QUALITY ASSURANCE (QA), QUALITY CONTROL (QC), AND TESTING

Mansi Thakur

Table of Contents

Quality is a critical component of software development, and QA, QC, and Testing play distinct yet interconnected roles in achieving it. Here's a detailed breakdown to help you understand each, their processes, purposes, and differences.

1. Quality Assurance (QA)

Quality Assurance is a **proactive process** that ensures software processes and practices are designed to produce high-quality outcomes. It focuses on **processes** rather than the actual product.

1.1 Purpose

- To prevent defects during the development lifecycle.
- To improve and standardize processes for consistent quality delivery.

1.2 Key Characteristics

1. **Process-Oriented:** Ensures adherence to defined processes (e.g., SDLC or Agile practices).
2. **Preventive Approach:** Identifies and mitigates risks early in the lifecycle.
3. **Documentation-Driven:** Involves guidelines, checklists, and quality standards.

1.3 Responsibilities in QA

- Establishing **quality policies and procedures**.
- Defining and maintaining **test strategies** and **plans**.
- Conducting **process audits** to ensure compliance.
- Organizing **training sessions** for teams to follow best practices.

1.4 Tools for QA

- **Jira** (process tracking)
- **Confluence** (documentation and collaboration)
- **TestLink** (test management)

1.5 Examples of QA Activities

- Establishing coding standards (e.g., **using SonarQube for code quality checks**).
- Process audits to check if testing aligns with standards.
- Risk assessments to avoid failures later in the lifecycle.

2. Quality Control (QC)

Quality Control is a **reactive process** that focuses on verifying the quality of the product after it has been developed. It identifies and rectifies defects in the software.

2.1 Purpose

- To detect defects in the product.
- To ensure the software meets the required specifications and standards.

2.2 Key Characteristics

1. **Product-Oriented:** Focuses on inspecting the actual deliverable.
2. **Corrective Approach:** Identifies defects and ensures they are fixed before release.
3. **Hands-On Testing:** Relies on manual and automated testing methods.

2.3 Responsibilities in QC

- Executing **test cases** to validate the product against requirements.
- Reporting and tracking **defects**.
- Conducting **peer reviews** of deliverables.
- Preparing **release notes** and certification reports.

2.4 Tools for QC

- **Selenium** (automation testing)
- **Postman** (API testing)
- **Jenkins** (CI/CD testing pipeline)

2.5 Examples of QC Activities

- Conducting **functional testing** to ensure a feature behaves as expected.
- Performing **load testing** to verify system performance under stress.
- Reviewing deliverables like UI designs or user manuals.

3. Testing

Testing is a subset of Quality Control that involves **executing software** to identify defects. It validates that the product works as intended and meets the requirements.

3.1 Purpose

- To evaluate functionality, performance, security, and usability.
- To uncover defects before the product reaches the end user.

3.2 Key Characteristics

1. **Execution-Oriented:** Focuses on running software in various conditions.
2. **Defect-Finding Approach:** Prioritizes detecting bugs in the code.
3. **Varied Techniques:** Includes both **manual** and **automated testing**.

3.3 Types of Testing

- **Functional Testing:** Validates features against requirements.
- **Non-Functional Testing:** Includes performance, security, and usability testing.
- **Regression Testing:** Ensures new changes haven't broken existing functionality.
- **Exploratory Testing:** Ad hoc testing without predefined scripts.

3.4 Testing Artifacts

- **Test Plans:** Detailed documentation of testing scope, objectives, and schedules.
- **Test Cases:** Specific steps to validate functionality.
- **Defect Reports:** Logs of identified bugs with details for resolution.

3.5 Tools for Testing

- **Selenium, Appium** (functional testing automation)
- **JMeter** (performance testing)
- **Burp Suite** (security testing)

3.6 Examples of Testing Activities

- Executing test scripts to check login functionality.
- Verifying API responses using Postman.
- Stress testing a payment gateway under high traffic.

4. Differences Between QA, QC, and Testing

Aspect	**Quality Assurance (QA)**	**Quality Control (QC)**	**Testing**
Focus	Processes and standards to prevent defects.	The product to detect defects.	Execution of the product to find bugs.
Proactive/Reactive	Proactive (prevention).	Reactive (correction).	Reactive (detection).
Scope	Broader: Encompasses QC and Testing.	Focused on verifying the quality of deliverables.	Focused on finding and fixing issues in code.
Involvement	Throughout the development lifecycle.	After product creation, during validation stages.	During the validation stages of QC.

Deliverables	Quality policies, test strategies, audit reports.	Inspection reports, defect logs.	Test cases, defect reports.
Example Activity	Defining coding standards for developers.	Reviewing design documents for correctness.	Testing a login feature for proper functioning.

5. How They Work Together

QA Oversees the Process:

- QA defines the **processes** and **standards** for development and testing.
- Example: Establishing coding guidelines, creating a test strategy.

QC Verifies the Product:

- QC ensures the processes outlined by QA are followed and the final product meets quality standards.
- Example: Peer-reviewing code for adherence to coding guidelines.

Testing Identifies Defects:

- Testing is the actual activity of running the product to identify bugs and inconsistencies.
- Example: Functional testing of a new feature before release.

6. How to Identify What Comes Under Whom

Activity	QA	QC	Testing
Defining Test Plans	✅ (QA responsibility	❌	❌
Executing Test Cases	❌	✅ (Testing is part of QC).	✅
Process Audits	✅	❌	❌
Regression Testing	❌	✅	✅
Code Reviews	✅ (If process-related).	✅ (If deliverable-related).	❌
Risk Identification	✅	❌	❌