QA TASKS

**Topic**: Manual Testing

**Date**: 29-01-2017

**Questions:**

1. **Why testing is required?**

It is required because of the following reasons:

* At first, we can find the bugs/defects that occur during the development phases.
* Validate and verify whether the software is satisfying all the requirements of the client.
* For improving the quality of the product so that we can satisfy the customer.

1. **What type of applications we test?**

There are different types of applications:

1. **Web applications**: Which are accessed through web browsers over the internet.

Example: facebook, gmail, any other sites.

1. **Desktop/windows applications** (we run from our desktop, ex: notepad, word)
2. **Mobile applications** (we run from mobile devices, android, ios, windows phone)
3. **Web services (SOAP/REST)** (we use them as part of web applications / desktop applications/mobile applications)
4. **ETL jobs, database validations**: these jobs don’t have UI and they run in back ground to load the data (informatica jobs, ssis job etc) and also Back end/batch programs/windows services.
5. **What is SDLC?**

**SDLC** is a process followed for a software project, within a software organization. It consists of a detailed plan describing how to develop, maintain, replace and alter or enhance specific software. The life cycle defines a methodology for improving the quality of software and the overall development process.



**Planning and Requirement Analysis**

Requirement analysis is the most important and fundamental stage in SDLC. It is performed by the senior members of the team with inputs from the customer, the sales department, market surveys and domain experts in the industry. This information is then used to plan the basic project approach and to conduct product feasibility study in the economical, operational, and technical areas.

Planning for the quality assurance requirements and identification of the risks associated with the project is also done in the planning stage. The outcome of the technical feasibility study is to define the various technical approaches that can be followed to implement the project successfully with minimum risks.

**Defining Requirements**

Once the requirement analysis is done the next step is to clearly define and document the product requirements and get them approved from the customer or the market analysts. This is done through **SRS** (Software Requirement Specification) document which consists of all the product requirements to be designed and developed during the project life cycle.

**Designing the product architecture**

SRS is the reference for product architects to come out with the best architecture for the product to be developed. Based on the requirements specified in SRS, usually more than one design approach for the product architecture is proposed and documented in a DDS - Design Document Specification.

This DDS is reviewed by all the important stakeholders and based on various parameters as risk assessment, product robustness, design modularity, budget and time constraints , the best design approach is selected for the product.

A design approach clearly defines all the architectural modules of the product along with its communication and data flow representation with the external and third party modules (if any). The internal design of all the modules of the proposed architecture should be clearly defined with the minutest of the details in DDS.

**Building or Developing the Product**

In this stage of SDLC the actual development starts and the product is built. The programming code is generated as per DDS during this stage. If the design is performed in a detailed and organized manner, code generation can be accomplished without much hassle.

Developers have to follow the coding guidelines defined by their organization and programming tools like compilers, interpreters, debuggers etc are used to generate the code. Different high level programming languages such as C, C++, Pascal, Java, and PHP are used for coding. The programming language is chosen with respect to the type of software being developed.

**Testing the Product**

This stage is usually a subset of all the stages as in the modern SDLC models, the testing activities are mostly involved in all the stages of SDLC. However this stage refers to the testing only stage of the product where products defects are reported, tracked, fixed and retested, until the product reaches the quality standards defined in the SRS.

**Deployment in the Market and Maintenance**

Once the product is tested and ready to be deployed it is released formally in the appropriate market. Sometime product deployment happens in stages as per the organizations. business strategy. The product may first be released in a limited segment and tested in the real business environment (UAT- User acceptance testing). Then based on the feedback, the product may be released as it is or with suggested enhancements in the targeting market segment. After the product is released in the market, its maintenance is done for the existing customer base.

**SDLC Models**

There are various software development life cycle models defined and designed which are followed during software development process. These models are also referred as "Software Development Process Models". Each process model follows a Series of steps unique to its type, in order to ensure success in process of software development.

Waterfall Model, Iterative Model, Spiral Model, V-Model, Big Bang Model

Agile Model, RAD Model, Rapid Application Development and Prototyping Models.

1. **What is Waterfall model?**

Waterfall approach was first SDLC Model to be used widely in Software Engineering to ensure success of the project. 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.



**Requirement Gathering and analysis:** All possible requirements of the system to be developed are captured in this phase and documented in a requirement specification doc.

**System Design:** The requirement specifications from first phase are studied in this phase and system design is prepared. System Design helps in specifying hardware and system requirements and also helps in defining overall system architecture.

**Implementation:** With inputs from system design, the system is first developed in small programs called units, which are integrated in the next phase. Each unit is developed and tested for its functionality which is referred to as **Unit Testing**.

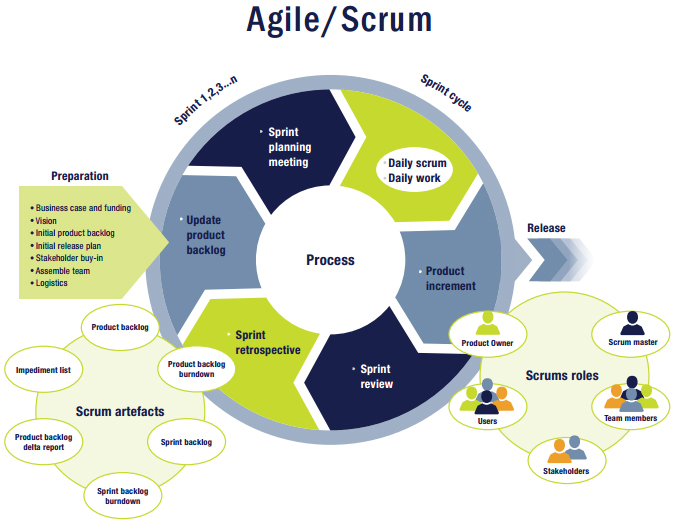
**Integration and Testing:** All the units developed in the implementation phase are integrated into a system after testing of each unit. Post integration the entire system is tested for any faults and failures.

**Deployment of system:** Once the functional and non-functional testing is done, the product is deployed in the customer environment or released into the market.

**Maintenance:** There are some issues which come up in the client environment. To fix those issues patches are released. Also to enhance the product some better versions are released. Maintenance is done to deliver these changes in the customer environment.

1. **What is Agile model?**

Not a methodology! The Agile movement seeks alternatives to traditional project management. Agile approaches help teams respond to unpredictability through incremental, iterative work cadences and empirical feedback. Agilists propose alternatives to waterfall, or traditional sequential development.



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.

A “process framework” is a particular set of practices that must be followed in order for a process to be consistent with the framework. (For example, the Scrum process framework requires the use of development cycles called Sprints, the XP framework requires pair programming, and so forth.)

“Lightweight” means that the overhead of the process is kept as small as possible, to maximize the amount of productive time available for getting useful work done.

1. **What is the process in Agile model?**

Agile model believes that every project needs to be handled differently and the existing methods need to be tailored to best suit the project requirements. In agile the tasks are divided to time boxes (small time frames) to deliver specific features for a release.



**Following are the principles:**

**Individuals and interactions** - in agile development, self-organization and motivation are important, as are interactions like co-location and pair programming.

**Working software** - Demo working software is considered the best means of communication with the customer to understand their requirement, instead of just depending on documentation.

**Customer collaboration** - As the requirements cannot be gathered completely in the beginning of the project due to various factors, continuous customer interaction is very important to get proper product requirements.

**Responding to change** - agile development is focused on quick responses to change and continuous development.

1. **What are daily standup meetings 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. 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 (obstacles) in your way?

Standing, rather than sitting, reinforces the idea that the meeting is intended to be short and discourages wasted time. The stand-up is not meant to be a place to solve problems, but rather to make the team aware of current status. If discussion is needed, a longer meeting with appropriate parties can be arranged.

1. **What is product back log item?**

A **product backlog** is a prioritized list of work for the development team that is derived from the roadmap and its requirements. The most important items are shown at the top of the product backlog so the team knows what to deliver first. The development team doesn't work through the backlog at the product owner's pace and the product owner isn't pushing work to the development team. Instead, the development team pulls work from the product backlog and it mostly contains the user stories and requirements of the project.

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

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.

1. **What is sprint planning meeting?**

Sprint planning is a collaborative effort involving a ScrumMaster, 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.

1. **What is sprint review meeting?**

During the sprint review, the project is assessed against the sprint goal determined during the sprint planning meeting. Ideally, the team has completed each product backlog item brought into the sprint, but it's more important that they achieve the overall goal of the sprint.

1. **What is sprint retrospective?**

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.

1. **What is sprint grooming?**

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

1. **What is burndown chart and velocity?**

**Burndown Chart**: 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.

1. **What is user acceptance criteria test cases?**

Acceptance Criteria are the conditions that a software product must satisfy to be accepted by a user, customer, or in the case of system level functionality, the consuming system.

1. **What is V model?**

V - Model is an extension of the waterfall model and is based on association of a testing phase for each corresponding development stage. This means that for every single phase in the development cycle there is a directly associated testing phase.



**Verification Phases**

* **Business Requirement Analysis:** This is the first phase in the development cycle where the product requirements are understood from the customer perspective. This phase involves detailed communication with the customer to understand his expectations and exact requirement. This is a very important activity and need to be managed well, as most of the customers are not sure about what exactly they need. The acceptance test design planning is done at this stage as business requirements can be used as an input for acceptance testing.
* **System Design:** Once you have the clear and detailed product requirements, it’s time to design the complete system. System design would comprise of understanding and detailing the complete hardware and communication setup for the product under development. System test plan is developed based on the system design. Doing this at an earlier stage leaves more time for actual test execution later.
* **Architectural Design:** Architectural specifications are understood and designed in this phase. Usually more than one technical approach is proposed and based on the technical and financial feasibility the final decision is taken. System design is broken down further into modules taking up different functionality. This is also referred to as High Level Design (HLD).
* **Module Design:** In this phase the detailed internal design for all the system modules is specified, referred to as Low Level Design (LLD). It is important that the design is compatible with the other modules in the system architecture and the other external systems. Unit tests are an essential part of any development process and helps eliminate the maximum faults and errors at a very early stage. Unit tests can be designed at this stage based on the internal module designs.
* **Unit Testing:** Unit tests designed in the module design phase are executed on the code during this validation phase. Unit testing is the testing at code level and helps eliminate bugs at an early stage, though all defects cannot be uncovered by unit testing.
* **Integration Testing:** Integration testing is associated with the architectural design phase. Integration tests are performed to test the coexistence and communication of the internal modules within the system.
* **System Testing:** System testing is directly associated with the System design phase. System tests check the entire system functionality and the communication of the system under development with external systems. Most of the software and hardware compatibility issues can be uncovered during system test execution.
* **Acceptance Testing:** Acceptance testing is associated with the business requirement analysis phase and involves testing the product in user environment. Acceptance tests uncover the compatibility issues with the other systems available in the user environment. It also discovers the non-functional issues such as load and performance defects in the actual user environment.

1. **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. Let’s quickly see what all stages are involved in typical Software Testing Life Cycle (STLC).

Following steps are involved in Software Testing Life Cycle (STLC). Each step is have its own Entry Criteria and deliverable.

1. Requirement Analysis
2. Test Planning
3. Test Case Development
4. Environment Setup
5. Test Execution
6. Test Cycle Closure



Each of these stages have a definite Entry and Exit criteria. Activities & Deliverables associated with it.

**Entry Criteria:** Entry Criteria gives the prerequisite items that must be completed before testing can begin.

**Exit Criteria:** Exit Criteria defines the items that must be completed before testing can be concluded

You have Entry and Exit Criteria for all levels in the Software Testing Life Cycle (STLC)

**Requirement Analysis**

During this phase, test team studies the requirements from a testing point of view to identify the testable requirements. The QA team may interact with various stakeholders (Client, Business Analyst, Technical Leads, System Architects etc) to understand the requirements in detail. Requirements could be either Functional (defining what the software must do) or Non Functional (defining system performance /security availability). Automation feasibility for the given testing project is also done in this stage.

**Activities**

* Identify types of tests to be performed.
* Gather details about testing priorities and focus.
* Prepare [Requirement Traceability Matrix (RTM)](http://www.guru99.com/traceability-matrix.html).
* Identify test environment details where testing is supposed to be carried out.
* Automation feasibility analysis (if required)

**Deliverables**

* RTM
* Automation feasibility report. (if applicable)

### **Test Planning**

This phase is also called **Test Strategy** phase. Typically, in this stage, a Senior QA manager will determine effort and cost estimates for the project and would prepare and finalize the Test Plan.

**Activities**

* Preparation of test plan/strategy document for various types of testing
* Test tool selection
* Test effort estimation
* Resource planning and determining roles and responsibilities.
* Training requirement

**Deliverables**

* [Test plan](http://www.guru99.com/test-plan.html) /strategy document.
* [Effort estimation](http://www.guru99.com/testing-estimation.html) document.

### **Test Case Development**

This phase involves creation, verification and rework of test cases & test scripts. [Test data](http://www.guru99.com/software-testing-test-data.html) , is identified/created and is reviewed and then reworked as well.

**Activities**

* Create test cases, automation scripts (if applicable)
* Review and baseline test cases and scripts
* Create test data (If Test Environment is available)

**Deliverables**

* Test cases/scripts
* Test data

### **Test Environment Setup**

Test environment decides the software and hardware conditions under which a work product is tested. Test environment set-up is one of the critical aspects of testing process and **can be done in parallel with Test Case Development Stage. Test team may not be involved in this activity** if the customer/development team provides the test environment in which case the test team is required to do a readiness check (smoke testing) of the given environment.

**Activities**

* Understand the required architecture, environment set-up and prepare hardware and software requirement list for the Test Environment.
* Setup test Environment and test data
* Perform smoke test on the build

**Deliverables**

* Environment ready with test data set up
* Smoke Test Results.

### **Test Execution**

 During this phase test team will carry out the testing based on the test plans and the test cases prepared. Bugs will be reported back to the development team for correction and retesting will be performed.

**Activities**

* Execute tests as per plan
* Document test results, and log defects for failed cases
* Map defects to test cases in RTM
* Retest the defect fixes
* Track the defects to closure

**Deliverables**

* Completed RTM with execution status
* Test cases updated with results
* Defect reports

### **Test Cycle Closure**

Testing team will meet, discuss and analyze testing artifacts to identify strategies that have to be implemented in future, taking lessons from the current test cycle. The idea is to remove the process bottlenecks for future test cycles and share best practices for any similar projects in future.

**Activities**

* Evaluate cycle completion criteria based on Time, Test coverage, Cost, Software, Critical Business Objectives, Quality
* Prepare test metrics based on the above parameters.
* Document the learning out of the project
* Prepare Test closure report
* Qualitative and quantitative reporting of quality of the work product to the customer.
* Test result analysis to find out the defect distribution by type and severity.

**Deliverables**

* Test Closure report
* Test metrics

1. **What is Defect?**

A defect is an **error** or **bug** in the application and it is found when a programmer observes some flaws in the software/ hardware while designing and building the application.

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

Basically, a defect will arise in certain conditions, For instance,

* Errors in use of the system
* Environmental conditions
* Intentional damage
* Potential consequences of earlier errors

**Logging defect:**

Defect logging, a process of finding defects in the application under test or product by testing or recording feedback from customers and making new versions of the product that fix the defects or the clients feedback. We usually specify the following fields while logging/tracking defect:

**Defect Id** - Lists each defect uniquely

**Priority** - Assigning priority to a defect on the basis of the degree of risk involved

**Severity** - Determines to what extent the defect can affect the system

**Created By** - The name of the person who has detected the defect

**Created Date** - The date on which the defect was captured

**Assigned to** - The person to whom the defect is reported for resolving it

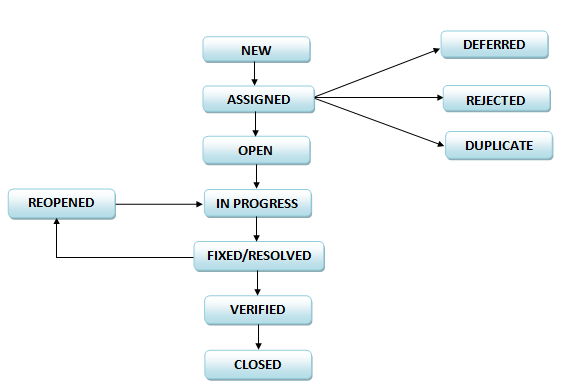
**Resolved Date** - The date on which the defect is resolved

**Resolved By** - The team/person who resolved the defect

**Status** - Defect fixed or postponed (based on the severity of the defect)

1. **What is Defect Life Cycle?**

* **Defect Life Cycle:**

It’s a journey where defect goes during its lifetime till it is fixed and its different from project to project and organization to organization and it is controlled by software testing process and also what tools it was used. 

**New** - When a Tester finds a defect then he/she logs it into a bug tracking tool with status New. In some cases a developer can also report a defect found during development phase.

**Assigned** - Project lead then assigns the New defect to the developer and changes its status to Assigned.

**Rejected** - Respective developer analyses the Assigned defect and may change its status to Rejected if he feels that the defect is not genuine or Works as per design or if it is not reproducible in development environment or not enough/clear information is provided by tester along with defect to reproduce it and if such is a case then Developer must provide the reason of rejection with necessary details. Tester rechecks the Rejected Defect and may change its status back to Assigned by providing more information or he may change it to Closed if he agrees with the reason of rejection.

**Deferred** - Developer may set status of Assigned defect as Deferred if it is not going to be fixed in current iteration. There are several factors because of which a defect could be deferred/postponed. For example, the defect is low priority, has no major impact, need more information from customer/product owner for fixing it or there is not enough time to fix it in current iteration etc.

**Duplicate -** If developer finds that Assigned defect is already reported in system and its not yet closed he changes its status to Duplicate. While marking a defect as Duplicate, developer should specify the **Defect ID** of its duplicate. Tester reviews the Duplicate defect and may set its status back to Assigned if he confirms that it is not a duplicate. Sometimes it happens that two issues look identical due to the error it produces but occurs in different scenarios or the cause of each is different. And the information provided in both the defects does not clearly differentiate them.

**Open/In Progress** - If developer agrees that Assigned defect is genuine he starts working on its resolution and marks the defect as Open. In some organizations In Progress status is used instead of Open.

**Fixed/Resolved** - Once developer Fixes the defect and Unit tests the code, he changes its status to Fixed/Resolved. This status indicates that the defect is fixed and ready to test. Developer must mention the **build number** in which the fix is planned to deliver to testing team.

**Re-opened**– Once the Fixed/Resolved defect is delivered to testing team, tester tests it and if the defect is still reproducible then he sets the defect status as Re-opened. Tester should provide the reason of re-opening the defect with necessary artifacts like screenshots, error log etc. as applicable. When Reopened, developer revisits the defect and resolves it changes its status to Fixed/Resolved.

**Verified**– After testing the Fixed/Resolved defect Tester confirms that it is working as expected by changing defect status to Verified.

**Closed**– Tester marks a defect as Closed when he confirms that the issue no longer exists in the software. Sometimes it happens that a Closed defect is reproducible again. In such cases, Tester Re-opens the defect.

1. **What is Unit testing?**

**Unit Testing** is a level of software **testing** where individual units/ components of a software are independently tested. A **unit** is the smallest testable part of software. It usually has one or a few inputs and usually a single output. **Unit testing** can be done manually but is often automated.

1. **When do we use Regression testing?**

**Regression testing** is the process of **testing** changes to computer programs to make sure that the older programming still works with the new changes. **Regression testing** is a normal part of the program development process and, in larger companies, is done by code **testing** specialists. Regression testing is nothing but full or partial selection of already executed test cases which are re-executed to ensure existing functionalities work fine.

Regression Testing is required when there is a

* Change in requirements and code is modified according to the requirement
* New feature is added to the software
* Defect fixing
* Performance issue fix



### **Retest All**

* This is one of the methods for regression testing in which all the tests in the existing test bucket or suite should be re-executed. This is very expensive as it requires huge time and resources.

**Regression Test Selection**

* Instead of re-executing the entire test suite, it is better to select part of test suite to be run
* Test cases selected can be categorized as 1) Reusable Test Cases 2) Obsolete Test Cases.
* Re-usable Test cases can be used in succeeding regression cycles.
* Obsolete Test Cases can't be used in succeeding cycles.

**Prioritization of Test Cases**

* Prioritize the test cases depending on business impact, critical & frequently used functionalities. Selection of test cases based on priority will greatly reduce the regression test suite.

Important tools used for both functional and regression testing:

* [**Selenium**](http://www.guru99.com/selenium-tutorial.html)**:** This is an open source tool used for automating web applications. Selenium can be used for browser based regression testing.
* [**Quick Test Professional (QTP)**](http://www.guru99.com/quick-test-professional-qtp-tutorial.html)**:** HP Quick Test Professional is automated software designed to automate functional and regression test cases. It uses VbScript language for automation. It is a Data driven, Keyword based tool.
* **Rational Functional Tester (RFT):** IBM's rational functional tester is a java tool used to automate the test cases of software applications. This is primarily used for automating regression test cases and it also integrates with Rational Test Manager.

1. **What is Integration testing?**

**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**.

1. **When do we use Integration testing?**

Integration Testing is performed after [Unit Testing](http://softwaretestingfundamentals.com/unit-testing/) and before [System Testing](http://softwaretestingfundamentals.com/system-testing/).

1. **When do we use Smoke and Sanity testing?**

**Smoke Testing** is a kind of Software Testing 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** is a kind of Software Testing performed after receiving a software build, with minor changes in code, or functionality, 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.

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:** It is a type of acceptance testing; performed to identify all possible issues/bugs before releasing the product to everyday users or public.  The focus of this testing is to simulate real users by using blackbox and whitebox techniques. The aim is to carry out the tasks that a typical user might perform. Alpha testing is carried out in a lab environment and usually the testers are internal employees of the organization. To put it as simple as possible, this kind of testing is called alpha only because it is done early on, near the end of the development of the software, and before beta testing.

**Beta Testing** It is performed by "real users" of the software application in a "real environment" and can be considered as a form of external user acceptance testing. Beta version of the software is released to a limited number of end-users of the product to obtain feedback on the product quality. Beta testing reduces product failure risks and provides increased quality of the product through customer validation. It is the final test before shipping a product to the customers. Direct feedback from customers is a major advantage of Beta Testing. This testing helps to tests the product in real time environment.

1. **When do we use Black box and White box testing?**

**Black box testing:** It is a software testing technique that focuses on the analysis of software functionality, versus internal system mechanisms. Black box testing was developed as a method of analyzing client requirements, specifications and high-level design strategies. A black box software tester selects a set of valid and invalid input and code execution conditions and checks for valid output responses. Black box testing is also known as functional testing.

**White-box testing:** It is a methodology used to ensure and validate the internal framework, mechanisms, objects and components of a software application. White-box testing verifies code according to design specifications and uncovers application vulnerabilities. It is also known as White-box testing is also known as transparent box testing, clear box testing, structural testing and glass box testing. Glass box and clear box indicate that internal mechanisms are visible to a software engineering team.

1. **What we will do if we don't have a time to test all stories/ execute test cases?**

* Have management define priorities.
* Look for duplicate coverage. Remove redundant tests.
* Use test cases with the most coverage.
* Enlist help from other teams if you can. E.g. have the documentation team walk through their documentation steps.

1. **What we will do if come across any critical severity issue before release day?**

If I were in a position, then I will engage the team who knows about it, so they have the information they need to determine -their- best course of action as soon as possible and reduce the risk of it happening again.

1. **When do we use automation testing?**

Automation testing uses automation tools to write and execute test cases, no manual involvement is required while executing an automated test suite. Usually, testers write test scripts and test cases using the automation tool and then group into test suites. The main goal of Automation testing is to increase the test efficiency and develop software value.

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

**Requirement Phase**

In this phase, a test engineer will get an opportunity to indentify the necessities of project. Normally the data are recorded by the architecture team in the architectural reference document. Data design, information design, system design are the main issues in this phase.

**Analysis and Design Phase**

In this phase, a test engineer will get an opportunity to indentify how the project is planned. This is a major phase, where the entire design of the project is documented in the JAD phase in the System requirement document, business requirement document, product requirement document commercial use cases. Planner, Commercial reviewer, project organization, execution, testing, maintenance of project teams etc are attended the JAD phase to give sign-off on these completed document.

**Implementation-coding Phase**

In this phase, programmers play an important role of building the application depends on the plan acknowledged during the JAD stage. Here tester group have to follow the programming group to identify several adjustments taken by the system. There may be any kind of fault which are overlooked by programmer, misapprehend the planed records, in that time, a tester can always rise the issue to the regarding programmer to solve the issue. A testing group requires developing the high level scenarios (HLS) on basis of the elaboration phase. High level scenarios may have more than one test case. A tester should ensure that all the necessities are discovered to a test case by a quality affirmation standard. It is mandatory to record test cases on the basis of all probable references of the newest modernized data and also signed-off.

**Maintenance Phase**

In this phase if any fault or errors are originate then these are test again and it goes though the regression testing. With the help of regression testing, consistent systems develop. By the helping of these testing methods, any fundamental result can be converted into a tough and consistent system.

1. **Difference between load, performance, stress testing, spike testing?**

**Performance Testing:** It is also known as system level testing where it checks whether the system can stand for a high volume of **usage** at a particular situation by measuring the **response time**. It includes load, stress, spike, scalability, reliability, capacity, endurance testing etc.

**Load Testing:** It is a subset of performance testing which can be done by constantly increasing the load on the application till it reaches the threshold limit. The main aim is to check the upper limit system in terms of database, hardware and network etc. The common goal of doing the load testing is to set the **SLAs** for the application. For example, Running multiple applications on a computer simultaneously - starting with one application, then start second application, then third and so on. Now see the performance of your computer.

**SLAs –** Service-level agreement/contract between the service provider which defines how the service would be for a particular product. Some metrics will be like:

1. Availability of the service
2. Certain benchmarks through which the actual performance will be measured.
3. Application response time
4. Prior notification in advance of network changes.

**Stress Testing:** It checks the application behaviour beyond normal or peak loads. These are related to synchronization issues, memory leaks and race conditions.

**Spike Testing**: It is done to estimate the weakness of the application when unusual increment and decrements happens in the load.

1. **What is functional and non-functional testing types?**

**Functional testing:** In this all the functionalities which are defined by the user are tested and also verifies whether the system meets all the specified requirements of the user. For instance, basic features.

Main types **-** White box, Black box

Sub types **–** Unit, Integration, System, User Acceptance, Regression

**Non-functional testing:** It works opposite to functional whether it checks for behaviour of the system. For instance, the response time.

Types:

Performance Testing Operational Readiness Testing

Installation Testing Configuration Testing

Compatibility Testing Interoperability Testing

Maintainability Testing Availability Testing

Recoverability Testing Miscellaneous Testing

Security Testing Usability Testing

Reliability and Dependability Testing Endurance testing, Ergonomics Testing

Load testing Localization testing and Internationalization testing

1. **What is test case?**

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.

1. **What is test planning/test strategy document**

A test strategy is an outline that describes the testing approach of the software development cycle. It is created to inform project managers, testers, and developers about some key issues of the testing process. They are created based on development design documents.

1. **What is Exit and Entry criteria?**

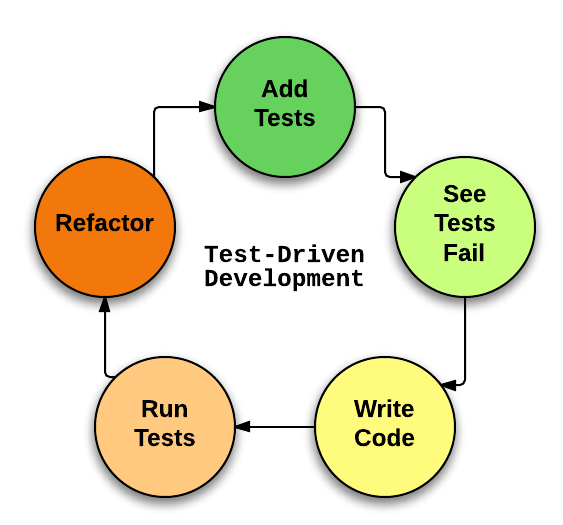
**Entry Criteria:** Entry Criteria gives the prerequisite items that must be completed before testing can begin.

**Exit Criteria:** Exit Criteria defines the items that must be completed before testing can be concluded

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

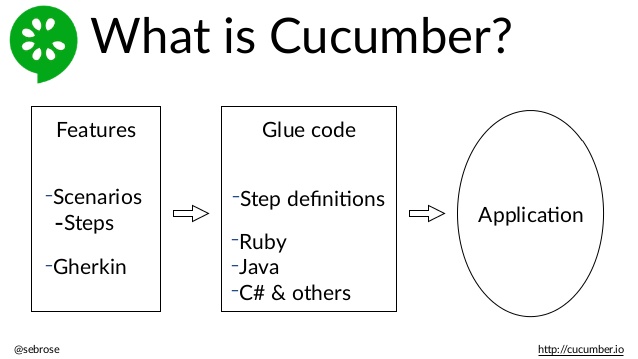
**TDD (Test Driven Development):**

* Mostly written by developers
* They are written to test each unit of the application so, its like unit testing
* To design low level scenarios
* It checks how each unit of the app works according to the desired specifications



**BDD (Behaviour Driven Development):**

* Written by stakeholders and Non-Technical people
* They are written for acceptance testing to check the behaviour of the application
* Design High-level Scenarios



**ATDD (Acceptance Test Driven Development):**

* It is also called as STDD (StoryTest driven development)
* All users, developers, testers collaboratively define the automated acceptance criteria.
* It helps the project people understand precisely what needs to be done.

1. **How do we write test cases in BDD format?**
2. Initially install Ruby Installer - > install Development kit for ruby (to import all the gems- use bundler for complete installation of gems)
3. Now, set a workspace for your cucumber project and then create feature file in features folder with **“.feature”** extension and write the code. For example

LoginFunctionality.feature

Feature: Verify LoginFunctionality

Scenario: Login with valid credentials

Given User is on Login page

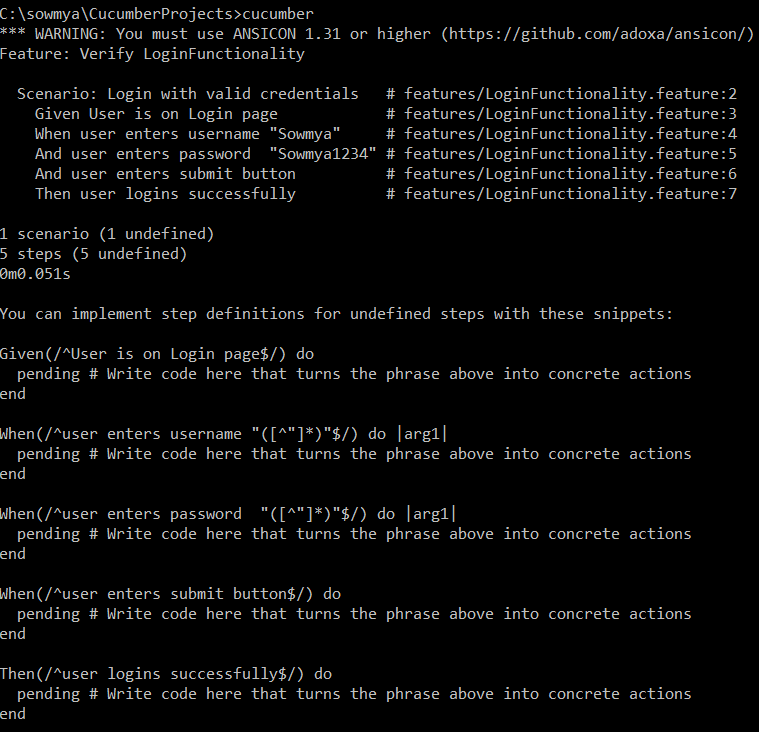
When user enters username "Sowmya"

And user enters password "Sowmya1234"

And user enters submit button

Then user logins successfully

1. After that open command prompt and execute the cucumber - C:\sowmya\CucumberProjects>cucumber Then you will get an output like



Next, Inorder to execute the acceptance tests which are written in the feature file we have to create step define file in step\_definitions folder with “.rb” extension and perform the necessary actions as shown below

StepFile.rb

Given(/^User is on Login page$/) do

puts "User is on Login page"

end

When(/^user enters username "([^"]\*)"$/) do |arg1|

puts "user name is " + arg1

end

When(/^user enters password "([^"]\*)"$/) do |arg1|

puts "user password is " + arg1

end

When(/^user enters submit button$/) do

puts "user enters submit button"

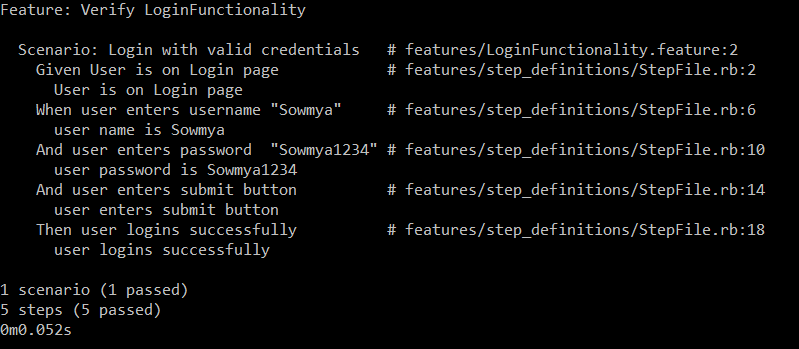
end

Then(/^user logins successfully$/) do

puts "user logins successfully"

end

Then, again we have to run like C:\sowmya\CucumberProjects>cucumber and now the output will be:



This way we can execute the test case but instead of writing something in the puts field, we have to implement the actual code in any language depending on the tool which we have taken

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

**Defect Severity:** It is a classification of software defect (bug) to indicate the degree of negative impact on the quality of software. It can be classified as follows:

Critical: The defect affects critical functionality or critical data. It does not have a workaround. Example: Unsuccessful installation, complete failure of a feature.

Major: The defect affects major functionality or major data. It has a workaround but is not obvious and is difficult. Example: A feature is not functional from one module but the task is doable if 10 complicated indirect steps are followed in another module/s.

Minor: The defect affects minor functionality or non-critical data. It has an easy workaround. Example: A minor feature that is not functional in one module but the same task is easily doable from another module.

Trivial: The defect does not affect functionality or data. It does not even need a workaround. It does not impact productivity or efficiency. It is merely an inconvenience. Example: Petty layout discrepancies, spelling/grammatical errors.

Severity is also denoted as:

S1 = Critical

S2 = Major

S3 = Minor

S4 = Trivial

**Defect Priority:** It indicates the importance or urgency of fixing a defect. Though priority may be initially set by the Software Tester, it is usually finalized by the Project/Product Manager. Priority can be categorized into the following levels:

**Urgent**: Must be fixed in the next build.

**High**: Must be fixed in any of the upcoming builds but should be included in the release.

**Medium**: May be fixed after the release / in the next release.

**Low**: May or may not be fixed at all**.**

1. **How to estimate test cases?**

Test estimation should be realistic and accurate.

List of Software Test Estimation Techniques

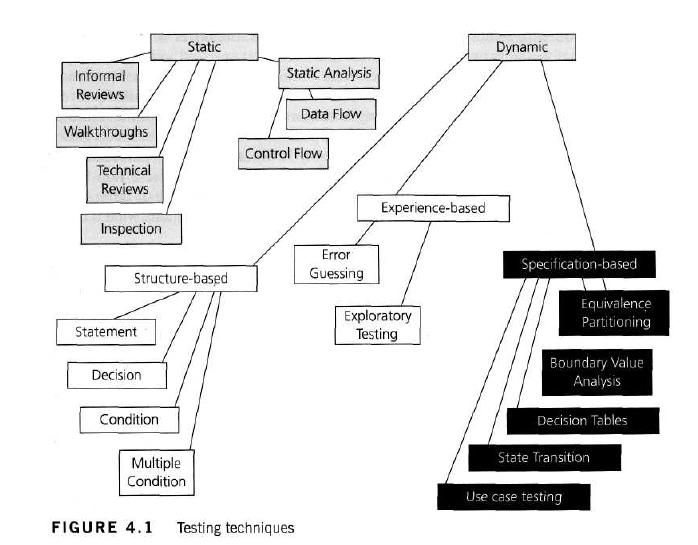
* Work Breakdown Structure
* 3-Point Software Testing Estimation Technique
* Wideband Delphi technique
* Function Point/Testing Point Analysis
* Use – Case Point Method
* Percentage distribution
* Ad-hoc method

1. **What is most challenge defect u came across?**

The most challenging defect are usually critical issues are the ones that might block the testing or cause data leaks or security breaches etc.

1. **What are test design techniques?**

Test Design is creating a set of inputs for given software that will provide a set of expected outputs. The idea is to ensure that the system is working good enough and it can be released with as few problems as possible for the average user. There are 2 for instance Static Techniques and Dynamic Techniques.



1. **If we don’t have time to test call test cases what we will do?**

When we do not have enough time to test, you might want to perform a risk analysis and determine which modules/areas of your AUT are prone to the highest risk and are critical to the success of the product and handle them first. Going the exploratory route instead of documenting the test cases is another way, but it is risky for sure.

1. **Functional testing?**

Functional testing is a software testing process used within software development in which software is tested to ensure that it conforms with all requirements. Functional testing is a way of checking software to ensure that it has all the required functionality that's specified within its functional requirements.

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

There are different tools for managing defects in the project. Below is the list.

1. Bugzilla
2. JIRA
3. HP ALM/Quality center
4. IBM Rational clearQuest
5. Lighthouse
6. **Who will assign the work?**

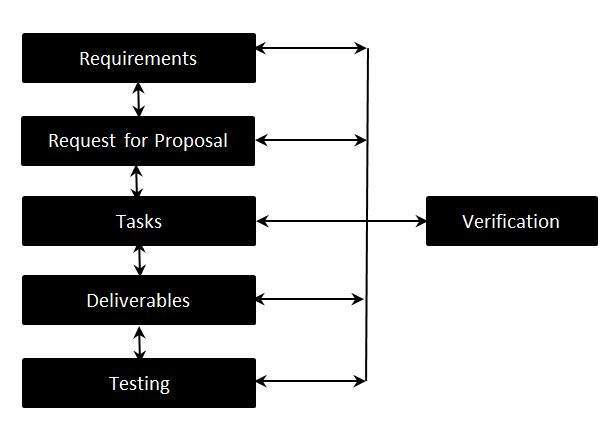
Project manager/ Team lead will be assigning the work.

1. **What is requirement 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.

**RTM** - WorkFlow:

The Matrix is created at the very beginning of a project as it forms the basis of the project's scope and deliverables that will be produced. It is bi-directional, as it tracks the requirement forward by examining the output of the deliverables and backward by looking at the business requirement that was specified for a particular feature of the product.



**Requirement traceability Matrix - Parameters:**

Requirement ID

Risks

Requirement Type

Requirement Description

Trace to Design Specification

Unit Test Cases

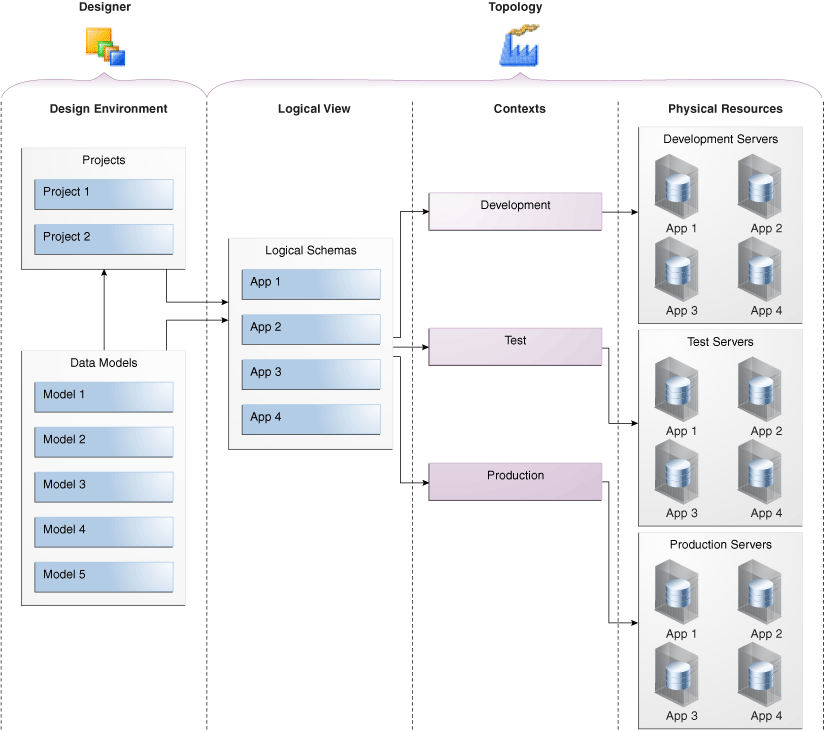
Integration Test Cases

System Test Cases

User Acceptance Test Cases

Trace to Test Script

1. **What are typical environments we have in projects**



1. **What are different defect metrics and measurements we prepare in testing**

There are many standard defect metrics such as:

* Defects Found/Resolved
* Defect Severity
* Defects Found/Test Cases Executed

1. **What is development environment?**

A development environment is a collection of procedures and tools for developing, testing and debugging an application or program. It has normally has three server tiers, called development, staging and production. All three tiers together are usually referred to as the DSP.

**Development Server**: Here is where the developer tests code and checks whether the application runs successfully with that code. Once the application has been tested and the developer feels that the code is working fine, the application then moves to the staging server.

**Staging Server**: This environment is made to look exactly like the production server environment. The application is tested on the staging server to check for reliability and to make sure it does not fail on the actual production server. This type of testing on the staging server is the final step before the application could be deployed on a production server. The application needs to be approved in order to deploy it on the production server.

**Production Server**: Once the approval is done, the application then becomes a part of this server.

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

1. **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.

1. **What is production environment?**

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.

1. **How to deal the production defects?**

At the production time, if we find any defect then I may choose to perform some kind of root cause analysis to determine when the defect was injected (in the current development cycle), why the defect wasn't discovered in previous development and QA cycles (if it wasn't recently introduced), and what types of tests should be created, executed, and managed to prevent this defect from returning in the future.