1. **Why testing is required?**

1. Testing is always needed for correctly understand the fault errors in software during its development phases.

2. It is necessary because it always ensures the users or customers satisfaction and reliability of the application.

3. It is needed in software development to increase the reliability and quality of the software.

4. **Testing is needed** to provide the various facilities to the users like delivery of high quality software or application, lower maintenance costs and more accurate and reliable results also.

1. **What types of application we need to test..?**

**Black box testing** – Internal system design is not considered in this type of testing. Tests are based on requirements and functionality.

**White box testing** – This testing is based on knowledge of the internal logic of an application’s code. Also known as Glass box Testing. Internal software and code working should be known for this type of testing. Tests are based on coverage of code statements, branches, paths, conditions.

**Unit testing** – Testing of individual software components or modules. Typically done by the programmer and not by testers, as it requires detailed knowledge of the internal program design and code. may require developing test driver modules or test harnesses.

**Incremental integration testing** – Bottom up approach for testing i.e continuous testing of an application as new functionality is added; Application functionality and modules should be independent enough to test separately. done by programmers or by testers.

**Integration testing** – Testing of integrated modules to verify combined functionality after integration. Modules are typically code modules, individual applications, client and server applications on a network, etc. This type of testing is especially relevant to client/server and distributed systems.

**Functional testing** – This type of testing ignores the internal parts and focus on the output is as per requirement or not. Black-box type testing geared to functional requirements of an application.

**System testing** – Entire system is tested as per the requirements. Black-box type testing that is based on overall requirements specifications, covers all combined parts of a system.

**End-to-end testing** – Similar to system testing, involves testing of a complete application environment in a situation that mimics real-world use, such as interacting with a database, using network communications, or interacting with other hardware, applications, or systems if appropriate.

**Sanity testing**– Testing to determine if a new software version is performing well enough to accept it for a major testing effort. If application is crashing for initial use then system is not stable enough for further testing and build or application is assigned to fix.

**Regression testing** – Testing the application as a whole for the modification in any module or functionality. Difficult to cover all the system in regression testing so typically automation tools are used for these testing types.

**Acceptance testing** -Normally this type of testing is done to verify if system meets the customer specified requirements. User or customer do this testing to determine whether to accept application.

**Load testing** – Its a performance testing to check system behavior under load. Testing an application under heavy loads, such as testing of a web site under a range of loads to determine at what point the system’s response time degrades or fails.

**Stress testing** – System is stressed beyond its specifications to check how and when it fails. Performed under heavy load like putting large number beyond storage capacity, complex database queries, continuous input to system or database load.

**Performance testing** – Term often used interchangeably with ‘stress’ and ‘load’ testing. To check whether system meets performance requirements. Used different performance and load tools to do this.

**Usability testing** – User-friendliness check. Application flow is tested, Can new user understand the application easily, Proper help documented whenever user stuck at any point. Basically system navigation is checked in this testing.

**Install/uninstall testing**– Tested for full, partial, or upgrade install/uninstall processes on different operating systems under different hardware, software environment.

**Recovery testing** – Testing how well a system recovers from crashes, hardware failures, or other catastrophic problems.

**Security testing** – Can system be penetrated by any hacking way. Testing how well the system protects against unauthorized internal or external access. Checked if system, database is safe from external attacks.

**Compatibility testing** – Testing how well software performs in a particular hardware/software/operating system/network environment and different combination s of above.

**Comparison testing** – Comparison of product strengths and weaknesses with previous versions or other similar products.

**Alpha testing** – In house virtual user environment can be created for this type of testing. Testing is done at the end of development. Still minor design changes may be made as a result of such testing.

**Beta testing** – Testing typically done by end-users or others. Final testing before releasing application for commercial purpose.

1. **what is SDLC and different phases in SDLC?**

**SDLC** :

Software Development Life Cycle is a process used by software industry to design, develop and test high quality softwares. The SDLC aims to produce a high quality software that meets or exceeds customer expectations, reaches completion within times and cost estimates.

**Different phases like:**

**Requirement Analysis and planning :** Senior team members analyze the requirements/input given by customers/business users. They will check whether the requirement is feasible or not (can be done or not). They also identify the risks associated with project.

Note: this high level requirements will be written in BRD (Business Requirement document) by Business Analyst

**Define/Design** : in the define stage Business Analyst define more details about requirements (which are in BRD) in the form of SRS (software requirement specification) or Use Case diagram.

As part of design,

Senior Developers write High Level Design Document (HLD)

Developers write Low Level Design Document (LLD)

Seniors Tester write Test Planning document

**Implementation/Development**: Developers write the code for the requirements

Testers write test cases as per SRS

**Testing** : Execute the test cases what we prepared in previous stage

**Deployment** : Release the tested code to production

**Maintenance** : Support team monitoring the system that is running in production

1. **what is waterfal in SDLC?**

In "The Waterfall" approach, the whole process of software development is divided into separate phases. In Waterfall model, typically, the outcome of one phase acts as the input for the next phase sequentially.

Following is a diagrammatic representation of different phases of waterfall model.

****

1. **what is the process in agile model**

In agile the tasks are divided to time boxes (small time frames) to deliver specific features for a release.

Iterative approach is taken and working software build is delivered after each iteration. Each build is incremental in terms of features; the final build holds all the features required by the customer.

Here is a graphical illustration of the Agile Model:



(Agile software development refers to a group of software development methodologies based on iterative development, where requirements and solutions evolve through collaboration between self-organizing cross-functional teams. Agile methods or Agile processes generally promote a disciplined project management process that encourages frequent inspection and adaptation.)

1. **what is scrum methodology**

Scrum is a subset of Agile. It is a lightweight process framework for agile development, and the most widely-used one. Scrum is most often used to manage complex software and product development, using iterative and incremental practices. Scrum significantly increases productivity and reduces time to benefits relative to classic “waterfall” processes.

1. **what is daily standup meeting and what we discuss**

The daily stand-up meeting (also known as a "dailyscrum.) is simple to describe: The whole team meets every day for a quick status update.

The purpose of the meeting is for each team member to answer the following three questions:

1) What did you do yesterday?

2) What will you do today?

3) Are there any impediments in your way?

1. **what is user story/feature/sprint back log items and tasks in user story**

xyzzzzzz

1. **what is sprint planning and spring retro..?**

**Sprint Planning Meeting :**

In [Scrum](https://www.mountaingoatsoftware.com/agile/scrum), the sprint planning meeting is attended by the product owner, ScrumMaster and the entire Scrum team. Outside stakeholders may attend by invitation of the team, although this is rare in most companies.

During the sprint planning meeting, the product owner describes the highest priority features to the team. The team asks enough questions that they can turn a high-level user story of the product backlog into the more detailed tasks of the sprint backlog

**Sprint Retrospective :** No matter how good a [Scrum](https://www.mountaingoatsoftware.com/agile/scrum) team is, there is always opportunity to improve. Although a good Scrum team will be constantly looking for improvement opportunities, the team should set aside a brief, dedicated period at the end of each sprint to deliberately reflect on how they are doing and to find ways to improve. This occurs during the sprint retrospective.

The sprint retrospective is usually the last thing done in a sprint. Many teams will do it immediately after the sprint review. The entire team, including both the ScrumMaster and the product owner should participate. You can schedule a scrum retrospective for up to an hour, which is usually quite sufficient. However, occasionally a hot topic will arise or a team conflict will escalate and the retrospective could take significantly longer.

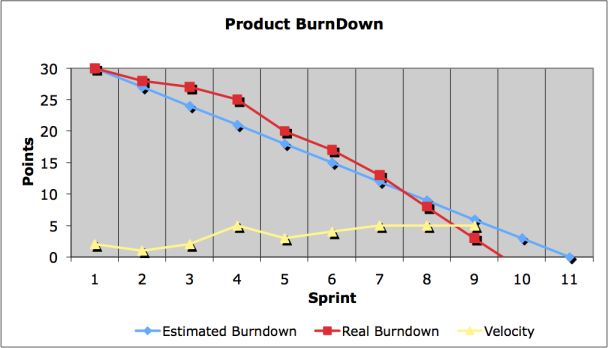
Although there are many [ways to conduct an agile sprint retrospective](http://www.mountaingoatsoftware.com/reviews/27-agile-retrospectives), our recommendation is to conduct it as a start-stop-continue meeting. This is perhaps the simplest, but often the most effective way to conduct a retrospective. Using this approach each team member is asked to identify specific things that the team should:

* Start doing
* Stop doing
* Continue doing

1. **what is burndown chart and velocity**

**Scrum Burndown Chart :**

The Scrum Burndown Chart is a visual measurement tool that shows the completed work per day against the projected rate of completion for the current project release. Its purpose is to enable that the project is on the track to deliver the expected solution within the desired schedule.



**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.

1. **what is product backlog item and sprint backlog items**

* **Product Backlog** => All work items related to a product/project, ordered by a Product Owner.
* **Sprint Backlog** => All work committed to and pushed into a Development Teams's upcoming Sprint/iteration as chosen by the Development Team in their Sprint Planning meeting.

1. **what is user acceptance criteria test cases**

User acceptance is a type of testing performed by the Client to certify the system with respect to the requirements that was agreed upon. This testing happens in the final phase of testing before moving the software application to Market or Production environment.

1. **what is v model?**

V- model means Verification and Validation model. Just like the [**waterfall model**](http://istqbexamcertification.com/what-is-waterfall-model-advantages-disadvantages-and-when-to-use-it/), the V-Shaped life cycle is a sequential path of execution of processes. Each phase must be completed before the next phase begins.  Testing of the product is planned in parallel with a corresponding phase of development in **V-model**.

**Diagram of V-model:**



1. **what is STLC?**

Contrary to popular belief, Software Testing is not a just a single activity. It consists of series of activities carried out methodologically to help certify your software product. These activities (stages) constitute the Software Testing Life Cycle (STLC).

The different stages in Software Test Life Cycle -

[](http://cdn.guru99.com/images/stories/software-test-life-cycle.jpg)

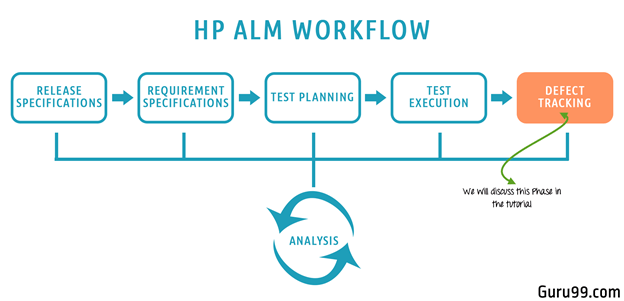
1. **what is defect?**

When a tester executes the test cases, he might come across the test result which is contradictory to expected result. This variation in the test result is referred as a **defect**. These defects or variation are referred by different names in a different organization like**issues, problem, bug or incidents**.

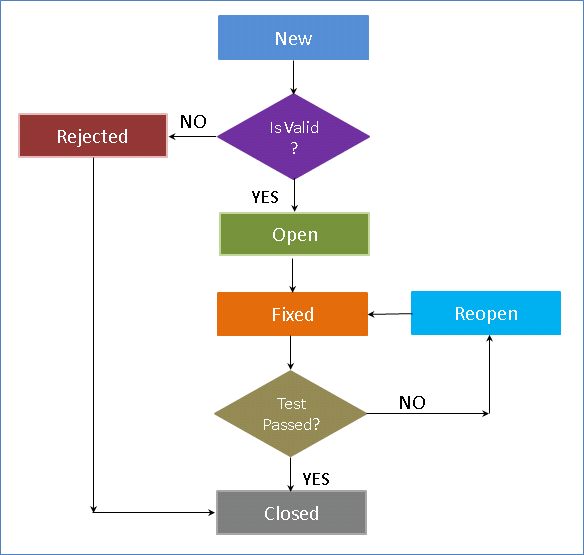
1. **how to arise a defect and what we specify while logging defect?**

A defect is logged during the test execution, when expected result and actual result don't match with each other.

Defect module in HP ALM not only helps users to post the defects but also enables them to track and gives the overall quality of the release at any stage of the development process.

[](http://cdn.guru99.com/images/hpalm/071114_0947_DefectManag1.png)

Default Defect Life Cycle in ALM:

[](http://cdn.guru99.com/images/hpalm/071114_0947_DefectManag2.png)

**Status** Explanation

**New** When a defect is posted, the default status is 'New'

**Open** When the defect is accepted by developers it is moved to 'Open' Status

**Rejected** When the defect is rejected by developers it is moved to 'Rejected' Status

**Fixed** When the defect is fixed by developers it is moved to 'Fixed' Status. Testers would pick up all defects for testing that are in status 'Fixed'.

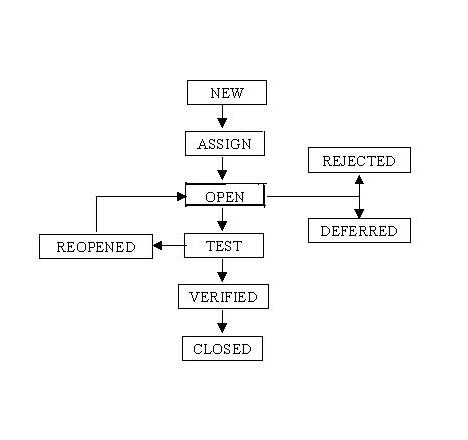
**Reopen** If the testing has failed, the defect is moved to 'Reopen' status

**Closed** If the testing has passed, the defect is moved to 'Closed' Status.

1. **defect lifecycle**

Defect life cycle is a cycle which a defect goes through during its lifetime. It starts when defect is found and ends when a defect is closed, after ensuring it’s not reproduced. [**Defect life cycle**](http://istqbexamcertification.com/what-is-a-defect-life-cycle/) is related to the bug found during testing.

The bug has different states in the Life Cycle. The Life cycle of the bug can be shown diagrammatically as follows:



## Defect Life Cycle States:

* **New -**Potential defect that is raised and yet to be validated.
* **Assigned -**Assigned against a development team to address it but not yet resolved.
* **Active -**The Defect is being addressed by the developer and investigation is under progress. At this stage there are two possible outcomes; viz - Deferred or Rejected.
* **Test -**The Defect is fixed and ready for testing.
* **Verified -**The Defect that is retested and the test has been verified by QA.
* **Closed -**The final state of the defect that can be closed after the QA retesting or can be closed if the defect is duplicate or considered as NOT a defect.
* **Reopened -**When the defect is NOT fixed, QA reopens/reactivates the defect.
* **Deferred -**When a defect cannot be addressed in that particular cycle it is deferred to future release.
* **Rejected -**A defect can be rejected for any of the 3 reasons; viz - duplicate defect, NOT a Defect, Non Reproducible.

1. **Different types of testing?**
2. **when do we use regression testing?**

When a [bug](http://testingbasicinterviewquestions.blogspot.in/2012/05/how-to-report-bug-in-small-companies.html) is fixed by the development team than testing the other features of the applications which might be affected due to the bug fix is known as **regression testing**.

**Regression testing** is always done to verify that modified code does not break the existing functionality of the application and works within the requirements of the system.

#### When will we do Regression Testing :

**Regression testing** is the re-testing of features to make safe that features working earlier are still working fine as desired.  
It is executed when any new build comes to QA, which has bug fixes in it or during releasing cycles (Alpha, Beta or GA) to originate always the endurance of product.

**Regression testing** will be conducted after any bug fixed or any functionality changed.

1. **when do we use integration testing?**

We normally do Integration [testing after “Unit testing”](http://www.softwaretestinghelp.com/unit-testing/).

Once all the individual units are created and tested, we start combining those “Unit Tested” modules and start doing the integrated testing. So the meaning of Integration testing is quite straight forward- Integrate/combine the unit tested module one by one and test the behavior as a combined unit.

The main function or goal of Integration testing is to test the interfaces between the units/modules.

The individual modules are first tested in isolation. Once the modules are unit tested, they are integrated one by one, till all the modules are integrated, to check the combinational behavior, and validate whether the requirements are implemented correctly or not.

Here we should understand that, Integration testing does not happens at the end of the cycle, rather it is conducted simultaneously with the development. So in most of the times all the modules are not actually available to test and here is what the challenge comes to test something which does not exists!

### ****Approaches****

There are fundamentally 2 approaches for doing Integration testing:

1. Bottom up approach
2. Top down approach.
3. **when do we use smoke testing and sanity testing?**

**Smoke Testing** is performed after software build to **ascertain that the critical functionalities of the program is working fine**.It is executed **"before"** any detailed functional or regression tests are executed on the software build.The **purpose is to reject a badly broken application**, so that the QA team does not waste time installing and testing the software application.

In Smoke Testing, the **test cases chosen cover the most important functionality** or component of the system. The objective is not to perform exhaustive testing, but to verify that the critical functionalities of the system is working fine.  
For Example a typical smoke test would be - Verify that the application launches successfully, Check that the GUI is responsive ... etc.

## Sanity Testing

After receiving a **software build, with minor changes in code, or functionality, Sanity testing is performed to ascertain that the bugs have been fixed and no further issues are introduced due to these changes**.The goal is to determine that the proposed functionality works roughly as expected. **If sanity test fails, the build is rejected to save the time and costs involved in a more rigorous testing**.

The **objective is "not" to verify thoroughly the new functionality**, but to determine that the developer has applied some rationality (sanity) while producing the software. For instance, if your scientific calculator gives the result of 2 + 2 =5! Then, there is no point testing the advanced functionalities like sin 30 + cos 50.

## Smoke Testing Vs Sanity Testing - Key Differences :

|  |  |
| --- | --- |
| Smoke Testing | Sanity Testing |
| Smoke Testing is performed to ascertain that the critical functionalities of the program is working fine | Sanity Testing is done to check the new functionality / bugs have been fixed |
| The objective of this testing is to verify the "stability" of the system in order to proceed with more rigorous testing | The objective of the testing is to verify the "rationality" of the system in order to proceed with more rigorous testing |
| This testing is performed by the developers or testers | Sanity testing is usually performed by testers |
| Smoke testing is usually documented or scripted | Sanity testing is usually not documented and is unscripted |
| Smoke testing is a subset of Regression testing | Sanity testing is a subset of Acceptance testing |
| Smoke testing exercises the entire system from end to end | Sanity testing exercises only the particular component of the entire system |
| Smoke testing is like General Health Check Up | Sanity Testing is like specialized health check up |

1. **what is unit testing?**

Unit testing is a [software](http://searchsoa.techtarget.com/definition/software) development process in which the smallest testable parts of an[application](http://searchsoftwarequality.techtarget.com/definition/application), called units, are individually and independently scrutinized for proper operation. Unit testing is often automated but it can also be done manually. This testing mode is a component of [Extreme Programming](http://searchsoftwarequality.techtarget.com/definition/Extreme-Programming) (XP), a pragmatic method of software development that takes a meticulous approach to building a product by means of continual testing and revision.

1. **what is UAT?**

User acceptance testing (UAT) is the last phase of the software testing process. During UAT, actual software users test the software to make sure it can handle required tasks in real-world scenarios, according to specifications.

UAT is one of the final and critical software project procedures that must occur before newly developed software is rolled out to the market.

UAT is also known as beta testing, application testing or end user testing

1. **what is alpha and beta testing?**

**Alpha testing** – In house virtual user environment can be created for this type of testing. Testing is done at the end of development. Still minor design changes may be made as a result of such testing.

**Beta testing** – Testing typically done by end-users or others. Final testing before releasing application for commercial purpose.

1. **when do we use white box testing and block box testing?**

|  |  |  |
| --- | --- | --- |
| **#** | **Black Box Testing** | **White Box Testing** |
| 1 | Black box testing is the [Software testing method](http://www.softwaretestingclass.com/what-is-software-testing/) which is used to test the software without knowing the internal structure of code or program. | White box testing is the software testing method in which internal structure is being known to tester who is going to test the software. |
| 2 | This type of testing is carried out by testers. | Generally, this type of testing is carried out by software developers. |
| 3 | Implementation Knowledge is not required to carry out Black Box Testing. | Implementation Knowledge is required to carry out White Box Testing. |
| 4 | Programming Knowledge is not required to carry out Black Box Testing. | Programming Knowledge is required to carry out White Box Testing. |
| 5 | Testing is applicable on higher levels of testing like System Testing, Acceptance testing. | Testing is applicable on lower level of testing like Unit Testing, Integration testing. |
| 6 | Black box testing means functional test or external testing. | White box testing means structural test or interior testing. |
| 7 | In Black Box testing is primarily concentrate on the functionality of the system under test. | In White Box testing is primarily concentrate on the testing of program code of the system under test like code structure, branches, conditions, loops etc. |
| 8 | The main aim of this testing to check on what functionality is performing by the system under test. | The main aim of White Box testing to check on how System is performing. |
| 9 | Black Box testing can be started based on Requirement Specifications documents. | White Box testing can be started based on Detail Design documents. |
| 10 | The Functional testing, Behavior testing, Close box testing is carried out under Black Box testing, so there is no required of the programming knowledge. | The Structural testing, Logic testing, Path testing, Loop testing, Code coverage testing, Open box testing is carried out under White Box testing, so there is compulsory to know about programming knowledge. |

1. **what we will do if we don’t have a time to test all stories?**
2. **what we will do if come across any severity issue before release day?**
3. **when do we use automation testing?**

**Regression Testing**: For re-testing preexisting application functions that are being carried forward to new versions (usually the majority, unless app is brand new)

**Smoke Testing:** For getting a quick high-level assessment on the quality of a build and making go / no-go decision on deeper testing

**Static & Repetitive Tests:** For automating testing tasks that are repetitive and relatively unchanging from one test cycle to the next

**Data Driven Testing:** For testing application functions where the same functions needs to be validated with lots of different inputs & large data sets (i.e. login, search)

**Load & Performance Testing:** No viable manual alternative exists.

1. **what tester will do in each phase of SDLC?**

The Role of a Tester in SDLC  
  
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

1. **difference between load and performance testing?**

[**Performance Testing**](http://www.softwaretestinghelp.com/introduction-to-performance-testing-loadrunner-training-tutorial-part-1/)**:**Evaluates the overall performance of the system. Key elements are as follows:

–  Validate that the system meets the expected response time.  
–  Evaluate that the significant elements of the application meets the desired response time.  
–  It can also be conducted as a pert of integration testing.  
–  It can also be conducted as a part of systems testing.

[**Load Testing**](http://www.softwaretestinghelp.com/hp-loadrunner-load-testing-tool-training-tutorials/)**:** Evaluates whether the system’s performance is as expected under normal and expected conditions. Key points are

–  Validate that the system performs as expected when concurrent users access the application and gets the expected response time.

–  This test is repeated with multiple users to get the response time and throughput.  
–  At the time of testing, the data base should be realistic.  
–  The test should be conducted on a dedicated server which stimulates the actual environment.

1. **different types of non-functional testing types?**

## 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

1. **what is test case?**

A test case is a documentation which specifies input values, expected output and the preconditions for executing the test.

1. **what is test plan/test strategy document**

**Difference between Test Plan and Test Strategy**

**Test strategy** is a high level document which defines the approach for software testing. It is basically derived from the Business Requirement document. Test strategy is developed by project manager or business analyst. It is kind of static document which sets the standards for testing so not updated often.

**Test plan** is derived from SRS (*Software Requirement Specification*) which is prepared by test lead or manager. The main goal of test plan is to include all the details related to testing such as what to test, when to test, how to test and who will be the tester. Test plan is often not updated but if there is some new feature or change is introduced then it has to be updated accordingly.

Now, let’s make a list of points which are included in both respectively.

**Test strategy** contains:

1. **Scope and objective:** The objective of the business and how much testing scope is there is defined under test strategy.
2. **Business Issues:** How much is the budget of the project, how much time is required for testing, how much resources are needed etc. are the part of business issues which needs to be considered before the actual testing starts.
3. **Testing approach:** What type of testing is needed (performance, load, stress, functional etc.) and whether the testing is only manual or automation or both are some of the crucial points which defines the testing approach.
4. **Test deliverables:** What are the documents required from the testing team, how they would keep the record of the testing cycles etc. will be included here.
5. **Defect tracking approach**: Which tool will be used for tracking the defects and how will the testing team communicate with the development team and how the flow would go for defects are decided at this point in test strategy.
6. **Training**: If there is some complex or new tool is introduced in the business then it is helpful if the team members are given proper training. What type of training and the responsible person to conduct such training is defined here.
7. **Automation**: If the project or business needs automation testing then the script language, tool used, reporting and code maintained is planned in test strategy.
8. **Risks**: Nobody can anticipate all the risks beforehand but obvious risks can be avoided and also solution (if risk occur) can be included in the document for future help.

**Test plan** contains:

1. **Test plan ID:**This is a unique ID which defines the test plan. It can be a number or name or mix of both, as per the convenience.
2. **Test environment:**This section defines what kind of environment is needed for the testing to carry out. For e.g. in device testing, usually a virtual set up is made to test emergency calling.
3. **Features to be tested/Not tested:**This will have all the details about the features which tester needs to test and what are the feature which are not tested (may be because it is not yet implemented or not tested for that particular release).
4. **Entry/Exit criteria:** These are the terms which define when to start or stop the testing. Standards will be defined under test strategy and followed by testers in test plan.
5. **Status:**Whether a test case is passed or failed or not tested, all these test results are included in test plan with a proper reason.
6. **Types of testing:** The types of testing required such as regression, functional, non-functional, stress etc. are defined and then executed by the respective tester.
7. **Brief Intro:** Brief introduction is also included sometimes so that if any new member joins the team, he should get an idea how things work.
8. **what is TDD and BDD (cucumber framework)**

### ****Test Driven Development****

**Test-driven development** (**TDD**) is a software development process that relies on the repetition of a very short development cycle: first the developer writes an (initially failing) automated test case that defines a desired improvement or new function, then produces the minimum amount of code to pass that test, and finally refactors the new code to acceptable standards. Kent Beck, who is credited with having developed or ‘rediscovered’ the technique, stated in 2003 that TDD encourages simple designs and inspires confidence

### ****Behavior Driven Development****

In software engineering, **behavior-driven development** (abbreviated **BDD**) is a software development process based on test-driven development (TDD). Behavior-driven development combines the general techniques and principles of TDD with ideas from domain-driven design and object-oriented analysis and design to provide software development and management teams with shared tools and a shared process to collaborate on software development.

**Cucumber** is a tool based on Behavior Driven Development (BDD) framework which is used to write acceptance tests for web application. It allows automation of functional validation in easily readable and understandable format (like plain English) to Business Analysts, Developers, Testers, etc.

1. **what is priority and severity in defect?**

### What is Defect Priority?

Defect Priority states 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.

#### Defect priority can be categorized into three class

* **Low:**The defect is an irritant but repair can be done once the more serious defect have been fixed
* **Medium:**During the normal course of the development activities defect should be resolved. It can wait until a new version is created
* **High:**The defect must be resolved as soon as possible as it affects the system severely and cannot be used until it is fixed.

### What is Defect Severity?

In software testing, defect severity can be defined as the degree of impact a defect has on the development or operation of a component application being tested.

Higher effect on the system functionality will lead to the assignment of higher severity to the bug. Quality Assurance engineer usually determines the severity level of defect

#### Defect severity can be categorized into four class

* **Critical**: This defect indicates complete shut-down of the process, nothing can proceed further
* **Major**: It is a highly severe defect and collapse the system. However, certain parts of the system remain functional
* **Medium**: It cause some undesirable behavior, but the system is still functional
* **Low**: It won't cause any major break-down of the system

1. **how to estimate test cases?**
2. **what is most challenge defect u came across?**
3. **how to deal the production defects?**

normally end user will report this issue.

we need to talk to them and reproduce the issue with test logins

Create defect in defect tool under the production release version

developers will fix the issue

we (QA) test the issue on production version code and release the fix to proudction after we verify we have to create a defect on current sprint/release so that developer will add this code to the current sprint/release

1. **test design review steps**

* Test design is to ensure that all requirements are met through a series of test procedures, increasing the probability of the software being capable of what is needed and wanted by the client.
* Test design should start the moment the system requirements have been approved and baselined.
* The test design changes/adapts during the system development lifecycle’s iterations
* We must first take into consideration the testing phase in which the test will be executed, as there are different types of test appropriate for testing:
  + Security
  + Performance
  + Usability
  + Compatibility
  + Functionality
  + And other phases testing

1. **if we dont have time to test call test cases what we will do**

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 are 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/" \t "_blank) 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.

1. **how we learn the functionality of system?**
2. **what are the tools to manage defects/stories?**
3. **who will assign the work?**
4. **types of test metrics we use normally**

In software testing, **Metric** is a quantitative measure of the degree to which a **system, system component, or process**possesses a given attribute.

In other words, metrics helps estimating the progress, quality and health of a software testing effort. The ideal example to understand metrics would be a weekly mileage of a car compared to its ideal mileage recommended by the manufacturer.

**Type of metrics**

**Base Metrics (Direct Measure)**   
  
Base metrics constitute the raw data gathered by a Test Analyst throughout the testing effort. These metrics are used to provide project status reports to the Test Lead and Project Manager; they also feed into the formulas used to derive Calculated Metrics.   
Ex: # of Test Cases, # of Test Cases Executed   
 **Calculated Metrics (Indirect Measure)**   
  
Calculated Metrics convert the Base Metrics data into more useful information. These types of metrics are generally the responsibility of the Test Lead and can be tracked at many different levels (by module, tester, or project).   
Ex: % Complete, % Test Coverage   
  
  
**Base Metrics & Test Phases**

* • # of Test Cases (Test Development Phase)
* • # of Test Cases Executed (Test Execution Phase)
* • # of Test Cases Passed (Test Execution Phase)
* • # of Test Cases Failed (Test Execution Phase)
* • # of Test Cases Under Investigation (Test Development Phase)
* • # of Test Cases Blocked (Test dev/execution Phase)
* • # of Test Cases Re-executed (Regression Phase)
* • # of First Run Failures (Test Execution Phase)
* • Total Executions (Test Reporting Phase)
* • Total Passes (Test Reporting Phase)
* • Total Failures (Test Reporting Phase)
* • Test Case Execution Time ((Test Reporting Phase)
* • Test Execution Time (Test Reporting Phase

**Calculated Metrics & Phases**   
The below metrics are created at Test Reporting Phase or Post test Analysis phase 

* • % Complete
* • % Defects Corrected
* • % Test Coverage
* • % Rework
* • % Test Cases Passed
* • % Test Effectiveness
* • % Test Cases Blocked
* • % Test Efficiency
* • 1st Run Fail Rate
* • Defect Discovery Rate
* • Overall Fail Rate

**III) Crucial Web Based Testing Metrics**   
**Test Plan coverage on Functionality**   
Total number of requirement v/s number of requirements covered through test scripts. 

* • (No of requirements covered / total number of requirements) \* 100

Define requirements at the time of Effort estimation   
Example: Total number of requirements estimated are 46, total number of requirements tested 39; blocked 7…define what is the coverage?   
Note: Define requirement clearly at project level   
 **Test Case defect density**   
  
Total number of errors found in test scripts v/s developed and executed. 

* • (Defective Test Scripts /Total Test Scripts) \* 100

Example: Total test script developed 1360, total test script executed 1280, total test script passed 1065, total test script failed 215   
So, test case defect density is   
215 X 100   
---------------------------- = 16.8%   
1280   
This 16.8% value can also be called as test case efficiency %, which is depends upon total number of test cases which uncovered defects   
  
  
 **Defect Slippage Ratio**   
Number of defects slipped (reported from production) v/s number of defects reported during execution. 

* • Number of Defects Slipped / (Number of Defects Raised - Number of Defects Withdrawn)

Example: Customer filed defects are 21, total defect found while testing are 267, total number of invalid defects are 17   
So, Slippage Ratio is   
[21/ (267-17)] X 100 = 8.4%   
  
**Requirement Volatility**   
  
Number of requirements agreed v/s number of requirements changed. 

* • (Number of Requirements Added + Deleted + Modified) \*100 / Number of Original Requirements
* • Ensure that the requirements are normalized or defined properly while estimating

Example: VSS 1.3 release had total 67 requirements initially, later they added another 7 new requirements and removed 3 from initial requirements and modified 11 requirements   
So, requirement Volatility is   
(7 + 3 + 11) \* 100/67 = 31.34%   
Means almost 1/3 of the requirement changed after initial identification   
  
**Review Efficiency**  
The Review Efficiency is a metric that offers insight on the review quality and testing   
Some organization also use this term as “Static Testing” efficiency and they are aiming to get min of 30% defects in static testing   
Review efficiency=100\*Total number of defects found by reviews/Total number of project defects   
Example: A project found total 269 defects in different reviews, which were fixed and test team got 476 defects which were reported and valid   
So, Review efficiency is [269/(269+476)] X 100 = 36.1%   
  
**Efficiency and Effectiveness of Processes** 

* • **Effectiveness:** Doing the right thing. It deals with meeting the desirable attributes that are expected by the customer.
* • **Efficiency:** Doing the thing right. It concerns the resources used for the service to be rendered

**Metrics for Software Testing**   
  
**• Defect Removal Effectiveness**  
DRE= (Defects removed during development phase x100%) / Defects latent in the product   
Defects latent in the product = Defects removed during development   
Phase+ defects found later by user   
  
**• Efficiency of Testing Process (define size in KLoC or FP, Req.)**  
Testing Efficiency= Size of Software Tested /Resources used

1. **what is traceability matrix?**

A traceability matrix is a document that co-relates any two-baseline documents that require a many-to-many relationship to check the completeness of the relationship. It is used to track the requirements and to check the current project requirements are met.

1. **what are typical environments we have in projects**
2. **what is development environment**

**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.

1. **what is QA environment**

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.

1. **what is production environment**

A **Production environment** is where the Waveset application is actually available for business use.

1. **what are different defect metrics and measurements we prepare**
2. **what are weakness and strong points**
3. **What is staging environment**

In software, a staging environment is used to test out newer versions of software before it is moved live – into production.

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.