Manual testing

1. Why testing is required?

Ans. Software testing is required to check if the system is working as per design. Every prj before going to the customer it needs to get tested. Using testing software, we test software projects. At the tine of testing to check whether all clients requirements are satisfied or available or not.

1. What type of applications we test?

Ans. In real time the software projects are tested we have 3 types of s/w prjs which are tested. o/p is in the form of a) window based apps: (they are graphical user interface gui) which are opened in our system. Eg: paint, calculator, browsers, msoffice,etc. b)web based are which are opened in our browser. Eg: gmail, fb, any other sites. C) mobile apps.:which can be opened in any mobile eg: watsapp, fb, playstore,etc.

1. WhT IS SDLC and different phases in sdlc?

Ans. Sdlc means software development life cycle of an application and it has 6 phases where initial or requirement phase: each and every requirement we gather from one client and maintain a doc for each and every requirement, analysis phase: we plan every pt in the prj with the prj team , design phase: development team starts with the design of prj uml we draw diagram in a model level. , coding phase: development team writes the script in programming code lang. testers write test cases as per SRS, testing phase:wrtiien in automation script we write test cases for the s/w prj and write test data and test case template and execute test cases prepared in previous stage., delivery or maintainance: in this stage we deliver the prj to the client and after testing gets completed we release test code to production and support team monitoring if the system is running correctly in production.

1. What is waterfall model?

Ans. First process to be introduced. Also referred as sequential life cycle model. Simple and easy to understand. It has stages as initial ,analysis or product requirement, design or s/w arc., coding or implementation, testing or verification, delivery. Each phase must be completed before going to the next phase. It’s a time taking and a complex process. Each doc or requirement is fixed before itself and cant be changed.

1. What is agile method?

Ans. . Its made in the place of waterfall model so that we can accomadate any changes in the middle of the prject and for the need of faster s/w development.The Agile software development lifecycle is dominated by the iterative process. Each iteration results in the next piece of the software development puzzle. Each iteration is usually two to four weeks in length and has a fixed completion time. Multiple iterations will take place during the Agile software development lifecycle and each follows its own workflow. During an iteration, it is important that the customers and business stakeholders provide feedback to ensure that the features meet their needs.

It’s the most popular model and its mainly focused on customer requirements. It can very fastly develop the product.doing some portion of the prj and then check with the client and stakeholder at each and every stage of the process. It’s the combination of iterative and incremental process models with focus on process adaptability and customer satisfaction by rapid delivery of working s/w product. In this method we break the project into small incremental builds. These builds provide iterations.

Agile Modeling is a critical element to the rapid and continuous delivery of software. The model provides the developer with an understanding of what he or she will develop. Simultaneously, it provides the stakeholders with the same picture, which they can examine to ensure that the end goal meets their needs. This allows for rapid feedback and incremental updates throughout the process.The Goals of Agile Modeling is that the Agile Modeling is designed to support the goals of software development. The overall objectives of Agile Modeling include:

* Defining best practices for effective modeling
* Offering a way to apply those best practices
* Showing how to improve the modeling approach

1. Scrum technology?

Ans. Scrum is an agile way to manage a project, usually software development. Agile software product development with Scrum is often perceived as a methodology; but rather than viewing Scrum as methodology, think of it as a framework for managing a process used for completing complex projects and there possibilities are endless.

1. What Is dialy stand up meetings?

Ans. It’s a short organizational meeting that lasts for 5-15min each day and also called daily scrum. Purpose of meeting is to know what have you done yesday and wil to you presently and what are the defects in the work.

8.what is product backlog items?

Ans. It used for a unit for the amount of work done by the team in 1 sprint iteration and decomposed into 1 or more tasks.

9what is user testing backlog items in user story?

Ans. Featuring of a s/w by an end user perspective that is user story .

10what is task?

Ans. Its means breaking of the user story into parts for getting the work needs done.

11. what is sprint planning and sprint review meeting?

Ans. Sprint planning is a collaboration effort where there is involvement of scrum master he plans a meeting wid the product owner regarding the details and clarifies of product backlog items according to their acceptance criteria and whole of the agile team who define amount of work done and effort made to meet the sprint commitment.

Sprint review meeting: This means that at the end of each sprint, the team has produced a coded, tested and usable piece of software. So at the end of each sprint, a sprint review meeting is held. During this meeting, the Scrum team shows what they accomplished during the sprint and deliver a potentially shippable product.

12. sprint retrospective?

Ans. The sprint retrospective is a meeting facilitated by the ScrumMaster at which the team discusses the just-concluded sprint and determines what could be changed that might make the next sprint more productive.

12. whats the process in agile model?

Ans. Its made in the place of waterfall model so that we can accomadate any changes in the middle of the prject and for the need of faster s/w development.

**Concept** - Projects are envisioned and prioritized depends on the selection of the project.

**Inception** - Team members are identified, funding is put in place, and initial environments and requirements are discussed and intiate the project.

**Iteration/Construction** - The development team works to deliver working software based on iteration requirements and feedback. Build a working system where the iteration is based on changes in the needs of the stake holders.

**Release** - QA (Quality Assurance) testing, internal and external training, documentation development, and final release of the iteration into production and display the production.

**Production** - Ongoing support of the software while they operate and support the release.

**Retirement** - End-of-life activities, including customer notification and migration and remove the production.

This view presents the full Agile lifecycle model within the enterprise. In any enterprise there may be projects operating simultaneously, multiple sprints/iterations being logged on different product lines, and a variety of customers, both external and internal, with a range of business needs.

The Agile software development lifecycle is dominated by the iterative process. Each iteration results in the next piece of the software development puzzle. Each iteration is usually two to four weeks in length and has a fixed completion time. Multiple iterations will take place during the Agile software development lifecycle and each follows its own workflow. During an iteration, it is important that the customers and business stakeholders provide feedback to ensure that the features meet their needs.

It’s the most popular model and its mainly focused on customer requirements. It can very fastly develop the product.doing some portion of the prj and then check with the client and stakeholder at each and every stage of the process. It’s the combination of iterative and incremental process models with focus on process adaptability and customer satisfaction by rapid delivery of working s/w product. In this method we break the project into small incremental builds. These builds provide iterations. We interact with the client at every stage. In this method the steps are,

Planning , Requirements, design, coding, unit testing: testing done for each module of process to check validity of each module by delveopers. By this for each scenarios of module and writing results occurs in each step for each module, acceptance testing by end user: final testing based on specifications of the end-user or customer, or based on use by end-users/customers over some limited period of time.

13. what is sprint grooming?

Ans. Product backlog refinement—sometimes called product backlog grooming in reference to keep the backlog clean and in order. Its a meeting that is held near the end of one sprint to ensure the backlog is ready for the next sprint.

14. what is burndown chart and velocity?

Ans. Burndown Charts: A burn down chart is a graphical representation of work left to do vs time and the outstanding work (or backlog) is often on the vertical axis, with time along the horizontal. That is, it is a [run chart](https://en.wikipedia.org/wiki/Run_chart) of outstanding work. It is useful for predicting when all of the work will be completed. It is often used in [agile software development](https://en.wikipedia.org/wiki/Agile_software_development) methodologies such as [Scrum](https://en.wikipedia.org/wiki/Scrum_%28development%29). However, burn down charts can be applied to any project containing measurable progress over time. The burndownis a chartthat shows how quickly you and your team are burning through customer's user stories. It shows the total effort against the amount of work we deliver each iteration. Velocity: The rate of progress of a Scrum Team is called "velocity". It expresses the amount of e.g. story points completed per iteration. An import rule for calculating the velocity is that only stories that are completed at the end of the iteration are counted.

15. what is user acceptance criteria test cases?

Ans. User acceptance testing *(*UAT testing*)* is the last phase of the software testing process. Firstly, the criteria which the software is considered to be “working” needs to be assembled. Each case covers a specific usage scenario of the software. UAT – User Acceptance Testing –  is essential, typically, it’s not able to be undertaken until the application is largely feature-complete 10 prerequisites, which must be met before UAT can begin. These are:

1. Business Requirements must be available
2. Application Code should be fully developed
3. Unit Testing, Integration Testing & System Testing should be completed
4. No Show stoppers, or High or Medium defects in the System Integration Test Phase
5. Only Cosmetic errors are acceptable before UAT
6. Regression Testing should be completed with no major defects
7. All the reported defects should be fixed and tested
8. Traceability matrix for all testing should be completed
9. UAT Environment must be ready
10. Sign off mail or communication from System Testing Team that the system is ready for UAT execution.

16. what is v model?

Ans. V-model means verification and validation model. It’s an exhibition model its an extension of waterfall model. Its association of testing phase with development of each and every phase we are testing. Paralelly we can test each and every stage of process. The V-model is an SDLC model where execution of processes happens in a sequential manner in a V-shape. The V-Model is an extension of the waterfall model and is based on the association of a testing phase for each corresponding development stage. Validation involves actual testing and takes place after verification is done. Before development system plan is created test plan focuses on meeting functionality specified in requirements gathering. In high level design is based on system architecture and design integration test plan is created as well as in order to test pieces of system abitlity to work together. low level design is where actual software companies are designed and unit tests are created. Implementation .

17. when do v use integration testing?

Ans Integration testing (sometimes called integration and testing, abbreviated I&T) is the phase in software testing in which individual software modules are **combined** and tested as a group. It occurs after unit testing and before validation testing.

18. What To Do When There Isn’t Enough Time To Test?

Ans. Firstly, Why does this happen? Many reasons – some of which are:

**#1) Incorrect Estimation**:

If you started with an inaccurate expectation, things are bound to fail. A good test estimate must take the following into account:

* **Time for preparatory tasks –** We are talking about tasks such as:
  + Identifying and putting together a regression suite
  + Creating Test data
  + Time to determine test readiness (E.g.: Smoke/Sanity Test), etc.
* **Test case maintenance**: Test cases are long-term usage assets. They are sure to undergo minor updates during execution. It is recommended that for new products up to 30% of your test execution time should be allocated for these minor maintenance tasks. All teams and projects might not need 30%, but do allocate some time and effort for this task.
* [**Ad-hoc**](http://www.softwaretestinghelp.com/ad-hoc-testing/)**/Exploratory testing** – The count of scripted tests is a major denominator for test estimation numbers. However, no test team in this world will deny exploring your software even if the model is dominantly scripted.
* **Reporting/Communication** – This includes triage/stand up meetings, updating work management tools etc.
* **Contingency factor:** Standards recommend 25-30% buffer to your original estimates. But teams can rarely afford it. Even then, leave a little breathing room, when possible.
* **Team and its capabilities:** If you have a new team or if they re using a tool for the first time, you might need to set some time aside for training. Tailor your estimates based on your team you are working with.

**2) Unstable builds and other technical problems:**

* **Smoke/Sanity test failure**: When the basic tests on the AUT fail after deployment into QA environment there is pretty much nothing the QA team can do towards test execution. It is true that we can work on other tasks while this happens, but it still will not fill the [test cycle](http://www.softwaretestinghelp.com/what-is-software-testing-life-cycle-stlc/) time. So, this is a major contributor to time wasted.
* [**Test data**](http://www.softwaretestinghelp.com/tips-to-design-test-data-before-executing-your-test-cases/) **unavailable**: Production-like data is a must for every testing project. Not getting this into the QA environment on time is also another blocking factor. Sometimes testers can work around this by [creating and managing their own test data](http://www.softwaretestinghelp.com/test-data-management-techniques/), but it is time-consuming and might not always be on-point.
* [**Environment issues**](http://www.softwaretestinghelp.com/test-bed-test-environment-management-best-practices/) – The build failing deployments, the server keeps getting timed out, many more such issues eat away your test cycle. This probably stems from the fact that, some companies (not all) undermine the importance of a good, live-like environment for effective QA. They often try to get away low-capacity servers and make-do set ups. This is really a short-time fix and does nobody any favors. In fact, it could cost them the quality of testing and loss of valuable test time.
* **3) Lack of agreement between all parties involved:**
* ------------
* This might be a rare problem with teams following Agile or [SAFe](http://www.scaledagileframework.com/) due to the close circles they work in, but many teams still suffer from disagreement or miscommunication as to when Dev, Ops, and QA is supposed to receive deliverables from one another. Hence, delays.

# 19. [How to handle client if you found a high severity defect on the release date?](https://sqa.stackexchange.com/questions/18226/how-to-handle-client-if-you-found-a-high-severity-defect-on-the-release-date)

**Ans.** The fact that the defect has been found close to the deadline is, in the short term, irrelevant. Your team has found a high severity defect, so you report it. Given the short timescales, you ensure that everyone who needs to know about it knows about it, so they have the information they need to determine -their- best course of action as soon as possible.

You must -absolutely- not ever hold off from reporting an issue, at least to your local management structure. That would, at the very least, ruin the reputation of your team and could potentially have much more serious consequences.

20. when do we use automation testing?

#### Ans. But with automated testing, test cases are executed with the assistance of tools, scripts, and software. automated testing cover two vast areas. Within each category, specific testing methods are available, such as **black box testing, white box testing, integration testing, system testing, performance testing, and** [**load testing**](https://www.apicasystem.com/load-testing/). Automated testing is the preferred option in the following areas/scenarios:

* **Regression Testing:** Here, automated testing is suitable because of frequent code changes and the ability to run the regressions in a timely manner.
* **Load Testing:** Automated testing is also the best way to complete the testing efficiently when it comes to load testing. [Learn more about load testing with our best practices guide here](http://resources.apicasystem.com/guide-load-testing-101)
* **Repeated Execution:** Testing which requires the repeated execution of a task is best automated.
* **Performance Testing:** Similarly, testing which requires the simulation of thousands of concurrent users requires automation.

Keeping these factors in mind, you can find the best approach in any given testing situation and achieve quality output well within your budget and timeline.

21. The Role of a Tester in SDLC in each phase?  
  
1. Tester prepares the Test cases, Test Scenarios  from the SRS  
2.  Using the script the tester performs different kinds of testing (Regression, Function)  
3. Tester Notes the results(pass/Fail)  
4. If Result=Fail then the scenario is raised in the Test director   
5. Once its fixed by the developer the tester performs a regression testing

22. Types of Non Functional Software Testing

## Ans. Non functional testing ensures that a system/application meets the specified performance requirements. In non functional software testing, by performance we do not only mean response time, but several other factors such as security, scalability and usability of the application as well. Types of Non Functional Testing

1. Performance Testing

2. Load Testing

3. Stress Testing

4. Volume Testing

5. Failover Testing

6. Security Testing

7. Compatibility Testing

8. Usability Testing

9. Scalability Testing

**Let's understand all the types of non functional testing in detail:**

### Performance Testing :

First and foremost type of non functional testing is performance testing. In order to ensure that the response time of a system is acceptable, performance testing is carried out. By setting up a considerable load and a production-sized database, the system is tested for response times of several business critical processes.

### Load Testing :

Types of non functional testing in [software testing](http://www.testing-whiz.com/) also includes load testing. To check whether the system can sustain the pressure or load of many users accessing the system at one time, load testing needs to be carried out. The production load is replicated in the test environment in this case after which the application/system is tested.

### Stress Testing :

This testing is done to pull the system far beyond its capabilities and see how it reacts. Contrary to load testing in which the maximum allowable load is generated, in stress testing, the load generated is more than what the system is expected to handle.

### Volume Testing :

When storage requirements and capabilities of the system are to be tested, volume testing is done. When a huge database size is encountered, system’s performance and its ability to exchange data and information are tested in this case.

### Failover Testing :

To test how well the redundancy mechanism works when the system encounters heavy load or unexpected failure is what failover testing is about. Also, when the specific failed system is back again, it must begin to function as per requirements – this is fail-back testing.

### Security Testing :

While performing non functional testing, to test how well the system can preserve itself and the data it holds in situation of malicious attacks is called security testing. Confidentiality, integrity, availability, authentication and authorization are the main areas that are tested when security testing is considered. Also, network security, system security and application security are other areas that will be tested in this case.

### Compatibility Testing :

An application is tested for its coordination with different hardware and software that it is expected to work on. Another testing that can be done is working of the application with different versions or releases of the same hardware or software.

### Usability Testing :

To verify the ease of usage of an interface within an application is what usability testing is about. Learnability and memorability of the application are main factors in this case. This testing is particularly important when [testing GUI](http://www.testing-whiz.com/web-ui-comparison-and-functional-test-automation).

### Scalability Testing :

When an application is tested for it ability to increase and scale up on any of its non-functionality requirements such as load, number of transactions, number of servers, volume of data etc., it is known as scalability testing.

The above list is not exhaustive since there are more than 150 testing types as of today. The different types of testing, including [automated testing](http://www.testing-whiz.com/), that needs to be done depending upon the scope of the project and maturity of the application under test.

24.what is test case>?

Ans. A test case is a set of conditions or variables under which a tester will determine whether a system under test satisfies requirements or works correctly. The process of developing test cases can also help find problems in the requirements or design of an application. A test case is a document, which has a set of test data, preconditions, expected results and postconditions, developed for a particular test scenario in order to verify compliance against a specific requirement.

Test Case acts as the starting point for the test execution, and after applying a set of input values, the application has a definitive outcome and leaves the system at some end point or also known as execution postcondition.

25. what is test planning/test strategy document?

Ans. A **Test Strategy document** is a high level **document** and normally developed by project manager. This **document** defines “Software **Testing Approach**” to achieve **testing** objectives. ... The **Test Strategy document** is a static **document** meaning that it is not updated too often.

26. what is Exit and Entry criteria ?

Ans. **Entry Criteria** for QA **testing** is defined as “Specific conditions or on-going activities that must be present before a process can begin”. In the Systems Development Life Cycle it also specifies which **entry criteria** are required at each phase. ... The second is what we produce that acts as input to later **test** process steps.

The Exit criteria is a set of conditions based on which you can say this particular task is finished.  
This can be difficult to determine. Many modern software applications are so complex, and run in such  
 as interdependent environment, that complete testing can never be done. "When to stop testing" is one of the most difficult questions to a test engineer.

27. what is TDD and BDD (cucumber framework)?

Ans. **BDD** – Behavior-Driven Development – is perhaps the biggest source of confusion. When applied to automated testing, **BDD** is a set of best practices for writing great tests. **BDD** can, and should be, used together with **TDD** and unit testing methods. One of the key things **BDD** addresses is implementation detail in unit tests.

28. how do we write test cases in BDD format?

Ans. writing test cases for BDD (Behavior Driven Development) using specflow. If I write comprehensive tests with BDD BDD test cases can be understood by non developers and testers.

29. what is priority and severity in defect?

**Ans. Priority** is defined as the order in which a **defect** should be fixed. Higher the **priority** the sooner the **defect** should be resolved. **Defects** that leave the software system unusable are given higher **priority** over **defects** that cause a small functionality of the software to fail.

30. how to estimate test cases?

1. Ans. 3-Point Software Testing Estimation Technique.
2. Use – Case Point Method:
3. Work Breakdown Structure.
4. Wideband Delphi technique.
5. Function Point/Testing Point Analysis.
6. Percentage of development effort method.
7. Percentage distribution.
8. Best Guess.

31. what is most challenge defect u came across?

Ans. thinking along the lines of time constraints, difficulty covering device fragmentation, poor communication, etc.

32, what are test design techniques?

By **design** we mean to create a plan for how to implement an idea and **technique** is a method or way for performing a task. So, **Test Design** is creating a set of inputs for given software that will provide a set of expected outputs. ... Broadly speaking there are two main categories of **Test Design Techniques**.

33. if we dont have time to test call test cases what we will do?

**1) Estimate accurately.** When in doubt over-estimate by a reasonable margin, but not underestimate. Don’t forget to make estimate adjustments based on your team, tools and processes. When done, seek official sign off so everyone is aware and is in kept in the loop.

**#2)** Take historical data into consideration – **The Test Management tool is your best friend**.

* How long did the earlier release test cycles take?
* What kind of issues caused interruptions to the previous test cycle?
* How many runs did most test cases take before they passed?
* What defects were reported?
* What defects caused the testing to be interrupted?

**3) Ask these questions and plan accordingly in crunch time:**

* Find out Important functionality is your project?
* Find out High-risk module of the project?
* Which functionality is most visible to the user?
* Which functionality has the largest safety impact?
* Which functionality has the largest financial impact on users?
* Which aspects of the application are most important to the customer?
* Which parts of the code are most complex, and thus most subject to errors?
* Which parts of the application were developed in rush or panic mode?
* What do the developers think are the highest-risk aspects of the application?
* What kinds of problems would cause the worst publicity?
* What kinds of problems would cause the most customer service complaints?
* What kinds of tests could easily cover multiple functionalities?

Considering these points, you can greatly reduce the risk of project releasing under less time constraint.

**#4) Use a Test Management tool.** This will significantly reduce the amount of preparation, reporting and maintenance time and effort.

**5)** There is not much we can do about incorrect builds/technical issues, but the one thing that can help is looking at the Unit test results. This will give us an idea as to whether the build was a success or not and what kind of tests did it fail – so we don’t reinvent the wheel.

If your **Test Management Tool supports** [**CI integration**](http://www.softwaretestinghelp.com/continuous-integration/), you have that information available without any fuss so you understand the stability of the application better.

**#6) Measure your productivity and progress often**. Don’t let status reports be a deliverable just for the benefit of the external teams. Make sure you are closely monitoring your daily targets and your ability to accomplish them.

Also, be sure to not get into the classic conundrum of ‘Velocity vs. Quality’. Because, when you report, say, 50 bugs a day, it might appear as if you are being super productive. But if most of them are coming back as invalid ones, you have got yourself a problem.

So monitor, monitor and monitor a little more.

28. how we learn the functionality of system?

### Ans. a Quality Management System,

29. what are the tools to manage defects/userstories?

Ans. In Lifecycle, you can create Defects to plan and track fixes. Defects are part of the [Backlog](https://community.versionone.com/VersionOne-Lifecycle/The_Agile_Process/Product_Planning/Building_and_Managing_the_Backlog) and are managed like other backlog items, just with different attributes. Just like stories/backlog items, you can assign size estimates to defects that factor into the team's velocity. You can then break them out into [tasks](https://community.versionone.com/VersionOne-Lifecycle/The_Agile_Process/Sprint_or_Iteration_Planning/Assigning_Work_to_Team_Members/Detail_Planning/Using_Tasks_to_Breakdown_Stories%2F%2FBacklog_Items) and [acceptance tests](https://community.versionone.com/VersionOne-Lifecycle/Testing_in_Lifecycle/Acceptance_Tests) to allocate the work across multiple team members, or to more precisely identify the work required in a larger effort.

30. who will assign the work?

Ans. Generally test lead assign the work to test engineer. It   
depends on the organisation where Test lead and Test   
enginerrs combindly may start the process based on SRS and   
DDD. Test lead (or) in some organizations Team Mentors will   
assign the work to the test engineer.

31. what is requirement traceability matrix?

## Ans. The Requirements Traceability Matrix (RTM) is a document that links requirements throughout the validation process. The purpose of the Requirements Traceability Matrix is to ensure that all requirements defined for a system are tested in the test protocols. Requirements tracing, a process of documenting the links between the requirements and the work products developed to implement and verify those requirements. The RTM captures all requirements and their traceability in a single document delivered at the conclusion of the life cycle. Requirement traceability Matrix - Parameters:

* Requirement ID
* Risks
* Requirement Type
* Requirement Description
* Trace to Design Specification
* Unit Test Cases
* Integration Test Cases
* System Test Cases
* User Acceptance Test Cases
* Trace to Test Script

32.

Ans. n software projects, it is most important to measure the quality, cost and effectiveness of the project and the processes. Without measuring these, project can’t be completed successfully.

*In today’s article we will learn with examples and graphs –* ***Software test metrics and measurements*** *and how to use these in software testing process.*

There is a famous statement: ***“We can’t control things which we can’t measure”.*** Here controlling the projects means, how a project manager/lead can identify the deviations from the test plan ASAP in order to react in the perfect time. Test metrics: Generation of test metrics based on the project needs is very much important to achieve the quality of the software being tested. *A Metric is a quantitative measure of the degree to which a system, system component, or process possesses a given attribute.*

***Metrics can be defined as “STANDARDS OF MEASUREMENT”.***

Software Metrics are used to measure the quality of the project. Simply, Metric is a unit used for describing an attribute. Metric is a scale for measurement.

Suppose, in general, “Kilogram” is a metric for measuring the attribute “Weight”. Similarly, in software, “How many issues are found in thousand lines of code?”, h*ere No. of issues is one measurement & No. of lines of code is another measurement. Metric is defined from these two measurements*.

**Test metrics example:**

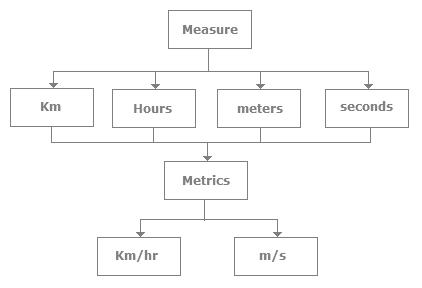
* How many defects are existed within the module?
* How many test cases are executed per person?
* What is the Test coverage %?

### What is Software Test Measurement?

Measurement is the quantitative indication of extent, amount, dimension, capacity, or size of some attribute of a product or process.

**Test measurement example:** Total number of defects.

Please refer below diagram for clear understanding of the difference between Measurement & Metrics.

[](http://cdn2.softwaretestinghelp.com/wp-content/qa/uploads/2014/09/difference-between-Measurement-Metrics.jpg)

34. wat s development environment?

Ans. In computer program and software product **development**, the **development environment** is the set of processes and programming tools used to create the program or software product. The term may sometimes also imply the physical **environment**. n computer program and software [product development](http://searchcio.techtarget.com/definition/product-development-or-new-product-development-NPD), the development environment is the set of processes and programming tools used to create the program or software product. The term may sometimes also imply the physical environment. An [integrated development environment](http://searchsoftwarequality.techtarget.com/definition/integrated-development-environment) is one in which the processes and tools are coordinated to provide developers an orderly interface to and convenient view of the development process (or at least the processes of writing code, testing it, and packaging it for use). An example of an IDE product is Microsoft's Visual Studio .NET. The term *computer-assisted software environment* ([CASE](http://searchcio-midmarket.techtarget.com/definition/CASE)) is generally used to describe a set of tools and practices that facilitate management of a software development project.

35. wats QA environment?

Ans. A **QA environment** is where you test your upgrade procedure against data, hardware, and software that closely simulate the Production **environment** and where you allow intended users to test the resulting Waveset application. A Production **environment** is where the Waveset application is actually available for business use.

36. What is staging environment?

Ans. A stage or **staging environment** is an **environment** for testing that exactly resembles the production **environment**. In other words, it's a complete but independent copy of the production **environment**, including the database. **Staging** provides a true basis for QA testing because it precisely reproduces what is in production.

37. what is production environment?

Ans. A **production environment** is where the real-time staging of programs that run an organization are executed, and includes the personnel, processes, data, hardware, and software needed to perform day-to-day operations. Production environment is a term used mostly by developers to describe the setting where software and other products are actually put into operation for their intended uses by end users. A production environment can be thought of as a real-time setting where programs are run and hardware setups are installed and relied on for organization or commercial daily operations.

One way to define a production environment is by contrasting it with a testing environment. In a testing environment, a product is still being used theoretically. Users, typically engineers, look for bugs or design flaws. In the production environment, the product has been delivered and needs to work flawlessly.  
  
A related term, production code, refers to code that is being used by end users in a real-time situation, or code that is useful for end-user operations. A debate over what constitutes production code shows that there is a lot of ambiguity about the formal application of either term to a specific scenario because of the many stages that code and tech products go through in their respective life cycles.

38. how to deal the production defects?

Ans. Unfortunately, no factory’s production is perfect, as defects are common in manufacturing. They can come in all shapes and sizes. And they’re a problem that can significantly affect you, as an importer, and your bottom line. The goal here is to get your supplier's factory to reduce the frequency of unacceptable product defects and be accountable for those that still appear in the finished goods (**related:** [How Experienced Importers Limit Product Defects in 3 Stages [eBook]](https://www.intouch-quality.com/how-experienced-importers-limit-product-defects-in-3-stages)). Let's discover three ways to handle defective products.

## 1. Manage expectations for allowable defects

When talking with a new supplier, or trying to encourage improvement in an established relationship, it's important to manage expectations. Part of this means stating clearly to your supplier the types and quantities of defects that may be acceptable.

## 2. Identify and address issues with product inspection

A great way to handle defective products is to catch quality issues early, before they make their way into the finished goods. Identifying issues before shipping helps you avoid making assumptions about product quality—assumptions that can cause supply chain disruptions and cost you money if you find a significant portion of the goods you receive is unsellable. There are different stages during the production process where inspection can be performed to show you the current state of your order.

## 3. Accepting any defects that remain in finished goods

Sometimes you may find defects on an item, such as small black dots or scratches, which are minor enough that you can accept, especially if the product is relatively inexpensive. Other times you may find defects you don’t want to accept but reworking costs are too high. And then there are times when you feel the best way to handle defective products is through rework.

From the development manager's standpoint, it absolutely matters whether the defect is new or whether it is an existing defect because it has a direct and immediate impact on how the bug needs to be handled. From the development manager's standpoint, the most important question is whether a new bug needs to be resolved in the current release cycle or whether or whether it can wait and be prioritized in a later cycle. That, in turn, often depends on whether the bug is new or not.

If you've found a new bug, that implies that one of the new features/ bug fixes in the current release cycle introduced the issue. If that's the case, someone either needs to remedy the issue as part of the current release (either by reverting the change that introduced the bug or by fixing the bug itself) or the business needs to decide whether adding new feature X is worth deploying even if it introduces new bug Y. Almost always, the bug has to be resolved in the current build cycle or the offending change needs to be rolled back. On the other hand, if you've found an old bug that existed prior to the current round of changes, the current build cycle can generally continue and the new bug can be prioritized for a future release. Of course, there are cases where a newly identified bug needs to be handled in the current release cycle because the bug is just that critical, those cases tend to be rare.

Now, whether it should be QA's responsibility to check whether the bug affects the current release or whether that should be done by whoever is prioritizing the bugs (assuming that prioritization happens immediately) is an open question. My bias would be to ask QA to do it since they're already writing up the bug. Since the QA person already knows how to reproduce the bug, they're best positioned to verify whether it exists in production or not. The QA department also tends to have more hours available for this sort of investigation than the person doing the prioritization does since the work can be spread across many analysts.