**SDLC**

**What is Software Development Life Cycle (SDLC)?**

SDLC is a process of methodology used to develop software in a systematic approach.

SDLC has 6 phases.

* Requirements Gathering (Business Analyst/Business Team)
* Planning and Analysis (Technical Manager or Senior Developer)
* Software Design (System Architect/Technical Lead)
* Development / Coding (Development Leads/Developers)
* Testing (Test Lead / Tester)
* Release and Maintenance (Project Manager/Released Team/Maintenance Team)

**Different types of SDLC Models?**

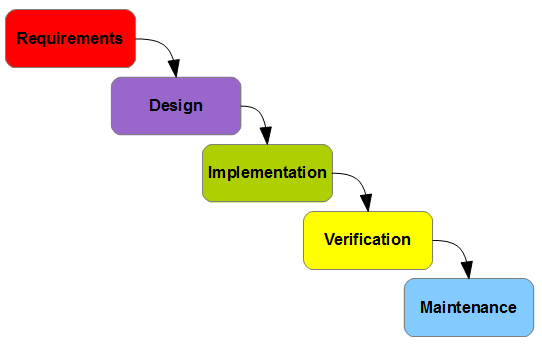
1. **Sequential Models:**development activities are executed in a sequential manner.
2. Waterfall model
3. V-Model
4. **Iterative / Incremental Models:**Process of developing software through repeated cycles (Iterative) and in smaller portions at a time (incremental) is called Iterative / Incremental SDLC
5. Prototyping model
6. Spiral Model
7. Agile Model

**Waterfall Model**

The Waterfall approach, the whole process of software development is divided into separate phases.

This is the most common and classic of life cycle models, also referred to as a linear-sequential life cycle model. In a waterfall model, each phase must be completed in its entirety before the next phase can begin.

(WATERFALL MODEL DIAGRAM)



**Advantages**

* Simple and easy to use.
* *Phases are processed and completed one at a time.*
* Easy to manage due to the rigidity of the model – each phase has specific deliverables and a review process.
* Works well for smaller projects where requirements are very well understood.

**Disadvantages**

* High amounts of risk and uncertainty.
* Poor model for complex, object-oriented, long and ongoing projects.
* Adjusting scope during the life cycle can kill a project
* No working software is produced until late during the life cycle.
* Poor model where requirements are at a moderate to high risk of changing.

**V-Model**

V-Model is also known as Verification and Validation Model. Testing is done in parallel with corresponding development activity.

It should be used for small to medium sized projects where requirements are clearly defined and fixed. Also should be chosen when plenty of technical resources are available with needed technical expertise.

Highly confidence customer choose V-Shaped model. Since, no prototypes are produced; there is high risk to fulfill the customer expectations.

Testing under V-Model:

1. Component Testing
2. Integration Testing: In general Developers perform integration testing in V-Model
3. System Testing
4. Acceptance Testing

(V-MODEL DIAGRAM)

[](http://istqbexamcertification.com/wp-content/uploads/2012/01/V-model.jpg)

**Advantages of V-model over Waterfall Model:**

Software quality gets improved in V-Model as it has testing activities in parallel for the respective development activity. Defects can be identified in early stages of life cycle and defects multiplication can be avoided.

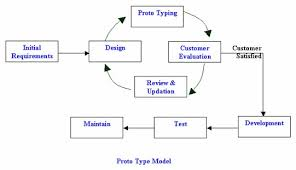
**Iterative / Incremental SDLC Model:**

Process of developing software through repeated cycles (Iterative) and in smaller portions at a time (incremental) is called Iterative / Incremental SDLC.

**Prototype Model:**

A prototype is the sample implementation of the system that shows limited and main functional capabilities of the proposed system. After a prototype is built, it is delivered to the customer for the evaluation. The prototype helps the customer determine how the feature will function in the final software. The customer provides suggestion and improvements on the prototype. The development team implements the suggestion in the new prototype, which is again evaluated by the customer. The process continues until the customer and the development team understands the exact requirement of the proposed system. When the final prototype is developed, the requirement is considered to be frozen.

(PROTOTYPE MODEL DIAGRAM)



***Example:*** *An e-commerce website, such as shopping site is an example where you can implement the prototyping approach. You can develop the prototype of the various web pages of the shopping site such as catalogue page, product order page etc., and present it to the customer for approval. If the customer approves the prototype of the site, requirements are states again and the design of the web site is initiated. If the customer does not approve the web site, the development team revisits the prototype and resubmits it to the customer for approval. This process continues until the prototype is approved.*

**Advantages of Prototype:**

* Clarity on requirements from the customer, success rate of the project is higher.
* Customer can see the system and he provides the feedback immediately.
* Errors can be detected much earlier as customer is getting involved.

**Disadvantages of Prototype:**

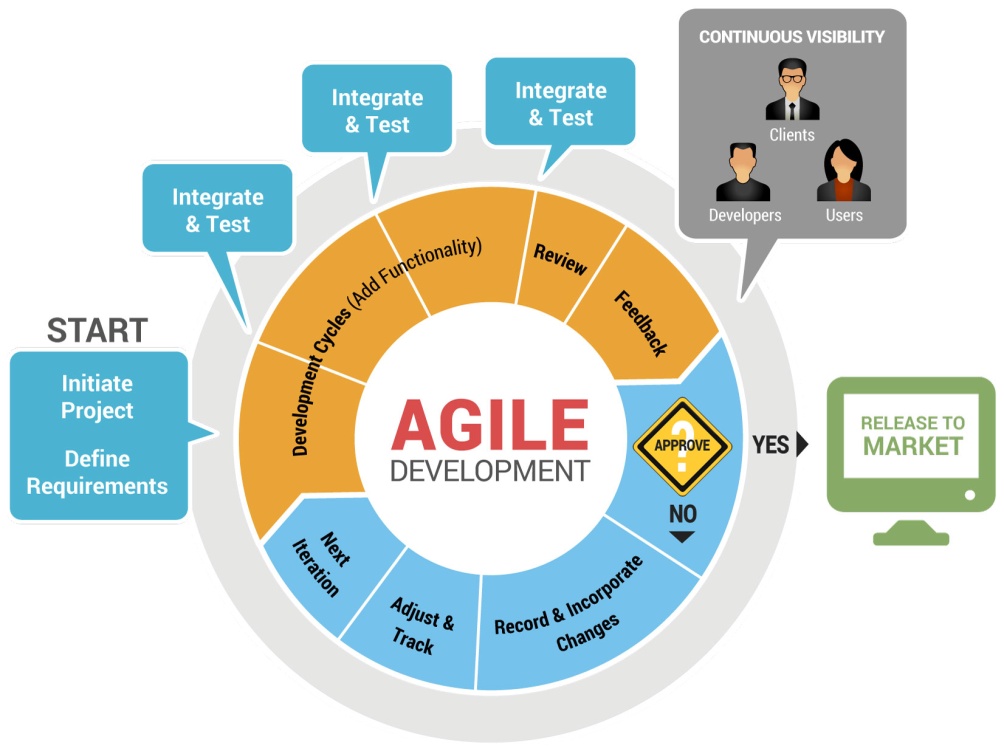
* Customer may think that prototype is the real system
* Time taking model as the actual development does not start until the prototype is finalized
* Adopting changes or adding new requirements is difficult
* Prototype is of no use once the development of actual system is started

**What is Agile Model?**

Agile means “able to move quickly and easily”. Agile software life cycle is an iterative process where software is ready at each of iteration but can always be improved in next iteration. When applied to Software Development, it means delivering the software within shortest possible time according to customer requirements.

(Agile is based on Iterative / Incremental development in which requirements and solutions develop through collaboration between self-organizing and cross-functional teams. It promotes adaptive planning, evolutionary development and delivery, a time-boxed iterative approach and encourages rapid and flexible response to change.)

(AGILE MODEL DIAGRAM)



**Advantages and Disadvantages of Agile:**

*Advantages:*

1. Success rate of the project very high compared to any other models
2. Can adopt changes in requirements at any point of time
3. Working software is delivered frequently
4. It emphasizes on responding to change rather than extensive planning and documentation
5. It is recommended for Product Development

*Disadvantages:*

1. Expensive Model as more number of resources are required
2. Complex in Managing
3. There is lack of emphasis on necessary designing and documentation
4. The project can easily get taken off track if there is any communication gap

**Why Agile model so popular than other SDLC models?**

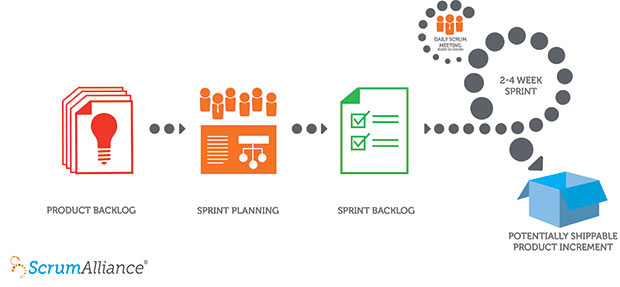
It is popular because of its flexibility in adopting changes in requirements and at the same time delivering software in shortest possible time.

**Different types of agile models:**

1. **Extreme Programming:** Extreme programming (XP) is a software development methodology which is intended to improve software quality and responsiveness to changing customer requirements. As a type an agile software development, it advocates frequent releases in short development cycle, which is intended to improve productivity and introduce checkpoints where new customer requirements can be adopted.
2. **Scrum** is an [Agile framework](https://www.scrumalliance.org/why-scrum/core-scrum-values-roles) for completing complex projects. Scrum originally was formalized for software development projects, but it works well for any complex, innovative scope of work. The possibilities are endless. The Scrum framework is deceptively simple.

### The Scrum framework in 30 seconds

* A product owner creates a prioritized wish list called a product backlog.
* During sprint planning, the team pulls a small chunk from the top of that wish list, a sprint backlog, and decides how to implement those pieces.
* The team has a certain amount of time — a sprint (usually two to four weeks) — to complete its work, but it meets each day to assess its progress (daily Scrum).
* Along the way, the ScrumMaster keeps the team focused on its goal.
* At the end of the sprint, the work should be potentially shippable: ready to hand to a customer, put on a store shelf, or show to a stakeholder.
* The sprint ends with a sprint review and retrospective.
* As the next sprint begins, the team chooses another chunk of the product backlog and begins working again.

[](https://www.scrumalliance.org/why-scrum/core-scrum-values-roles)

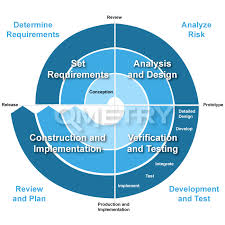
### Beyond the sprint

The cycle repeats until enough items in the product backlog have been completed, the budget is depleted, or a deadline arrives. Which of these milestones marks the end of the work is entirely specific to the project. No matter which impetus stops work, Scrum ensures that the most valuable work has been completed when the project ends.

**Spiral Model:**

The spiral model is an incremental model with more emphasis placed on risk analysis. This model has 4 phases: **Planning, Risk analysis, Engineering and Evaluation**. A software project repeatedly passes through this phases in iterations called Spiral in this model.

(SPIRAL MODEL DIAGRAM)

[](http://www.google.com/url?sa=i&rct=j&q=&esrc=s&source=images&cd=&cad=rja&uact=8&docid=t0lKVY7OjwL90M&tbnid=KIvV6_6OHIMebM:&ved=0CAUQjRw&url=http://www.qmetry.com/spiral.html&ei=NmbsU53YJcuNyASFzoGIDA&psig=AFQjCNFYqrVAcdK9iKysNvk7G9Zy6kMg9g&ust=1408087767511571)

*Advantages:*

* High amount of risk analysis and hence avoidance of Risk is enhanced
* Good for large and mission-critical project
* Strong approval and documentation control
* Additional Functionality can be added at a later date.
* Software is produced early in the software life cycle.

*Disadvantages:*

* Can be a costly model to use
* Risk analysis requires highly specific expertise
* Project’s success is highly dependent on the risk analysis phase
* Doesn’t work well for smaller project

**Difference between sequential and iterative models:**

1. In sequential models, software is developed at once and delivered where as in Iterative models; software is divided into increments and developed.
2. In sequential models, working software is not produced until late in the life cycle, where as in Iterative models, software is developed and delivered early in increments.

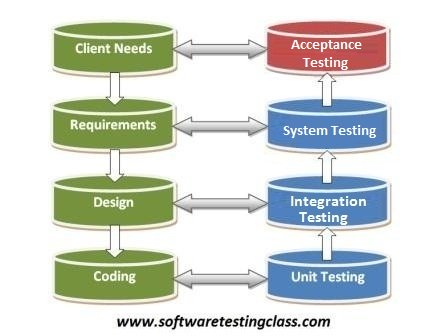
**TEST LEVELS**

**Two types of Testing:**

* 1. **Functional**
  2. **Non Functional**

1. **Functional Testing:**

This is a type of black box testing that is based on the specifications of the software that is to be tested. The application is tested by providing input and then the results are examined that need to conform to the functionality it was intended for. Functional Testing of the software is conducted on a complete, integrated system to evaluate the system's compliance with its specified requirements.

[](http://www.softwaretestingclass.com/wp-content/uploads/2012/10/acceptance-testing-levels-of-testing.jpg)

**Different types of Functional Test Levels are:**

1. Unit/Component/Program/Module Testing
2. Integration Testing
3. System Testing
4. Regression Testing
5. User Acceptance Testing (UAT)

**1. Unit/Component/Program/Module Testing:** Test each unit (module). Mostly are done by the Developer.

Test Plan: Prepare > Review > Rework > Baseline > Revise (if necessary Review > Rework > Baseline)

Test Case/Script: Prepare > Review > Rework > Baseline > Execute> Revise

E.g. for login:

With valid ID and password

With invalid ID and password

Empty (without ID and password)

#### 2. Integration Testing is software testing methodology used to test individual software components or units of code to verify interaction (communication) between various software components and identify interface defects.

OR

The testing of combined parts of an application to verify if they function correctly together is Integration testing. There are two methods of doing Integration Testing Bottom-up Integration testing and Top-Down Integration testing.

*Different types of Integration Testing*

* + Big Bang approach: *all the developed modules are coupled together to form a complete system and it is tested as whole.*
  + Top-down approach*: highest levels modules are integrated first and testing takes place from to down.*
  + Bottom-up approach: *testing of low levels components is done first and move towards the high level components.*
  + Hybrid approach: *is combination of top-down and bottom-up approach.*

**3. System Testing:** Concerned with the behavior of a system as a whole. Checks complete end-to-end scenarios as the way a customer would use the system.

Or

This is the next level in the testing and tests the system as a whole. Once all the components are integrated, the application as a whole is tested rigorously to see that it meets Quality Standards. This type of testing is performed by a specialized testing team.

**4. Regression Testing:** is retesting or previously tested program following modification to ensure that defects have not been introduced or uncovered in unchanged areas of the software, as a result the changes made. It is performed when the software or its environment is changed.

**5.** [**User Acceptance testing**](http://www.softwaretestingclass.com/user-acceptance-testing-what-why-how): is the software testing process where system tested for acceptability & validates the end to end business flow. Such type of testing executed by client in separate environment (similar to production environment) & confirm whether system meets the requirements as per requirement specification or not.

**6. Acceptance testing** is **“**[**black box**](http://www.softwaretestingclass.com/category/black-box-testing/)**”** tests, means UAT users doesn’t aware of internal structure of the code, they just specify the input to the system & check whether systems respond with correct result.

**7. Alpha testing:** This test is the first stage of testing and will be performed amongst the teams (Developer and QA teams). Unit testing, integration testing and system testing when combined are known as ALPHA testing. During this phase, the following will be tested in the application:

* Spelling Mistakes
* Broken Links
* Cloudy Directions
* The Application will be tested on machines with the lowest specification to test loading times and any latency problems.

### 8. Beta Testing: is performed after Alpha testing has been successfully performed. In beta testing a sample of the intended audience tests the application. Beta testing is also known as pre-release testing. Beta test versions of software are ideally distributed to a wide audience on the Web, partly to give the program a "real-world" test and partly to provide a preview of the next release. In this phase the audience will be testing the following:

* Users will install, run the application and send their feedback to the project team.
* Typographical errors, confusing application flow, and even crashes.
* Getting the feedback, the project team can fix the problems before releasing the software to the actual users.
* The more issues you fix that solve real user problems, the higher the quality of your application will be.
* Having a higher-quality application when you release to the general public will increase customer satisfaction.

# Non-Functional Testing:

This section is based upon the testing of the application from its non-functional attributes. Non-functional testing of Software involves testing the Software from the requirements which are non functional in nature related but important a well such as performance, security, user interface etc.

Some of the important and commonly used non-functional testing types are mentioned as follows:

## Performance Testing:

It is mostly used to identify any bottlenecks or performance issues rather than finding the bugs in software. There are different causes which contribute in lowering the performance of software:

* Network delay.
* Client side processing.
* Database transaction processing.
* Load balancing between servers.
* Data rendering.

Performance testing is considered as one of the important and mandatory testing type in terms of following aspects:

* Speed (i.e. Response Time, data rendering and accessing)
* Capacity
* Stability
* Scalability

It can be either qualitative or quantitative testing activity and can be divided into different sub types such as

**Load testing** and **Stress testing**.

### Load Testing

A process of testing the behavior of the Software by applying maximum load in terms of Software accessing and manipulating large input data. It can be done at both normal and peak load conditions. This type of testing identifies the maximum capacity of Software and its behavior at peak time.

Most of the time, Load testing is performed with the help of automated tools such as Load Runner, AppLoader, IBM Rational Performance Tester, Apache JMeter, Silk Performer, Visual Studio Load Test etc.

Virtual users (VUsers) are defined in the automated testing tool and the script is executed to verify the Load testing for the Software. The quantity of users can be increased or decreased concurrently or incrementally based upon the requirements.

### Stress Testing

This testing type includes the testing of Software behavior under abnormal conditions. Taking away the resources, applying load beyond the actual load limit is Stress testing.

The main intent is to test the Software by applying the load to the system and taking over the resources used by the Software to identify the breaking point. This testing can be performed by testing different scenarios such as:

* Shutdown or restart of Network ports randomly.
* Turning the database on or off.
* Running different processes that consume resources such as CPU, Memory, server etc.

## Usability Testing

This section includes different concepts and definitions of Usability testing from Software point of view. It is a black box technique and is used to identify any error(s) and improvements in the Software by observing the users through their usage and operation.

According to Nielsen, Usability can be defined in terms of five factors i.e. Efficiency of use, Learn-ability, Memory-ability, Errors/safety, satisfaction. According to him the usability of the product will be good and the system is usable if it possesses the above factors.

Nigel Bevan and Macleod considered that Usability is the quality requirement which can be measured as the outcome of interactions with a computer system. This requirement can be fulfilled and the end user will be satisfied if the intended goals are achieved effectively with the use of proper resources.

**Molich in 2000 stated that user friendly system should fulfill the following five goals i.e. Easy to Learn, Easy to Remember, Efficient to Use, Satisfactory to Use and Easy to Understand.**

In addition to different definitions of usability, there are some standards and quality models and methods which define the usability in the form of attributes and sub attributes such as ISO-9126, ISO-9241-11, ISO-13407 and IEEE std.610.12 etc.

### UI Vs Usability Testing

UI testing involves the testing of Graphical User Interface of the Software. This testing ensures that the GUI should be according to requirements in terms of color, alignment, size and other properties.

On the other hand Usability testing ensures that a good and user friendly GUI is designed and is easy to use for the end user. UI testing can be considered as a sub part of Usability testing.

## Security Testing

Security testing involves the testing of Software in order to identify any flaws ad gaps from security and vulnerability point of view. Following are the main aspects which Security testing should ensure:

* Confidentiality.
* Integrity.
* Authentication.
* Availability.
* Authorization.
* Non-repudiation.
* Software is secure against known and unknown vulnerabilities.
* Software data is secure.
* Software is according to all security regulations.
* Input checking and validation.
* SQL insertion attacks.
* Injection flaws.
* Session management issues.
* Cross-site scripting attacks.
* Buffer overflows vulnerabilities.
* Directory traversal attacks.

## Portability Testing

Portability testing includes the testing of Software with intend that it should be re-useable and can be moved from another Software as well. Following are the strategies that can be used for Portability testing.

* Transferred installed Software from one computer to another.
* Building executable (.exe) to run the Software on different platforms.Portability testing can be considered as one of the sub parts of System testing, as this testing type includes the overall testing of Software with respect to its usage over different environments. Computer Hardware, Operating Systems and Browsers are the major focus of Portability testing. Following are some pre-conditions for Portability testing:
* Software should be designed and coded, keeping in mind Portability Requirements.
* Unit testing has been performed on the associated components.
* Integration testing has been performed.
* Test environment has been established.

**Test Documentation**

Testing documentation involves the documentation of artifacts which should be developed before or during the testing of Software.

Documentation for Software testing helps in estimating the testing effort required, test coverage, requirement tracking/tracing etc. This section includes the description of some commonly used documented artifacts related to Software testing such as:

* Test Plan
* Test Scenario
* Test Case
* Traceability Matrix

## Test Plan

A test plan outlines the strategy that will be used to test an application, the resources that will be used, the test environment in which testing will be performed, the limitations of the testing and the schedule of testing activities. Typically the Quality Assurance Team Lead will be responsible for writing a Test Plan.

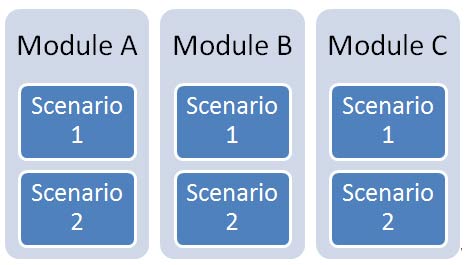
A test plan will include the following.

* Introduction to the Test Plan document
* Assumptions when testing the application
* List of test cases included in Testing the application
* List of features to be tested
* What sort of Approach to use when testing the software
* List of Deliverables that need to be tested
* The resources allocated for testing the application
* Any Risks involved during the testing process
* A Schedule of tasks and milestones as testing is started

## Test Scenario

A one line statement that tells what area in the application will be tested. Test Scenarios are used to ensure that all process flows are tested from end to end. A particular area of an application can have as little as one test scenario to a few hundred scenarios depending on the magnitude and complexity of the application.

The term test scenario and test cases are used interchangeably however the main difference being that test scenarios has several steps however test cases have a single step. When viewed from this perspective test scenarios are test cases, but they include several test cases and the sequence that they should be executed. Apart from this, each test is dependent on the output from the previous test.



## Test Case

Test cases involve the set of steps, conditions and inputs which can be used while performing the testing tasks. The main intent of this activity is to ensure whether the Software Passes or Fails in terms of its functionality and other aspects. There are many types of test cases like: functional, negative, error, logical test cases, physical test cases, UI test cases etc.

Furthermore test cases are written to keep track of testing coverage of Software. Generally, there is no formal template which is used during the test case writing. However, following are the main components which are always available and included in every test case:

* Test case ID.
* Product Module.
* Product version.
* Revision history.
* Purpose
* Assumptions
* Pre-Conditions.
* Steps.
* Expected Outcome.
* Actual Outcome.
* Post Conditions.

Many Test cases can be derived from a single test scenario. In addition to this, some time it happened that multiple test cases are written for single Software which is collectively known as test suites.

## Traceability Matrix

Traceability Matrix (also known as Requirement Traceability Matrix - RTM) is a table which is used to trace the requirements during the Software development life Cycle. It can be used for forward tracing (i.e. from Requirements to Design or Coding) or backward (i.e. from Coding to Requirements). There are many user defined templates for RTM.

Each requirement in the RTM document is linked with its associated test case, so that testing can be done as per the mentioned requirements. Furthermore, Bug ID is also include and linked with its associated requirements and test case. The main goals for this matrix are:

* Make sure Software is developed as per the mentioned requirements.
* Helps in finding the root cause of any bug.
* Helps in tracing the developed documents during different phases of SDLC.

**Tasks involved in Test Execution phase:**

1. Creating test batches
2. Verifying test environment set up
3. Test execution
4. Test reporting
5. Defect reporting
6. Re and regression testing

**Input for test execution phase**: Requirements specification, test plan, test case documents, test data, defect report and test report.

**Output for test execution phase**: Test report, opened/closed defect reports

**Task involved Test closure phase:**

1. Evaluating exit criteria
2. Collecting all from test activities
3. Sending test deliverables to the customer
4. Improvement suggestions for future projects

**Input for test closure phase:** requirements, test plan, test reports , opened/closed defect reports, test summery report template.

**Output for test closure phase:**  test summery report, test deliverables (test plan, test scenario, test case, test data, test report, opened/closed defect reports)

**Test metrics**: used in decision making

**Test deliverables:** any test work product that must be delivered to someone other than the test work product’s author.

**Base lined documents:** Approved and finalized documents.

**Defect Report:** a document reporting any flaw in a component or system that can cause the system to fail to perform its required function.

**Test summery report:** a document summarizing testing activities and result. It also contains an evaluation of the corresponding test items against exit criteria.

**Software Testing Estimation Techniques**

Estimating effort for test is one of the major and important tasks in SDLC. Correct estimation helps in testing the Software with maximum coverage. This section describes some of the techniques which can be useful during the estimating of effort for testing.

## Functional Point Analysis

This method is based on the analysis of functional user requirements of the Software with following categories:

* Outputs
* Inquiries
* Inputs
* Internal files
* External files

## Test Point Analysis

This estimation process is used for function point analysis for Black box or Acceptance testing. It is use the main elements of this method are: Size, Productivity, Strategy, Interfacing, Complexity and Uniformity etc.

## Mark-II method

It is estimation method used for analysis and measuring the estimation based on end user functional view. The procedure for Mark-II method is:

* Determine the View Point
* Purpose and Type of Count
* Define the Boundary of Count
* Identify the Logical transactions
* Identify and Categorize Data Entity Types
* Count the Input Data Element Types
* Count the Functional Size

## Miscellaneous

You can use other popular estimation techniques like:

* Delphi Technique
* Analogy Based Estimation
* Test Case Enumeration Based Estimation
* Task (Activity) based Estimation
* IFPUG method

**Software Testing:**

Testing is executing a system in order to identify any gaps, errors or missing requirements in contrary to the actual desire or requirements.

Or

Software testing is a process that should be done during the development process. In order words software testing is a verification and validation process.

**What is Verification?**

Verification is the process to make sure the product satisfies the conditions imposed at the start of the development phase. In other words, to make sure the product behaves the way we want it to. *White box testing is often used for verification.*

**What is Validation?**

Validation is the process to make sure the product satisfies the specified requirements at the end of the development phase. In other words, to make sure the product is built as per customer requirements. *Black box testing is often used for validation.*

## Verification & Validation

These two terms are very confusing for people, who use them interchangeably. Let's discuss about them briefly.

|  |  |  |
| --- | --- | --- |
| **S.N.** | **Verification** | **Validation** |
| 1 | Are you building it right? | Are you building the right thing? |
| 2 | Ensure that the software system meets all the functionality. | Ensure that functionalities meet the intended behavior. |
| 3 | Verification takes place first and includes the checking for documentation, code etc. | Validation occurs after verification and mainly involves the checking of the overall product. |
| 4 | Done by developers. | Done by Testers. |
| 5 | Have static activities as it includes the reviews, walkthroughs, and inspections to verify that software is correct or not. | Have dynamic activities as it includes executing the software against the requirements. |
| 6 | It is an objective process and no subjective decision should be needed to verify the Software. | It is a subjective process and involves subjective decisions on how well the Software works. |

**There are three types testing are:**

* 1. **Black box Testing:** The technique of testing without having any knowledge of the interior workings of the application is Black Box testing. The tester is oblivious to the system architecture and does not have access to the source code. Typically, when performing a black box test, a tester will interact with the system's user interface by providing inputs and examining outputs without knowing how and where the inputs are worked upon.
  2. **White box Testing:** White box testing is the detailed investigation of internal logic and structure of the code. White box testing is also called glass testing or open box testing. In order to perform white box testing on an application, the tester needs to possess knowledge of the internal working of the code.

The tester needs to have a look inside the source code and find out which unit/chunk of the code is behaving inappropriately.

* 1. **Grey Box Testing:** Grey Box testing is a technique to test the application with limited knowledge of the internal workings of an application. In software testing, the term “*the more you know the better”* carries a lot of weight when testing an application.

Mastering the domain of a system always gives the tester an edge over someone with limited domain knowledge. Unlike black box testing, where the tester only tests the application's user interface, in grey box testing, the tester has access to design documents and the database. Having this knowledge, the tester is able to better prepare test data and test scenarios when making the test plan.

## Black Box Vs Grey Box Vs White Box

|  |  |  |  |
| --- | --- | --- | --- |
| **S.N.** | **Black Box Testing** | **Grey Box Testing** | **White Box Testing** |
| 1 | The Internal Workings of an application are not required to be known | Somewhat knowledge of the internal workings are known | Tester has full knowledge of the Internal workings of the application |
| 2 | Also known as closed box testing, data driven testing and functional testing | Another term for grey box testing is translucent testing as the tester has limited knowledge of the insides of the application | Also known as clear box testing, structural testing or code based testing |
| 3 | Performed by end users and also by testers and developers | Performed by end users and also by testers and developers | Normally done by testers and developers |
| 4 | Testing is based on external expectations - Internal behavior of the application is unknown | Testing is done on the basis of high level database diagrams and data flow diagrams | Internal workings are fully known and the tester can design test data accordingly |
| 5 | This is the least time consuming and exhaustive | Partly time consuming and exhaustive | The most exhaustive and time consuming type of testing |
| 6 | Not suited to algorithm testing | Not suited to algorithm testing | Suited for algorithm testing |
| 7 | This can only be done by trial and error method | Data domains and Internal boundaries can be tested, if known | Data domains and Internal boundaries can be better tested |

Random Questions

**What is Quality Assurance?**

Quality Assurance (QA) aims to prevent defects with a focus on the process of producing the software. It is defect Prevention oriented.

**What is Quality Control (QC)?**

Quality Control aims to identify defects in the developed software. It is defect-detection and correction oriented.

**What is testing?**

It is a process of executing a program to find defects and helps to measure the quality of software.

**What is Static Testing?**

Static testing is done during verification process. This testing includes reviewing of the documents (including source code) and static analysis. This is useful and cost effective way of testing. For example: reviewing, walkthrough, inspection, etc.

**What is Dynamic Testing?**

In dynamic testing the software code is executed to demonstrate the result of running tests. It’s done during validation process. For example: unit testing, integration testing, system testing, etc.

**Functional Testing:** test the functionalities of the application to make sure it’s working as per the Business Requirement Document (BRD). Major test is where 80% of the tests are done.

**Non-Functional:**

**Smoke Testing:** initial testing to check whether the major functionalities are working.

**Sanity/build verification Testing:** ensure basic functionality, more likely smoke.

**Differences between smoke and sanity testing:** According to standard industry, there is no difference between both of them.

**Re-testing VS Regression:** Retesting is done to verify defect fixes, and regression is done to check that existing functionality was working that was working fine before changes.

**Adhoc testing:** testing done without using any formal testing technique**.**

**Exploratory testing:** formal testing process where there is no test case is available to test the application. It is understood by exploring app and understands the functionality.

**Internationalization testing:** to know how it displays the message or various languages and how it handles the inputs, strings and sorting items.

**Localization testing:** testing whether the software functions as expected in localized environment after translating the message, documents and customizing the resource.

**Exhaustive testing**: testing which covers all combination of input values and preconditions for an element of the software under test.

**Compatibility testing**: compatibility with other elements such as browser, operating system.

**Reliability testing**: to check the ability of software to function under given environmental conditions for particular amount of time.

**Volume testing:** it is about the certain amount of data which help you to find the data size, generic terms etc.

**Spike testing:** Spike testing is done by suddenly increasing the number of or load generated by, users by a very large amount and observing the behavior of the system. The goal is to determine whether performance will suffer, the system will fail, or it will be able to handle dynamic changes in load.

**Endurance testing:** it is usually done to determine if the system can sustain continues expected load. This is, to ensure that the throughput and /or response time after some long period of sustained activity are as good as or better than at the beginning of the test.

**Soak testing:** It involves testing a system with a significant load extended over a significant period of time, to discover how the system behaves under sustained use.

**Configuration testing:** it is determined where system components changes effects on system performance and behavior.

**Risk based testing**: It’s prioritizing and executing the test of features and functions based on the risk of their failure.

**Recovery Testing**: It’s an activity of testing how well an application is able to recover from crashes, hardware failures and other similar problems.

**Database Testing**: Database testing mainly deals with finding errors in the databases so as to eliminate them. This will improve the quality of the database.

**GUI Testing:** Graphical User Interface (GUI) testing is the process of testing software user interface against the provided requirements/mockups/HTML designs.

**Formal Testing:** Software verification carried out by following test plan, testing procedures and proper documentation with an approval from customer

**Dynamic Testing:** It is the testing done by executing the code or program with various input values and output is verified.

**Positive testing:** test to pass with valid data.

**Negative testing:** test to make sure the system is not doing what it’s not supposed to do.

**Shakeout Testing:** to check the networking facility, database connectivity and the integration of modules-done by the configuration team.

**TEST DESIGN TECHNIQUE**

**Test Design Technique:** Test design is procedure to select few test cases out of many with the likelihood of identifying defects.

**Different types of Test Design Technique:**

* Specification-based or Black-Box Testing
* Structure-based or White-box Testing
* Experience based Testing

**Different Types of Black Box Testing Techniques:**

* Equivalent partitioning or Equivalent class
* Boundary Value analysis
* Decision Tables
* State Transition Testing
* Use cases testing

**Different Types of White Box Testing Techniques:**

* Statement Coverage
* Decision Coverage

**What is Equivalence Partitioning or Equivalent Class?**

Equivalence partitioning involves identifying a small set of input values that produce many different output conditions as possible. This reduces number of permutations and combinations of input, output values used for testing thereby increasing the coverage and reducing the effort involved in testing.

**What is Boundary Value Analysis?**

Boundary value analysis is based on testing the boundaries between partitions. Test Cases are designed based on boundary values.

**State Transition Testing:**

State Transition Testing is a test design technique in which test cases are designed to execute valid and invalid state transitions.

**Used Case Testing:**

Used case testing is technique that helps us identify test cases that exercise the whole system on a transaction by transaction basis from start to finish.

**Statement Coverage:**

Statement coverage is the percentage of executable statements that have been exercised by a test suite.

**Decision Coverage:**

Decision coverage is the percentage of decision outcomes that have been exercised by a test suite. 100% decision coverage implies 100% statement coverage.

**Condition Coverage:**

The percentage of condition is outcomes that have been exercised by test suite. 100% code coverage requires each single condition in every decision statement to be tested as True or False.

**Multi-Condition Coverage:**

Design test cases for each combination of condition are called multi-condition testing.

**Experienced-based testing or Informal Testing:**

Experience-based testing is a procedure to select test cases based on the tester’s experience, knowledge and intuition.

**Different types of Informal Testing Techniques?**

**Error guessing:** Error guessing is a technique where the experience of the tester is used to anticipate what defects might be present in the component or system under test as a result of errors made, and to design tests specifically to expose them.

**Exploratory Testing:** Exploratory Testing is a test design technique where the tester actively controls the design of the tests as those tests are performed and uses information gained while testing to design new and better tests.

**Decision Table:**

A table showing combinations of inputs and stimuli (causes) with their associated outputs and/or actions (effects), which can be used to design Test cases.

**State Diagram:**

A diagram that depicts the states that a component or system can assume and shows the events or circumstances that cause or result from a change from one state to another.

**When to use Decision Table Technique?**

When different combinations of inputs result in different actions being taken, it would be more difficult to show using equivalence partitioning and boundary value analysis, which tend to be more focused on the user interface.

**What is Decision Table Testing?**

Decision table testing is a test design technique in which test cases are designed to execute the combination of inputs shown in a decision table.

**Software Testing Life Cycle (STLC)**

STLC it is a systematic approach to test the software. Main object is to monitor and control testing activities.

The testing of software has its own life cycle: Four different phases: (basic)

* Requirement study 1. Test Planning
* Test planning 2. Test Design
* Writing test scripts 3. Test Execution
* Reviewing test scripts 4. Test Closure
* Executing test scripts
* Bug logging and tracking
* Close or reopen bugs

**Difference between SDLC and STLC:**

STLC is a part of SDLC. SDLC involves the complete verification and validation of a software where as STLC involves only validation of the system.

**What is a Test Plan?**

A document is describing the detail approach to test the software and what the eventual workflow will be. It consists of features to be tested, features not to be tested, approach, entry criteria, exit criteria, suspension criteria, resume criteria, test environment, training needs, resources, roles and responsibilities, risk and contingency plan.

* Test strategy: Organization level document that describes the testing approach during SDLC
* Entry Criteria: Describes when to start testing
* Exit Criteria: Describes when to stop testing
* Suspension criteria: Describes when to stop testing temporarily
* Resume criteria: Describes when resume or restart testing

**Tasks involved in Test Planning:**

1. Understand and analyze the requirements
2. Risk analysis
3. Test strategy and implementation:
4. Test estimation
5. Team formation
6. Test plan documentation
7. Conflagration management planning
8. Traceability matrix
9. Define test environment set up

**Tasks involved in Test Design Phase:**

1. Creating test scenarios
2. Test case documentation
3. Test data collection

**Input for test design**: Requirements specification, test plan, test scenario template, test case template, and test data template.

**Output for test design:** Test Cases documents and Test Data

**Traceability Matrix:** Document showing the relationship between requirements and test cases. (Test lead/team lead creates and testers update it throughout the STLC)

**Tasks involved in Test Execution phase:**

1. Creating test batches
2. Verifying test environment set up
3. Test execution
4. Test reporting
5. Defect reporting
6. Re and regression testing

**Input for test execution phase**: Requirements specification, test plan, test case documents, test data, defect report and test report.

**Output for test execution phase**: Test report, opened/closed defect reports

**Task involved Test closure phase:**

1. Evaluating exit criteria
2. Collecting all from test activities
3. Sending test deliverables to the customer
4. Improvement suggestions for future projects

**Input for test closure phase:** requirements, test plan, test reports , opened/closed defect reports, test summery report template.

**Output for test closure phase:**  test summery report, test deliverables (test plan, test scenario, test case, test data, test report, opened/closed defect reports)

**Test metrics**: used in decision making

**Test deliverables:** any test work product that must be delivered to someone other than the test work product’s author.

**Base lined documents:** Approved and finalized documents.

**Defect Report:** a document reporting any flaw in a component or system that can cause the system to fail to perform its required function.

**Test summery report:** a document summarizing testing activities and result. It also contains an evaluation of the corresponding test items against exit criteria.

**Operational maintain testing:** validates whether the system meets the requirements for operation. (System admin jobs)

**Certification/Compliance testing:** is performed against the regulations of, such as governmental, legal or safety relations. (Certified testers job)

**Pilot/Field Testing**

Installation or Production Testing