1. **What is meant by software Testing?**

- Software testing is a matured process of verification or validation of software against the features, requirements or specifications, which are both functional as well as non-functional. It involves creating test plans, test specifications, test code development, execution of tests and checking the documentation. Also making sure that the product code changes doesn't cause the regressions, which means failure of earlier working features.

1. **What is meant by software quality?**

- Software quality means the expected level of meeting the specifications or requirements, which are both functional as well as non-functional. The different levels of low, medium and high represent overall quality. In general, high-quality products will have higher customer satisfaction and recognition in the same line of low-level products. Software testing contributes to determining or assess the product quality.

1. **What is a software bug or defect?**

- Bug is nothing but an unexpected or deviation from expected behavior in the product. It could be detected during either positive or negative testing. That means either no errors or exceptions or missing things in the positive cases or an expected errors during the negative cases where the user should receive it. Bugs are recorded with all the relevant information to re-produce the observation or test scenario in a bug software like bugzilla tool.

1. **Why do you need to Test the software? What happens if you don't test separately than the developer?**

- In general, the product customer is important for any of the product producer. Higher quality of the customer satisfaction better to the company. It would be too expensive if customer experiences the bugs both in-terms of fixing those bugs and credibility. The quality assessment can be done through software testing. Thereby, every product has to be tested and assessed for quality before delivering to the customer. Typically, the developer testing is minimal and limits to the unit code. Also covered only few peripheral units but not at the customer focused functional, end to end integration at a system level. So, it has to be tested separately than the developers and typically done by software quality group.

1. **How Quality Engineering different from Quality Assurance or simply execution?**

- Quality Engineering is to apply engineering principles such as consistency and repeatability in the software development processes and whereas Quality Assurance is more of monitoring activity of processes. Many times, these are interchangeable, but the key is to apply consistency and repeatability in the testing than simply running a scenario once.

1. **What is the need for having a separate Quality Engineering Team?**

- Developing code and verifying by the same coder might make some of the bugs under cracks and need a team who would use the product as if like customers. That's why QE engineers are like first customers.

1. **Can we eliminate all bugs in a given product through extensive testing? What should we do?**

- Because of the complexity of software programming algorithms and code, it is impossible to eliminate all the bugs in a given product. It would need unlimited resources, time and even then no guarantee for zero bugs. Thereby, typically quality is assessed as per planned criteria and customers can view the quality at different levels such as low, medium and high.

1. **What are functional requirements and nonfunctional requirements?**

- The functional requirements are the use cases relevant to the end user visible features of the product. For example, bank transactions in an online banking site.
- Nonfunctional requirements are the ones that needed to have the system/soft-ware functioning correctly. The examples are security, performance, reliability, availability, and scaling, etc.

1. **What are the different types of Testing? Can you list some?**

Some of the testing types are as below:

- Unit testing
- Functional testing
- Integration/System testing
- Security testing
- Performance testing
- Stress and Longevity testing
- High Availability testing
- Exploratory testing
- Installer testing
- Regression testing

1. **What is meant by white box testing? List down some testing types for white box testing. Who does this type of testing?**

- White box or unit testing is nothing but testing by looking at the actual or product code and focus on the units of the code to verify for basic code health.
- It is also called Unit testing.
- Typically performed by product developers.
- Example test frameworks used are JUnit, TestNG for Java and nUnit family frameworks.

1. **What is meant by black box testing? List down some of the testing types for Black box testing. Who does this type of Testing?**

- Black box testing is nothing but testing of the product without looking at the developed code and focuses on the customer functional use cases or scenarios.
- It is also called functional testing.
- Typically performed by QA team
- Example test frameworks used are TestNG, Selenium, Load Runner, etc.

1. **What is Unit Testing? What is its relation to whitebox and black box testing.**

- Unit testing is units of the code to verify the basic code functioning and done by developers while developing the product code. It is related to White box testing.

1. **What is Functional Testing? What is its relation to whitebox and black box testing? Who does this testing?**

- Functional testing is the testing with more focused on the functional specification or customer use cases and features testing without looking at the product code itself.
- It is related to black box testing because the test scenarios are viewed simply as whole without going through the code while testing.
- Typically, functional testing is performed by QA team.

1. **What is Security Testing? Is it functional or nonfunctional requirement testing?**

- Security testing is focusing on the security aspects of the product including security penetration testing.
- Security is a nonfunctional requirement of a product.

1. **What is Regression Testing? When and how often do you run these tests?**

- Regression testing is the execution of already created test suite or executing manual scenarios again. The goal is to make sure that already-existing features continuing to work while code changes are happening. Typically, these tests executed on hourly, daily, weekly or bi-weekly on different platforms and test matrix based on the product release cycles. Also performed regression testing at the time of build creation and developer check-ins.

1. **What is Performance Testing? Who does this testing - QA or separate team?**

- Performance testing is the measurement of speed (time per transaction or request process time) and throughput (number of users per second served) under adequate infrastructure that customers use.
- Also teams performance comparison analysis with competitive products.
- Typically, performed by a separate team (Performance team) than QA team in a larger organization.

1.What is a Test Plan? Describe briefly the contents of a Test Plan. Have you ever written a Test plan?

- The test plan is a testing artifact in the planning process to plan the testing with details like scoped, non-scoped requirements to test, test strategy, resources, matrix and schedule (estimates) created by the QA team.
- At a high level, there are two types of plans based on the scope on the project/module level.
- Master test plan, which is for the entire product/project.
- Module test plan, which is for the individual modules.
- Master test plan is based on Market requirements document (MRD)/ Business requirements document (BRD)/product requirements document (PRD)/Architecture & Design document.
- Module test plan is similar but focused on a module/component under test based on the Master test plan and component functional specification.
- A typical module test plan content is shown as below. You can take this as a template for the test plan even though some variations might exist in the organizations.
- Introduction or Overview
- Objectives
- Features Covered (in Scope)
- Features NOT Covered (relevant but not in scope)
- Test Methodology
- Entry Criteria
- Exit Criteria
- Test Certification Matrix
- Schedule, Responsibilities, Deliverables
- Risks and Mitigations
- Reviewers and Approvals
- References
- Revision History

- Yes. In the past, created both master and module test plans.

1. What is a Test design and specification? Describe briefly contents of Test Spec or Test Design spec. Have you ever written a Test Spec?

- Test design process is more of thinking of the test scenarios and their priority. The Test Specification (TS or TDS) is documenting those scenarios as test cases before actually testing as either manually or automating the tests.
- A typical test specification content is as below and can be taken as a template to create the document.
- Introduction
- Prioritized list of Test Scenarios for each requirement in the functional spec to be covered pointing to the below details
- Details
- Test Environment Setup
- Reviewers and Approvals
- References
- Revision History

1. **What is meant by a Test scenario? Where do you put test scenarios?**

- The Test scenario is nothing but a sequence of steps to be performed with a goal of verifying a user story or use case or requirement. These scenarios are going to become the test cases or tests and grouped as test suites.
- Test scenarios are going to be created during the test design process and will be documented in the test specification.

1. **Can you describe the bug life cycle? What are different states of a bug?**

- The bug life cycle is the process managing the events starting from creating a new bug until it is closed.
- Below are the typical bug lifecycle states but might have different named states based on the bug tool:
    - New (state) → sets when filed a bug by QA or re-opened.
    - Assign → sets when assigned to a developer by QA/QA manager/ Dev Manager/Developer.
    - Evaluate and in-progress → sets by the developer during evaluation.
    - Fixed → sets by the developer when code change happened for bug fix.
    - Duplicate bug → sets by the developer if already that bug exists.
    - Not a bug → sets by the developer if it is not a bug.
    - Not reproducible → sets by the developer if it is not reproduced.
    - Verified → sets by QA after verified as OK.
    - Closed → by QA/Filer after verification.

1. **What is meant by bug Priority? Who sets the priority of a bug? Can we change it after filed the bug?**

- Bug priority is nothing but an indication of how important or urgent to get the bug fix.
- Typically divided into five levels: P1 (highest), P2, P3, P4, P5 (lowest).
    - P1: Need immediate fix (tests blocking) within 24hrs
    - P2: Need by next build (major functionality is not working)
    - P3: It is ok to wait but fix before release
    - P4/P5: It is not mandatory, it is ok to fix or not in the current release.
- The bug filer would determine the priority and set while filing the bug. The thumb rule is to set the same priority as severity while filing the bugs. Example: P1/S1, P2/S2, P3/S3, P4/S4, P5/S5.
- Yes. Anyone can change the bug priority and bug tools would allow that. Typically, changed after the conversations with stakeholders like release managers and product managers. Example: QA can file a bug, but Dev manager or product manager can change the priority.

1. **What is meant by downgrading a bug and upgrading a bug?**

- Downgrading the bug means lowering the priority of a bug. That is the change from higher to lower the priority of the bug (in bug tool).
- Example: P1 → P2/P3/P4/P5, P1/P2 → P3/P4/P5, P3 → P4/P5
- Upgrading the bug means increasing the priority of a bug. That is the change from lower to higher the priority of the bug (in the bug tool).
- Example: P2/P3/P4/P5 → P1, P3/P4/P5 → P2/P1, P4/P5 → P3/P2/P1

1. **What is meant by bug Severity? Who sets the severity of a bug? Can we change it after filed the bug?**

- Severity (S) is the indication of bug's effect and impact on the customers. For any new bug, this is the first thing to be determined while filing the bug.
- Typically, severity is divided into five levels and are: S1 (Highest), S2, S3, S4, S5 (Lowest)
- S1: means that the feature can't be tested or blocking further testing
- S2: majority features are not working, and many tests are failing
- S3: some cases are not working
- S4, S5: very corner cases and not impacting the major functionality
- The bug filer would determine the severity and set while filing the bug.
- No. Typically, once set the severity in the bug, then no changes can be done. Typically, bug tools will not allow changing the severity.

1.**How do you file a bug? What details would be added? Give an example.**

- The bugs would be filed using the organization's bug tool and follow the bug guidelines. Adjust the above terms based on the tool being used.
- The below information is needed before filing any bug.
- Which product id, component id/name
- Which scenario or test
- What is the result/output
- What are its priority and severity
- More details like console output or screenshot etc. to add in description
- Typically, bug tools are web tools and filling the forms and later editing as needed. Below is the example snapshot to create a new bug.

Bug state: new/duplicate
Product:___
Product version:__
Component/module:___
Sub Component/sub module:___
Priority:__
Severity:__
OS/HW/DB/JDK:___

Subject: <u>Login failure - invalid password error even with correct password</u>
Description: (text area with limited size, say 3000 chars)

It has been observed that there is a login failure on Today's latest build b20 dated 12/22 even when entered the right password. The following error noticed on the UI screen.

......

Also, the below error message/stack trace/exception noticed in the server log file, server.log

......

Steps to reproduce:
1. Go to URL: http://....
2. Click login button
3. Enter user name: user1
4. Enter password: password1
5. Click Go button

Now you can notice the error on the screen. Double checked the user name in the database, and it worked fine during user creation.

Also attached the screenshots. You can also find the entire log file at /home/jmunta/error_logs/bug12221_server.log

Attach: upload the UI screenshot.
upload the log file.

Click submit and see the successful message as below and get email notifications to the filer and default assignee. The bug number is used in the conversations with stakeholders.

Bug <number> created successfully.

1. **What is Stress or load testing? What is longevity testing? How does stress and longevity testing are related? What kinds of bugs can be found? Who does this testing - QA or Dev or any other team?**

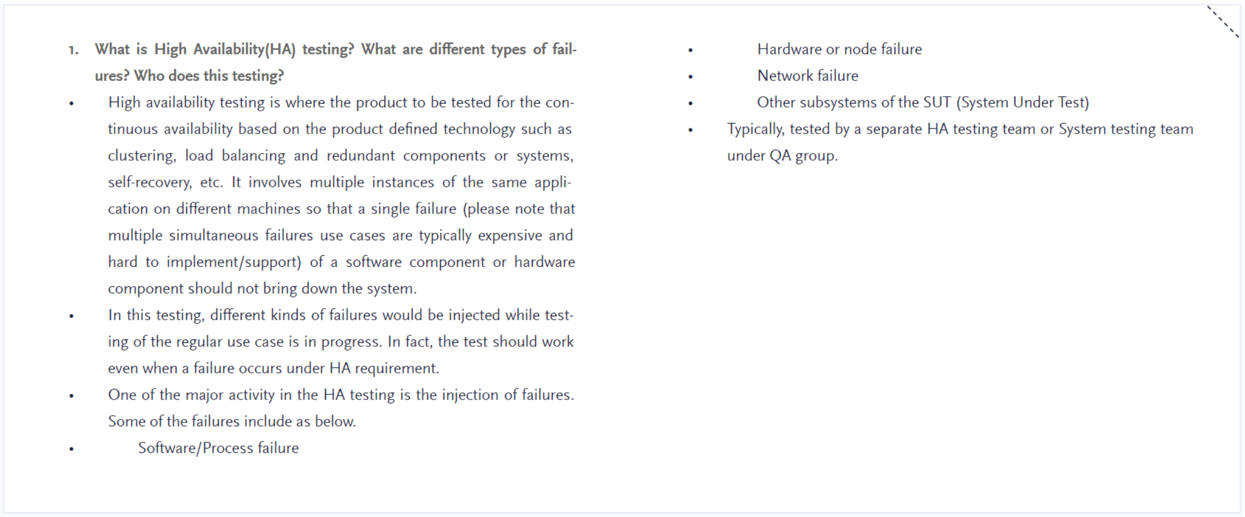
- The stress or load is the number of concurrent or simultaneous users or requests (say 100 users/second) sending to the system under test. The stress or load testing is to find the limits of those concurrent users/requests without breaking the system or within the acceptable behavior under the load.
- The longevity is the period, and longevity testing is the where continuously running of the tests for 24x7 a week or 3 days or a day under the load to make sure system doesn't break when it is serving the requests for a long time.
- Typically, stress and longevity testing are done together. It means that load test (say 100 users/second sending to the system under test) is running continuously (no stoppage) for seven days to test the overall reliability of the system under test.
- The bugs in the stress/longevity testing are difficult to find and difficult to reproduce and fix. Below of few types of issues that can be seen.
- Resource leaks such as memory leaks and file leaks
- Deadlocks
- Process hangs
- Typically, tested by a separate Stress testing team under QA group or by a System Reliability team.
- In general, the tests are complex to reproduce and verify because of time and complexity.

1. **What is System testing? Who does this testing? What are the key elements in system testing?**

- The system testing is the end-to-end use cases testing with all the infrastructure and environment in place for the system under test to be tested.
- The system testing is initiated when most of the product under test is available, and it is also called integration testing where all the modules integrated into a product. In fact, system testing is a big umbrella covering many test areas.
- Most of the time, big test apps and the infrastructure related tests would be considered here. Sometimes the system testing might be covered only the Reliability testing (Longevity and Stress).
- Under the system testing, some of the major areas included are Security Testing, Performance Testing, Reliability Testing (Longevity and Stress or Load testing), Installer Testing, Interoperability Testing, Compatibility Testing, Upgrade Testing and User experience Testing.

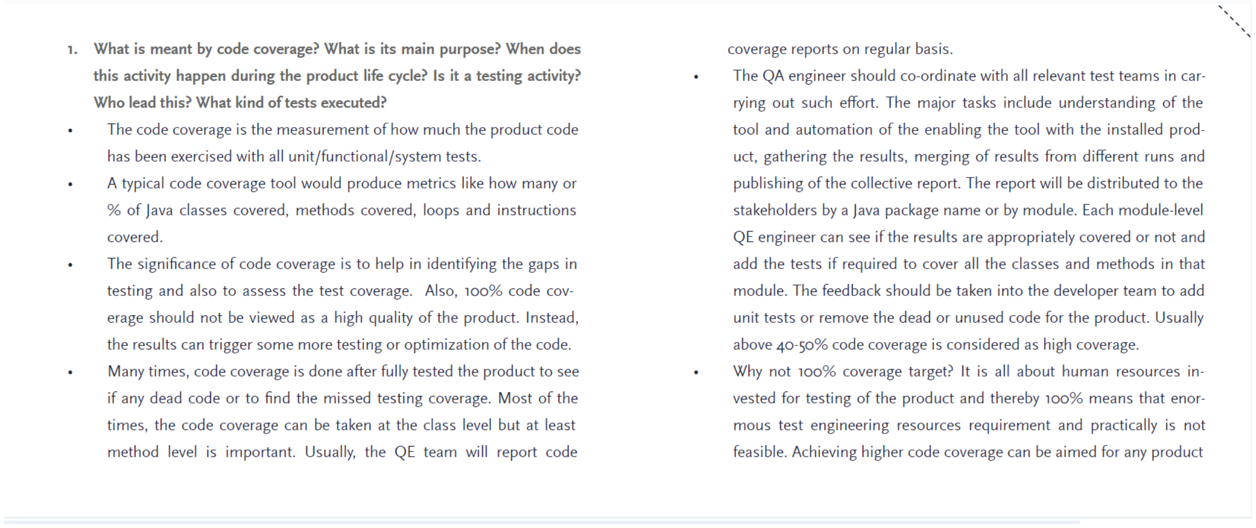
1. **What is a topology? How does it relate to testing?**

- Topology is the reference architecture of the product deployment with reference to customer scenario. These topologies typically documented in the deployment user guide like enterprise deployment guide (EDG). Customers would use these topologies as a reference at their sites.
- The common topologies and reference configurations would be used in the system testing to mimic the customer use cases with big apps testing, stress/longevity testing, performance testing, etc.



1. **What is High Availability(HA) testing? What are different types of failures? Who does this testing?**

- High availability testing is where the product to be tested for the continuous availability based on the product defined technology such as clustering, load balancing and redundant components or systems, self-recovery, etc. It involves multiple instances of the same application on different machines so that a single failure (please note that multiple simultaneous failures use cases are typically expensive and hard to implement/support) of a software component or hardware component should not bring down the system.
- In this testing, different kinds of failures would be injected while testing of the regular use case is in progress. In fact, the test should work even when a failure occurs under HA requirement.
- One of the major activity in the HA testing is the injection of failures. Some of the failures include as below.
- Software/Process failure
- Hardware or node failure
- Network failure
- Other subsystems of the SUT (System Under Test)
- Typically, tested by a separate HA testing team or System testing team under QA group.



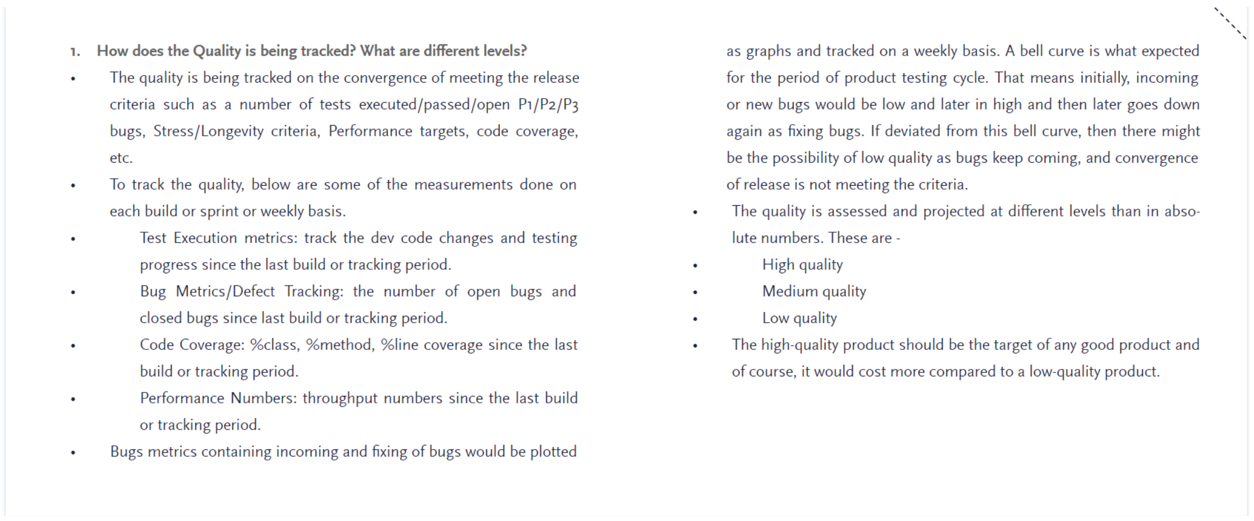
1. **What is meant by code coverage? What is its main purpose? When does this activity happen during the product life cycle? Is it a testing activity? Who lead this? What kind of tests executed?**

- The code coverage is the measurement of how much the product code has been exercised with all unit/functional/system tests.
- A typical code coverage tool would produce metrics like how many or % of Java classes covered, methods covered, loops and instructions covered.
- The significance of code coverage is to help in identifying the gaps in testing and also to assess the test coverage. Also, 100% code coverage should not be viewed as a high quality of the product. Instead, the results can trigger some more testing or optimization of the code.
- Many times, code coverage is done after fully tested the product to see if any dead code or to find the missed testing coverage. Most of the times, the code coverage can be taken at the class level but at least method level is important. Usually, the QE team will report code coverage reports on regular basis.
- The QA engineer should co-ordinate with all relevant test teams in carrying out such effort. The major tasks include understanding of the tool and automation of the enabling the tool with the installed product, gathering the results, merging of results from different runs and publishing of the collective report. The report will be distributed to the stakeholders by a Java package name or by module. Each module-level QE engineer can see if the results are appropriately covered or not and add the tests if required to cover all the classes and methods in that module. The feedback should be taken into the developer team to add unit tests or remove the dead or unused code for the product. Usually above 40-50% code coverage is considered as high coverage.
- Why not 100% coverage target? It is all about human resources invested for testing of the product and thereby 100% means that enormous test engineering resources requirement and practically is not feasible. Achieving higher code coverage can be aimed for any product

as part of the quality testing of the product.

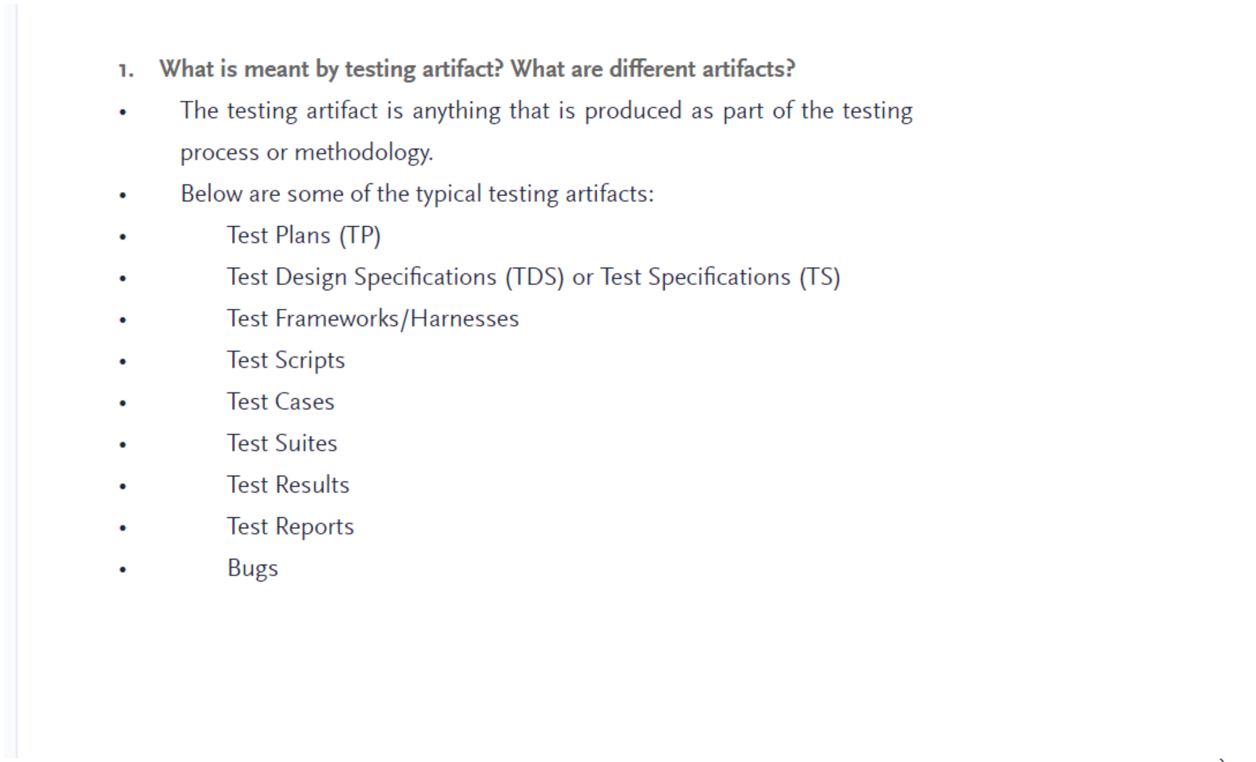
1. **What are different code coverage tools for Java?**

- Some of the free and open source code coverage tools are
- JaCoCo (now a days more popular and originally developed from Emma)
- JCov
- Emma
- Cobertura
- Clover
- PIT



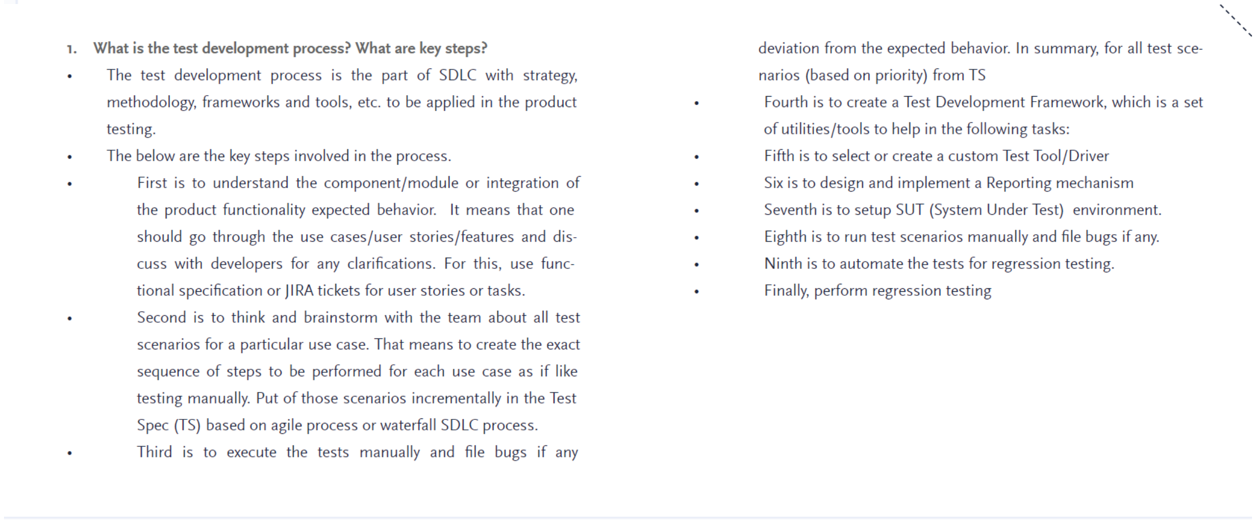
1. **How does the Quality is being tracked? What are different levels?**

- The quality is being tracked on the convergence of meeting the release criteria such as a number of tests executed/passed/open P1/P2/P3 bugs, Stress/Longevity criteria, Performance targets, code coverage, etc.
- To track the quality, below are some of the measurements done on each build or sprint or weekly basis.
- Test Execution metrics: track the dev code changes and testing progress since the last build or tracking period.
- Bug Metrics/Defect Tracking: the number of open bugs and closed bugs since last build or tracking period.
- Code Coverage: %class, %method, %line coverage since the last build or tracking period.
- Performance Numbers: throughput numbers since the last build or tracking period.
- Bugs metrics containing incoming and fixing of bugs would be plotted as graphs and tracked on a weekly basis. A bell curve is what expected for the period of product testing cycle. That means initially, incoming or new bugs would be low and later in high and then later goes down again as fixing bugs. If deviated from this bell curve, then there might be the possibility of low quality as bugs keep coming, and convergence of release is not meeting the criteria.
- The quality is assessed and projected at different levels than in absolute numbers. These are -
- High quality
- Medium quality
- Low quality
- The high-quality product should be the target of any good product and of course, it would cost more compared to a low-quality product.



1. **What is meant by testing artifact? What are different artifacts?**

- The testing artifact is anything that is produced as part of the testing process or methodology.
- Below are some of the typical testing artifacts:
- Test Plans (TP)
- Test Design Specifications (TDS) or Test Specifications (TS)
- Test Frameworks/Harnesses
- Test Scripts
- Test Cases
- Test Suites
- Test Results
- Test Reports
- Bugs



1. **What is the test development process? What are key steps?**

- The test development process is the part of SDLC with strategy, methodology, frameworks and tools, etc. to be applied in the product testing.
- The below are the key steps involved in the process.
- First is to understand the component/module or integration of the product functionality expected behavior. It means that one should go through the use cases/user stories/features and discuss with developers for any clarifications. For this, use functional specification or JIRA tickets for user stories or tasks.
- Second is to think and brainstorm with the team about all test scenarios for a particular use case. That means to create the exact sequence of steps to be performed for each use case as if like testing manually. Put of those scenarios incrementally in the Test Spec (TS) based on agile process or waterfall SDLC process.
- Third is to execute the tests manually and file bugs if any deviation from the expected behavior. In summary, for all test scenarios (based on priority) from TS
- Fourth is to create a Test Development Framework, which is a set of utilities/tools to help in the following tasks:
- Fifth is to select or create a custom Test Tool/Driver
- Six is to design and implement a Reporting mechanism
- Seventh is to setup SUT (System Under Test) environment.
- Eighth is to run test scenarios manually and file bugs if any.
- Ninth is to automate the tests for regression testing.
- Finally, perform regression testing

1. **What is the difference between manual testing and automated testing?**

- Both manual and automated testing would need in understanding the use cases/user stories.
- Manual testing process
- It is a simple process of manually testing of scenarios by executing interactively (like performing clicks & form submissions) and making decisions for pass or fail.
- Required to understand the basic user interfaces and has less learning curve.
- It is the basis for initial testing of use cases and file bugs.
- A lot of time and human resources cost involved in performing each build regression testing by repeating the tests.
- Hard to execute and scale if many test scenarios existed
- Potential to have inconsistent results and human errors
- General test steps
- Automated Testing process
- It is a process of testing the scenarios using automated code (using single command or scheduled process) with pass/fails reports generated.
- It is hard to create automated tests and requires high learning curve compared to manual testing.
- More skills and knowledge is required to automated tests. Sometimes hard to automate or complex to automate some cases.
- Easy to run and scale for large number of tests
- Less cost and effort in performing build to build regression testing.
- Less error prone once properly automated.
- Different test areas

1. **What is the test execution process? What do you need to focus?**

- The test execution is the process of executing the tests/test scenarios and file bugs or verifies the bugs on each product build. It is called "test cycle".
- First time, execute the test scenarios during test development process.
- Perform regression testing involves the below actions.
- Please note that avoid inconsistency in test results so that one can rely on the results. This can be achieved by repeating the tests many times with negative cases like software processes are down or network down or connection to the db/remote systems down etc. and fix the tests.
- Archive Results and Reports for audit purpose in future.
- A typical execution process flow is as below with multiple stages of testing:
- Gate 1 - Dev Gate: Run sanity/quick tests locally when development code is ready for integration. If any code failures, then analyze and fix or else if any test failures, then fix the tests. In any case, until all tests passed, dev code should not be integrated into the repository. After quick tests passed then, development code integration can happen.
- Gate 2 - Release Gate: Execute Smoke tests on the integrated build using CI and typically executed by RE (Release Engineer). If any failures in the smoke tests, then build should not be promoted for any further testing. Fix code or test failures and rebuild the integrated code and release the build.
- Gate 3 - QA functional Gate: Execute all BAT (Basic Acceptance Tests) to on core platforms. If BAT tests passed, then carry out the full test suite on all core platforms and BAT on other secondary platforms.
- Gate 4 - System Gate (Stress/Longevity/HA/Performance): Once the BAT tests passed and gave thumbs up by QA team, then System testing teams can pick the build to continue on their testing including Performance testing.
- There might be variations on the names of the tests categories like used in the above such as Quick or Sanity, Smoke Tests or BAT tests, SRG (Short Run Group), LRG (Large Run Group), Large suite, Short suite, etc.
- For each build cycle, the above gates would be opened in the build testing process.

1. **What is smoke testing? When do you run? Who run these tests?**

- The smoke tests are of a short number of P1 use cases to check the basic functionality of the integrated product is ok or not. The purpose is to have this gate in promoting the good builds to avoid the waste of time by other teams when basic functionality is not working.
- Typically, these tests executed when a build is being created (integrating of code) and also dev check-ins can use smoke as gate keeping on the regressions from the code changes.
- These tests are created by QA team by adding selected tests from each module or core integrated test scenarios.

1. **What is Basic Acceptance Testing (BAT)? Who and when do you execute BAT tests?**

- BAT tests are the functional test scenarios covering the customer acceptable use cases. These are used to verify the code before executing Full test suites and System/Integration testing.
- BAT tests also executed as per test execution matrix (smart test matrix) to cover some of the secondary platforms to save the execution time with appropriate test coverage.
- These tests also used by lower stack components as a basic check to make sure no regressions introduced for the upper stack components. For example, App Server BAT tests given to JDK QA for validating different JDK version changes.
- BAT tests are developed by the QA team and are a subset of the whole functional test suite.

1. **How do you sign-off from QA? What do you do?**

- The below simple checklist can be done followed to signoff from QA on the testing before releasing the product.
- QA should make sure all the committed/planned tasks as in the Test Plans of all modules have been completed. The checklist should include all bugs verification, no P1/P2/P3 open bugs, 100% tests execution and expected pass rate, code coverage measurement with targeted % of numbers etc.
- The test engineers should send the email to Dev/QA lead and Dev/QA manager with signoff message or update the dashboards/wiki on the release status update.
- Finally, QA representative - lead/manager should send or update the signoff status from QA to Release Manager and all project stakeholders.

← End: Skill#2. Software Quality Concepts ←

Good!
Keep going and never give up!!
Please re-read again for more clarity and feel free to contact for help!!!

→ Start: Skill#3. Object Oriented Programming & System (OOPS) →