**Why testing is required?**

Testing is required for an **effective performance** of software application or product. It’s important to ensure that the application should not result into any failures because it can be very **expensive** in the **future** or in the later stages of the development. It’s required to stay in the business. Software testing is really required to point out the defects and **errors** that were made during the **development phases**.

**What types of application we test?**

Web based, desktop, mobile.

**What is SDLC and different phases in SDLC?**

Ans: Software development life cycle (SDLC) is a process to develop the application

**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

**What is waterfall in SDLC?**

The **waterfall** model is a **sequential** (non-iterative) design process, used in software development processes, in which progress is seen as flowing steadily downwards (like a **waterfall**) through the phases of conception, initiation, analysis, design, construction, testing, production/implementation and maintenance.

**What is the process in agile model?**

Agile Methods break the product into small incremental builds. These builds are provided in iterations.

**What is scrum methodology?**

Scrum is not a methodology. It simply provides structure, discipline and a framework for agile development.

In the SCRUM methodology a **sprint** is the **basic unit of development**. Each sprint starts with a **planning meeting**, where the **tasks** for the sprint are **identified** and an **estimated** commitment for the sprint goal is made. A Sprint ends with a review or retrospective meeting where the progress is reviewed and lessons for the next sprint are identified. During each sprint, the team creates finished portions of a product.

What did you do yesterday?

What will you do today?

Are there any challenges in your way?

In the Agile methods each iteration involves a team working through a full software development cycle, including planning, requirements analysis, design, coding, unit testing, and acceptance testing when a working product is demonstrated to stakeholders.

**What is daily standup meeting and what we discuss?**

A daily stand-up meeting is a short organizational meeting that is held each day. The meeting, generally limited to between five and fifteen minutes long, is sometimes referred to as a stand-up, a morning roll-call or a daily scrum.

Member talks about progress since the last stand-up, the anticipated work until the next stand-up and any impediments, taking the opportunity to ask for help or collaborate

Also referred to as the "stand-up", "morning rollcall" or "daily scrum".

What did you do yesterday?

What will you do today?

Are there any challenges in your way?

**What is user story/feature/sprint back log items and tasks in user story?**

A **user story** is a tool used in agile software development to capture a description of a software feature from an end-**user** perspective. The **user story** describes the type of **user**, what they want and why. A **user story** helps to create a simplified description of a requirement.

The sprint backlog is a list of tasks identified by the Scrum team to be completed during the Scrum sprint. During the sprint planning meeting, the team selects some number of product backlog items, usually in the form of user stories, and identifies the tasks necessary to complete each user story. Most teams also estimate how many hours each task will take someone on the team to complete.

It's critical that the team selects the items and size of the sprint backlog. Because they are the people committing to completing the tasks, they must be the people to choose what they are committing to during the Scrum sprint.

The sprint backlog is commonly maintained as a spreadsheet, but it is also possible to use your defect tracking system or any of a number of software products designed specifically for Scrum or agile.

**what is sprint planning and spring retro**

**Sprint planning** is a collaborative effort involving a Scrum Master, who facilitates the meeting, a Product Owner, who clarifies the details of the product backlog items and their respective acceptance criteria, and the Entire Agile Team, who define the work and effort necessary to meet their **sprint** commitment

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

**What is burndown chart and velocity**

Its purpose is to enable that the project is on the track to deliver the expected solution within the desired schedule. Simple **Burndown Chart**. The rate of progress of a Scrum Team is called "**velocity**". It expresses the amount of e.g. story points completed per iteration.



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

In **Scrum**, a **product backlog item** ("PBI", "**backlog item**", or "**item**") is a unit of work small enough to be completed by a team in one **Sprint** iteration. **Backlog items** are decomposed into one or more tasks. See also **backlog** effort estimation unit

**Product Backlog**-All work items related to a product/project, ordered by a Product Owner.

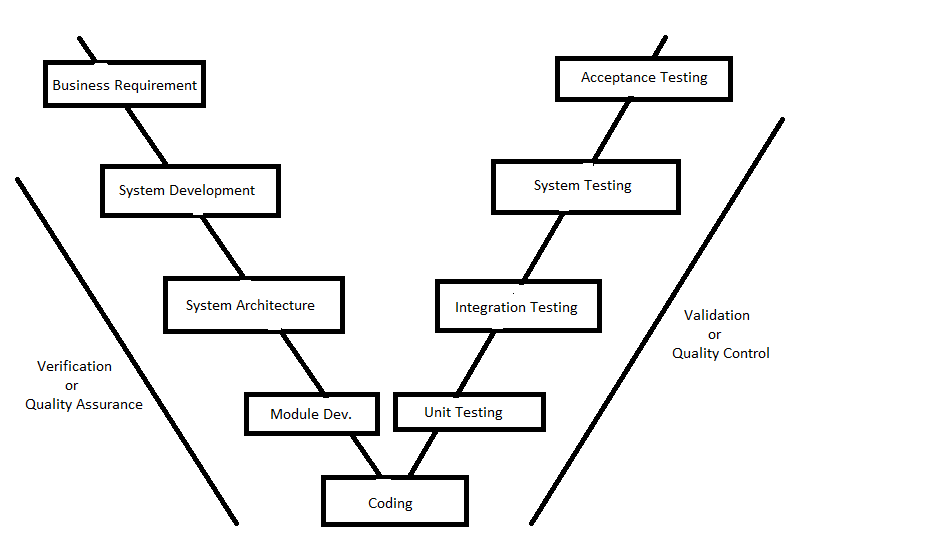
**Sprint Backlog** => All work committed to and pushed into a Development Teams's upcoming Sprint/iteration as chosen by the Development Team in their Sprint Planning meeting.

**what is user acceptance criteria test cases**

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.

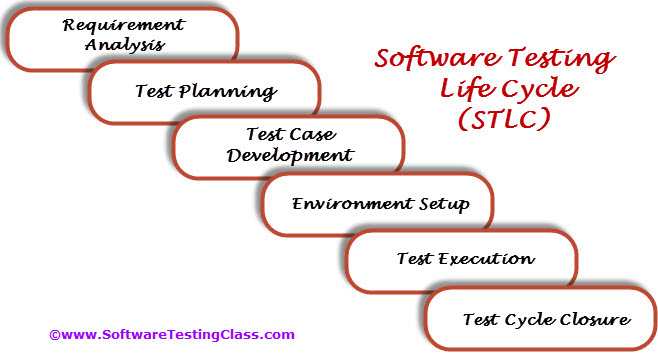
**what is v model?**

The **V** - **model** is SDLC **model** where execution of processes happens in a sequential manner in **V**-shape. It is also known as Verification and Validation **model**. **V** - **Model** is an extension of the waterfall **model** and is based on association of a testing phase for each corresponding development stage.



**what is STLC?**

Software Testing Life Cycle (**STLC**) is the testing process which is executed in systematic and planned manner. In **STLC** process, different activities are carried out to improve the quality of the product.



**what is defect?**

 defect is an error or a bug, in the application which is created

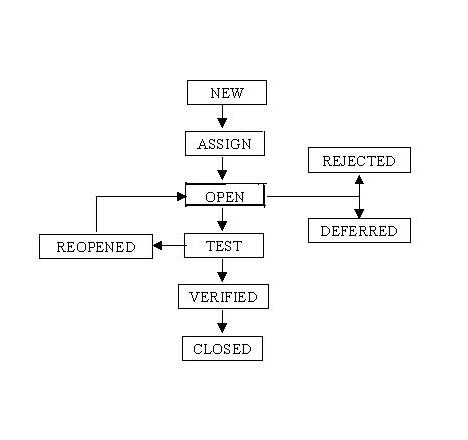
**how to arise a defect and what we specify while logging defect?**

**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 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.

****

**Different types of testing?**

**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.

**Smoke testing**- Smoke testing, in the context of software development, is a series of test cases that are run before the commencement of more rigorous tests. The goal of smoke testing is to verify that an application's main features work properly. A smoke test suite can be automated or a combination of manual and automated testing.

**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.

**When do we use regression testing?**

When a **bug is fixed** by the development team then 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.

**When do we use integration testing?**

We normally do Integration testing after “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.

There are fundamentally 2 approaches for doing Integration testing:

1. Bottom up approach
2. Top down approach.

**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.

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.

**What is unit testing?**

**Unit testing** is a software development process in which the smallest testable parts of an application, called units, are individually and independently scrutinized for proper operation. **Unit testing** is often automated but it can also be done manually.

**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.

**What is alpha and beta testing?**

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

**what we will do if we don’t have a time to test all stories?**

**what we will do if come across any severity issue before release day?**

**When do we use automation testing?**

Manually repeating these **tests** is costly and time consuming. Once created,**automated tests** can be run over and over again at no additional cost and they are much faster than manual **tests**. **Automated** software **testing** can reduce the time to run repetitive **tests** from days to hours.

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

**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 part 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.

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

* Load/Performance testing.
* Compatibility testing.
* Localization testing.
* Security testing.
* Reliability testing.
* Stress testing.
* Usability testing.
* Compliance testing.

**what is test case?**

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

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

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:

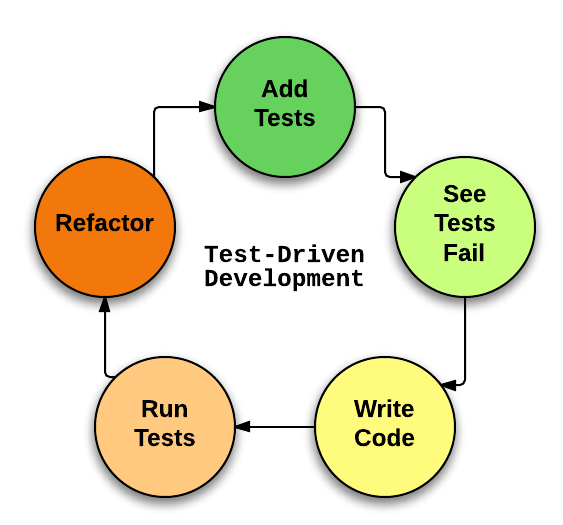
* Scope and objective: The objective of the business and how much testing scope is there is defined under test strategy.
* Business Issues: How much is the budget of the project, how much time is required for testing, how much resources are needed etc. are the part of business issues which needs to be considered before the actual testing starts.
* 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.
* 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.
* 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.
* 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.
* Automation: If the project or business needs automation testing then the script language, tool used, reporting and code maintained is planned in test strategy.
* 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:

* 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.
* 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.
* 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).
* 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.
* 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.
* 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.
* 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.

**What is TDD and BDD (cucumber framework?)**

**Test-driven development** (**TDD**) is a software development process that relies on the repetition of a very short development cycle: requirements are turned into very specific test cases, then the software is improved to pass the new tests, only.



Behavior Driven testing is an extension of TDD. Like in TDD in BDD also we write tests first and the add application code. The major difference that we get to see here are

* Tests are written in plain descriptive English type grammar
* Tests are explained as behavior of application and are more user focused
* Using examples to clarify requirements

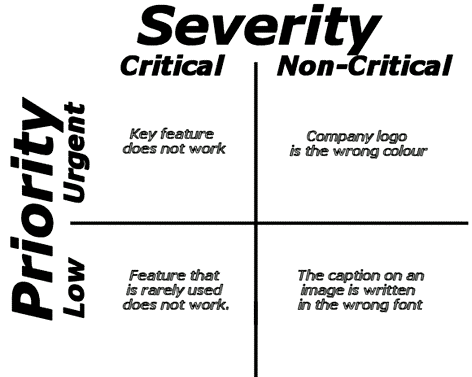
This difference brings in the need to have a language which can define, in an understandable format.

Features of BDD

1. Shifting from thinking in “tests” to thinking in “behavior”
2. Collaboration between Business stakeholders, Business Analysts, QA Team and developers
3. Ubiquitous language, it is easy to describe
4. Driven by Business Value
5. Extends Test Driven Development (TDD) by utilizing natural language that non technical stakeholders can understand
6. BDD frameworks such as Cucumber or JBehave are an enabler, acting a “bridge” between Business & Technical Language

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

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.



**How to estimate test cases?**

**what is most challenge defect u came across?**

**how to deal the production defects?**

Ans: 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

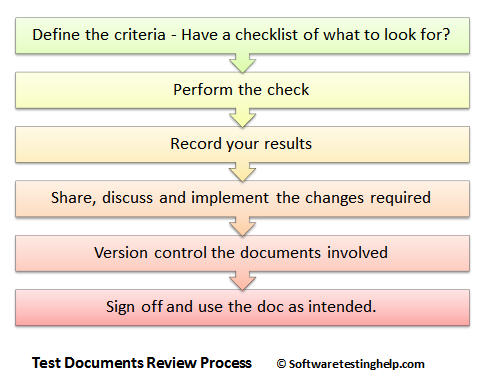
we have to create a defect on current sprint/release so that developer will add this code to the current sprint/release

**test design review steps**

1. Test plan
2. Test scenarios
3. Test templates
4. Test cases
5. Test data
6. Reports…etc

**How to review? –**The following are the list of activities involved:

1. Define the criteria – Have a checklist of what to look for?
2. Perform the check
3. Record your results
4. Share, discuss and implement the changes required
5. Version control the documents involved
6. Sign off and use the doc as intended.



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

**how we learn the functionality of system?**

**What are the tools to manage defects/stories?**

Bugzilla, JIRA, HP ALM/QUALITY CENTER

**Who will assign the work?**

**Types of test metrics we use normally**

Generation of Software Test Metrics is the most important responsibility of the Software Test Lead/Manager.

Test Metrics are used to,

Take the decision for next phase of activities such as, estimate the cost & schedule of future projects.

Understand the kind of improvement required to success the project

Take decision on process or technology to be modified etc.

Importance of Software Testing Metrics:

As explained above, Test Metrics are the most important to measure the quality of the software.

Now, how can we measure the quality of the software by using Metrics?

Suppose, if a project does not have any metrics, then how the quality of the work done by a Test analyst will be measured?

### Conclusion:

The metrics provided in this article are majorly used for generating the [daily/weekly status report](http://www.softwaretestinghelp.com/how-to-write-software-testing-weekly-status-report/) with accurate data during test case development/execution phase & this is also useful for tracking the project status & Quality of the software.

**what is traceability matrix?**

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.

**What are typical environments we have in projects?**

**What is development environment?**

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.

**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 Wave set application.

**what is production environment**

A Production environment is where the Wave set application is actually available for business use.

**what are different defect metrics and measurements we prepare**

**what are weakness and strong points**

**What is staging environment**

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.