Software Testing

JD Kilgallin

CPSC:480

10/31/22

Photo: Jon Hall, Diversity in Tech
Known for: First Linux port, Founder and Executive Director of Linux International, Chairman of Linux Professional Institute, Author of "Linux for Dummies"

Quality Assurance

JD Kilgallin

CPSC:480

10/31/22

Pressman Ch 17

Notes

- Project 2 dates pushed slightly to allow more time for writing tests.
  - Checkpoint pushed 3 days to Wednesday, Nov 9.
  - Project due date pushed 1 day to Monday, Nov 14.
  - Early submission dates and team participation survey also pushed 1 day.
- Exercise 6 Wednesday; bring a laptop to write software tests.
- Keyfactor has a new posting for a Software Engineer role: see description at https://www.keyfactor.com/jobs/keyfactorinc/software-engineer-2/?gh_jid=4673843004. Email me a resume and a statement about whether or not I can discuss your grade to apply.

Recruiting Agencies

- I get a lot of questions about finding and applying for roles. My top piece of advice is almost always "work with a recruiting agency".
- Recruiting agencies are professional firms with expertise in the job market and connections to many hiring managers.
- Employers pay recruiters on commission to fill an opening with someone who stays at least 90 days (may be longer), so they really have an incentive to find you a good match.
- Another strategy is to look for contract and contract-to-hire roles too.
    - Contract: Temporarily hired for a set time or project with definite terms.
    - Contract-to-hire: Starts as a contract job, with an expectation the contractor will join full-time after the contract. Hiring managers may ask entry-level candidates with no experience to do this as a low-risk "probation" period.

Robert Half

- I (and others) joined Keyfactor through Robert Half – one of the largest global staffing firms with 345 locations worldwide.
- Contact: Dawson DiPietro
  - Robert Half tech recruiter in Cleveland
  - 216-738-8123
  - Dawson.Dipietro@roberthalf.com
- Create a profile on https://www.roberthalf.com/jobs, especially if applying through email.
- Dawson's office only hires for NEO jobs, but you can apply elsewhere online and work with him for a referral to another Robert Half agent.
- Dawson focuses on contract-to-hire roles, but works in conjunction with another local representative for full-time positions as well.

Learning objectives

- Quality assurance concepts
- Software testing concepts
- Types of testing
- Writing and running tests

What is Software Quality Assurance?

- Quality Assurance: "The maintenance of a desired level of quality in a service or product, especially by means of attention to every stage of the process of delivery or production."
- A process that assesses all software engineering activities and work items for adherence to the team's defined software quality standards.
- An activity that occurs at each step of the software development lifecycle mapping the discipline of quality assurance to the software engineering process
- A quality assurance system is a set of organizational structures, responsibilities, resources, processes and procedures to ensure products satisfy stakeholder expectations.

Terminology

- Failure – An occasion in which a user of released product encounters a software malfunction preventing a requirement from being fulfilled.
- Mean Time Between Failures (MTBF) – Average interval for the time to recover from one failure ("Mean Time to Recover") and operate the software until the next failure ("Mean Time *To* Failure").
- Reliability – Ability to use the software without failure. MTBF=MTTR+MTTF is one measure. Can invert and calculate average failures per *n* hours of use
- Availability = MTTF/MTBF – Percentage of time software can be used according to requirements. ("Downtime" = MTTR/MTBF)
- Verification – Ensure product meets design (Building the product right)
- Validation – Ensure product meets requirements (Building the right product)

Pillars of Software Quality Assurance

- Process – A well-defined, documented, repeatable QA process ensures quality is consistent and details don't fall through the cracks.
- Reviews and Tests – Regular quality assurance tasks like code reviews and software testing are the best tools for assessing software quality.
- Software engineering practices – A rigorous process with ample planning and design improves the quality of the final product.
- Change Control – Management of all software engineering artifacts and changes made to them.
- Measurement and Reporting – Internal mechanisms to assess and quantify quality give visibility into quality of product by the team.
- Audits and Compliance – Conformance to external quality standards.

Who does Quality Assurance?

- Project management coordinates activities and tracks status of QA work and reported bugs.
- Developers write unit tests in conjunction with new code/bug fixes.
- QA Engineers monitor test results and write additional tests.
- Entire team keeps quality as a priority throughout entire process.
- Alpha/beta testing and end users identify failures ("Everybody has a test environment, some are lucky enough to have a separate one for production")

QA for Requirements Engineering

- Correctness, completeness, and consistency of software requirements specifications should be assessed both before moving on AND throughout the remainder of the development cycle.
- Some attributes include:
  - Completeness – Number of empty/stub sections, "TBD", or "TODO".
  - Ambiguity – Number of terms that are too imprecise ("many", "easy")
  - Volatility – Number of changes per requirement.
  - Traceability – Number of requirements that can't be traced from design document, code commits, or test cases.
  - Clarity – Number of requirements models; number of pages per model

QA for Product Design

- Product design should correspond to requirements specification and scope of project plan. Suitability should be assessed both before moving on AND throughout the remainder of the development cycle.
- Some attributes include:
  - Completeness – Presence of model elements for all requirements
  - Scope – All design elements traceable to requirements.
  - Complexity – Number of design elements, classes, and/or functions
  - Architecture – Number of interfaces between modules and interactions between them.
  - UI – Average number of clicks/keystrokes to complete a user story or other task specified in a requirement

QA for Software Construction

- All code created for the product should be assessed for correctness of behavior, presence of defects, and adherence to requirements specification, product design, and applicable coding standards.
- The most QA work occurs here, especially through code reviews and manual + automated testing.
- Some attributes include:
  - Code quality metrics - cyclomatic complexity, scattering, tangling, duplicated code, amount of codebase in compliance with naming conventions and style guidelines, number of reviewers per code commit, etc.
  - Test coverage (requirements w corresponding tests), code coverage (paths covered by tests), documentation.
  - Bugs found after release; user satisfaction reports; sales and reviews.

What is software testing?

- Examining the artifacts and behavior of a software product to verify and validate functionality.
- Process to gain confidence that the software product works as intended. The process aims to identify and correct or document defects in software functionality.
- A collection of procedures to verify that attributes of a product meet expectations.

Why is testing important?

- It's the best tool to ensure software quality over the lifetime of the product.
- Adequate testing reduces the amount of engineering work and cost related to debugging, troubleshooting, and re-engineering.
- Testing reduces costs of code changes by providing visibility into the effects of a change.
- Testing improves bottom-line profit by increasing revenues based on software quality while reducing losses related to failures.
- Inadequate testing leads to real physical and economic harm as a result of software defects.

Test Cases

- A test case is a set of initial conditions and expected results with steps that can be executed to confirm that the actual results match expectations.
- Should be pass/fail, have a defined schedule or events when it should be run, and record results from each run.
- A test case may be manual (tester executes the steps and checks the results) or automated (test code is written to perform the steps and programmatically compare results).
- May be a unit test, integration test, end-to-end test, or other type.
- May be white-box (aware of source code) or black-box (testing behavior alone without considering implementation).

Test Case Lifecycle

- Requirements and design are analyzed for functionality that should be tested.
- Test plan is developed for test cases to cover requirements – objectives, resources, scheduling, budget, environments, test case design, execution and results monitoring strategy.
- Test cases are written according to plan.
- Test environment is set up to run tests.
- Tests are executed as needed to verify software functionality.
- Results are monitored, coverage is checked, and bugs are reported.
- Test cases that no longer apply or that are discovered not to work correctly are removed.

Test Plans

- It is important to consider what test cases should be written when planning and designing a product.
- Requirements may translate directly into a set of tests, and being able to test them demonstrates you've met the requirements.
- Testability is an important architectural consideration, as the ability to demonstrate progress on requirements, and to catch bugs and regressions, depends on being able to effectively test the program.
- Larger teams will have QA Engineers work on test plans in conjunction with developers during software design.

Unit Tests

- White-box test cases that test one code path of one method.
- Calls to local helper functions are okay, but calls to other modules should be *mocked*. A mock component implements the interface accessed by the code under test and returns static content or results with very basic processing.
- Important to cover edge cases and error-handling paths.
- Cyclomatic complexity gives a lower bound for number of unit tests that cover all of the code.
- Usually written by a developer when checking in new code or bug fix; sometimes even *first* ("Test-Driven Development" (TDD) advocates for writing a test that fails, then fixing the code and watching it switch to passing, to confirm the test distinguishes correct/incorrect behavior).

Coverage

- Code coverage measures the amount of code that is executed by at least one test case. Usually measured as a percentage of statements (lines) tested over total number of statements, but may also calculate branch coverage, function coverage, and others.
- Test coverage measures the percentage of requirements that can be traced from at least one test case.
- Both are valuable, and the distinction is important. Tests cover code and requirements cover tests.
- Code coverage can be computed automatically very easily by a testing framework; test coverage is a more manual process.
- Code coverage is generally the responsibility of the developers through unit tests, while test coverage is the responsibility of QA.

Other types of test

- Integration test – Testing interactions between components/larger blocks of code.
- Scenario or end-to-end test – Complete user story or use case
- Stress test or load test – Measure performance under heavy (simulated) use, atypical usage patterns, edge cases, high demands, or significant constraints.
- Security test – Verify the product is resilient to a particular vulnerability or attack.
- Fuzz test – Randomized input to test edge cases and security.
- UI test – Verify appearance of user interface and input/output meets design.
- Usability test – Focus group or user feedback measuring ease of use.
- A/B test – Usability test of two different versions of a feature.
- Installation/upgrade – Verify that deployment succeeds in various conditions.
- Smoke test – Verify that the product launches and has basic functionality (usually done as a post-build or checkin step).

QA for QA process itself

- Efficiency and impact of the QA process itself should be assessed.
- Attributes include:
  - Completion time – Amount of time taken to complete QA activities vs budgeted time.
  - Test effectiveness – Bugs uncovered by test cases.
  - Bug tracking – Known defects are documented, all bugs are triaged and closed as appropriate.
  - Verification & Validation – Percent of requirements or designed elements implemented within project.
  - MTBF and availability
  - Audits and standards compliance reports (ISO 9000, for example)

References

- Paw Prints: Writings of the maddog. Jon Hall. Apr 2009. Linux Magazine.
- Search Jobs. 2022. Robert Half Talent Solutions.
- What is Software Quality Assurance: A Guide for Beginners. Oct 2022. Softwaretestinghelp.
- Software Testing. Wikipedia.
- Test Coverage in Software Testing. Vineet Nanda. Sept 2021. Tutorialspoint.
- Software Testing Tutorials. Swati Tawde. Sept 2019. Educba.

- *Bring a laptop to next lecture*