tags: - cocubes - technical - qa - testing

# Software QA and Testing Tools

## QA, QC and testing

| Term | Main focus |
|---|---|
| Quality assurance | process-oriented defect prevention |
| Quality control | product-oriented quality checking |
| Testing | execute/evaluate software to find defects and build confidence |

Testing can reveal defects; it cannot prove their complete absence.

## Testing levels

| Level | Scope | Typical owner |
|---|---|---|
| Unit | smallest component/function | developer |
| Integration | interaction between components | developer/tester |
| System | complete integrated system | test team |
| Acceptance/UAT | business/user acceptance | customer/business |

## Testing types

| Type | Purpose |
|---|---|
| Functional | verify specified behaviour |
| Performance | response time, load, stress, scalability |
| Security | vulnerabilities and protection |
| Usability | ease of use |
| Compatibility | browsers/devices/OSs |
| Regression | check existing features after a change |
| Retesting | confirm a specific reported defect is fixed |
| Smoke | broad critical build check |

| Type | Purpose |
|---|---|
| Sanity | focused quick check after a small change |

## Static and dynamic testing

- Static testing: no program execution; reviews, inspections, walkthroughs, static analysis.
- Dynamic testing: execute the software.

## Black-box techniques

| Technique | Method |
|---|---|
| Equivalence partitioning | choose representative values from behaviourally equivalent groups |
| Boundary value analysis | test edges; for valid range 1–100, consider 0, 1, 100, 101 |
| Decision table | cover combinations of conditions and actions |
| State transition | test states, valid and invalid transitions |

## White-box techniques

Use code structure:

- statement coverage: execute each statement;
- branch/decision coverage: execute each decision outcome;
- path coverage: execute possible paths, often impractical for complex code.

## Defect lifecycle

Typical states:

New → Assigned → Open/In progress → Fixed → Retest → Closed.

Possible alternatives: Reopened, Deferred, Duplicate, Rejected, Cannot reproduce.

Severity = technical/business impact. Priority = urgency/order of fixing. A low-severity typo may have high priority in a public campaign; a severe rare issue may have a different priority depending on risk.

## Common tools: know the category, not every feature

| Category | Examples | Purpose |
|---|---|---|
| UI automation | Selenium, Cypress, Playwright | automate web UI checks |
| Mobile automation | Appium | automate Android/iOS tests |

| Category | Examples | Purpose |
|---|---|---|
| API testing | Postman, SoapUI | send/validate API requests |
| Performance | JMeter, LoadRunner | simulate load/measure response |
| Unit testing | JUnit, TestNG, pytest | automate component tests |
| Defect/project tracking | Jira, Bugzilla | record defects/work |
| Test management | TestRail, Zephyr | plan and track test cases |
| CI | Jenkins, GitHub Actions | run builds/tests automatically |

## Test case anatomy

A good test case has:

- identifier and title;
- preconditions;
- test data;
- steps;
- expected result;
- actual result and status.

[[03_Technical/Technical_Drills|Technical drills]] · [[03_Technical/Software_Development_Methodologies|SDLC]]