of rework by identifying and addressing issues early.

2. Importance of Software Quality

- High-quality software:
- Reduces maintenance costs and operational failures.
- Enhances user satisfaction and trust.
- Complies with legal, regulatory, and contractual requirements.
- Supports scalability and reliability over the software lifecycle.

3. Software Quality Attributes

- Software quality is assessed based on **functional** and **non-functional** attributes:
- **A. Functional Quality**
  - Ensures that the software performs the intended tasks correctly and reliably.
  - Validated through functional testing.
- **B. Non-Functional Quality**
  - **Reliability:** Ability to perform consistently under specific conditions.
  - **Usability:** Ease of use for end-users.
  - **Performance:** Efficiency in resource utilization, such as response time and throughput.
  - **Maintainability:** Ease of modifying the software to address changes or issues.
  - **Security:** Protection against unauthorized access, data breaches, and vulnerabilities.

4. The SQA Process

SQA activities occur throughout the software development lifecycle (SDLC) and include:

- **A. Planning Phase**
  - Define quality objectives and standards.
  - Identify key performance indicators (KPIs) and metrics.
- **B. Design Phase**
  - Perform design reviews to ensure adherence to architectural principles and standards.
  - Use modeling tools to detect design flaws early.
- **C. Implementation Phase**
  - Conduct code reviews and static analysis.
  - Follow coding standards and best practices.

4. The SQA Process

SQA activities occur throughout the software development lifecycle (SDLC) and include:

- **D. Testing Phase**
  - Execute test plans to validate functionality, performance, and security.
  - Conduct regression and system-level testing.
- **E. Deployment and Maintenance Phase**
  - Monitor the software for defects and performance issues in production.
  - Apply patches and updates to address newly identified vulnerabilities.

5. SQA Techniques

Several techniques are employed in SQA to ensure software quality:

- **A. Reviews and Inspections**
  - **Code Reviews:** Identify defects in code during the implementation phase.
  - **Design Reviews:** Verify that design specifications align with requirements.
  - **Peer Reviews:** Engage team members to assess work products.
- **B. Testing**
  - Functional, non-functional, and regression testing to verify and validate software quality.
  - Automated testing for repetitive tasks.

5. SQA Techniques

- **C. Static Analysis**
  - Tools like SonarQube or Checkstyle analyze code without executing it, identifying issues such as code smells, dead code, and potential vulnerabilities.
- **D. Quality Audits**
  - Periodic reviews to ensure processes comply with established standards e.g.
    - ISO 9001
    - CMMI

6. SQA Standards and Models

Organizations adopt industry standards **to guide** and **certify quality assurance processes**:

- **ISO/IEC 25010:** Defines software quality attributes, including functionality, usability, and reliability.
- **CMMI (Capability Maturity Model Integration):**
  - A process improvement framework for software organizations.
  - Maturity levels: Initial, Managed, Defined, Quantitatively Managed, Optimizing.
- **IEEE Standards:** Provide guidelines for software testing, requirements, and documentation (e.g., IEEE 829 for test documentation).

7. Metrics in SQA

Metrics provide measurable indicators of software quality and process effectiveness:

- **A. Process Metrics**
  - Defect density: Defects per module or line of code.
  - Test coverage: Percentage of code or functionality tested.
- **B. Product Metrics**
  - Mean Time Between Failures (MTBF): Average time between system failures.
  - Response time: Time taken by the system to respond to user requests.
- **C. Project Metrics**
  - Schedule variance: Difference between planned and actual timelines.
  - Effort estimation: Accuracy of time and resource predictions.

8. Tools for Software Quality Assurance

- **Static Analysis Tools:** SonarQube, ESLint, Checkstyle.
- **Test Management Tools:** TestRail, Zephyr.
- **Defect Tracking Tools:** Jira, Bugzilla.
- **Performance Testing Tools:** Apache JMeter, LoadRunner.

9. Example: SQA in an Online Banking System

Scenario: Ensuring the quality of an online banking application.

- **Functional Quality:**
  - Validate that users can transfer funds and view transaction history.
- **Non-Functional Quality:**
  - Reliability:
    - Ensure the system operates 24/7 with 99.9% uptime.
  - Performance:
    - Response time for account balance inquiries should be < 2 seconds.
  - Security:
    - Validate secure login using two-factor authentication.
- **SQA Techniques:**
  - Conduct penetration testing to identify security vulnerabilities.
  - Use static analysis tools to check for coding standards compliance.

10. Key Takeaways

- SQA ensures software quality by combining process management, technical reviews, and testing.
- Quality attributes such as reliability, security, and performance must be addressed comprehensively.
- Metrics and tools are essential for measuring and improving software quality.
- Adherence to industry standards builds trust and ensures compliance.

Discussion Questions

- What are the advantages of adopting an industry standard like ISO/IEC 25010 for software quality?
- How can SQA activities reduce the cost of software maintenance?
- Why are both functional and non-functional quality attributes critical to software success?

Practical Activity

- **Objective:** Implement SQA techniques in a sample project.
- **Task:**
  - Choose a system (e.g., e-commerce or library management).
  - Define three functional and two non-functional quality attributes.
  - Perform a peer review of the codebase or design.
  - Use a static analysis tool (e.g., SonarQube or ESLint) to evaluate code quality.