'**Software**' refers to the set of electronic program instructions or data a computer processor reads in order to perform a task or operation.

There are two main types of software: **systems software** and **application software**.

**Systems Software**

**Systems software** includes the programs that are dedicated to managing the computer itself, such as the **operating system**, file management utilities, and disk operating system (or DOS). The operating system manages the computer hardware resources in addition to applications and data. Without systems software installed in our computers we would have to type the instructions for everything we wanted the computer to do

## Applications Software

**Application software**, or simply **applications**, are often called productivity programs or end-user programs because they enable the user to complete tasks, such as creating documents, spreadsheets, databases and publications, doing online research, sending email, designing graphics, running businesses, and even playing games! Application software is specific to the task it is designed for and can be as simple as a calculator application or as complex as a word processing application.

**The different types of application software include the following:**

| **Application Software Type** | **Examples** |
| --- | --- |
| Word processing software | MS Word, WordPad and Notepad |
| Database software | Oracle, MS Access etc |
| Spreadsheet software | Apple Numbers, Microsoft Excel |
| Multimedia software | Real Player, Media Player |
| Presentation Software | Microsoft Power Point, Keynotes |
| Enterprise Software | Customer relationship management system |
| Information Worker Software | Documentation tools, resource management tools |
| Educational Software | Dictionaries: Encarta, BritannicaMathematical: MATLABOthers: Google Earth, NASA World Wind |
| Simulation Software | Flight and scientific simulators |
| Content Access Software | Accessing content through media players, web browsers |
| Application Suites | OpenOffice, Microsoft Office |
| Software for Engineering and Product Development | **IDE** or Integrated Development Environments |

### What is Software Testing?

Software testing is defined as an activity to check whether the actual results match the expected results and to ensure that the software system is [Defect](https://www.guru99.com/the-unconventional-guide-to-defect-management.html)free.

Software testing also helps to identify errors, gaps or missing requirements in contrary to the actual requirements. Either it can be done manually or using automated tools.

## Why is Software Testing Important?

## Testing is important because software bugs could be expensive or even dangerous. Software bugs can potentially cause monetary and human loss

* In April 2015, Bloomberg terminal in London crashed due to software glitch affected more than 300,000 traders on financial markets. It forced the government to postpone a 3bn pound debt sale.
* Nissan cars have to recall over 1 million cars from the market due to software failure in the airbag sensory detectors. There has been reported two accident due to this software failure.
* Starbucks was forced to close about 60 percent of stores in the U.S and Canada due to software failure in its POS system. At one point store served coffee for free as they unable to process the transaction.
* Some of the Amazon’s third party retailers saw their product price is reduced to 1p due to a software glitch. They were left with heavy losses.

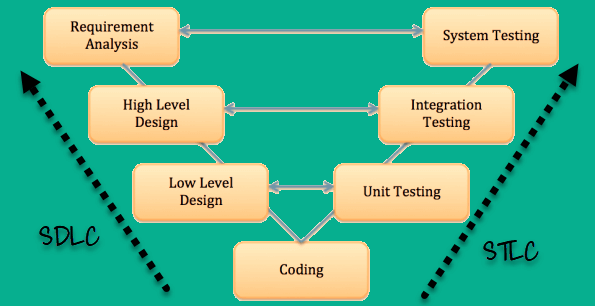
## Who does Testing?

* Software Tester
* Software Developer
* End User

## When to Start Testing?

Testing is done in different forms at every phase of SDLC −

* During the requirement gathering phase, the analysis and verification of requirements are also considered as testing.
* Reviewing the design in the design phase with the intent to improve the design is also considered as testing.
* Testing performed by a developer on completion of the code is also categorized as testing.

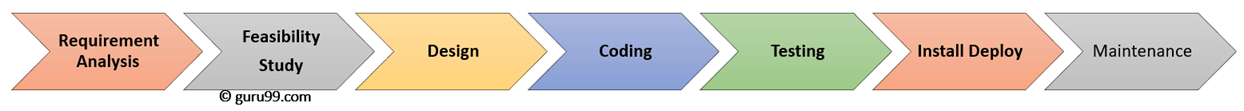


**What is SDLC?**

The Software Development Lifecycle is a systematic process for building software that ensures the quality and correctness of the software built. SDLC process aims to produce high-quality software which meets customer expectations.

## SDLC Phases

The entire SDLC process divided into the following stages:

[](https://www.guru99.com/images/1/080118_0641_SDLCSoftwar1.png)

* Phase 1: Requirement collection and analysis
* Phase 2: Feasibility study:
* Phase 3: Design:
* Phase 4: Coding:
* Phase 5: Testing:
* Phase 6: Installation/Deployment:
* Phase 7: Maintenance:

### Phase 1: Requirement collection and analysis:

The requirement is the first stage in the SDLC process. It is conducted by the senior team members with inputs from all the stakeholders and domain experts in the industry. Planning for the quality assurance requirements and recognization of the risks involved is also done at this stage.

This stage gives a clearer picture of the scope of the entire project and the anticipated issues, opportunities, and directives which triggered the project.

Requirements Gathering stage need teams to get detailed and precise requirements. This helps companies to finalize the necessary timeline to finish the work of that system.

### Phase 2: Feasibility study:

Once the requirement analysis phase is completed the next step is to define and document software needs. This process conducted with the help of 'Software Requirement Specification' document also known as 'SRS' document. It includes everything which should be designed and developed during the project life cycle.

**There are mainly five types of feasibilities checks:**

* **Economic:**Can we complete the project within the budget or not?
* **Legal:** Can we handle this project as cyber law and other regulatory framework/compliances.
* **Operation feasibility:** Can we create operations which is expected by the client?
* **Technical:** Need to check whether the current computer system can support the software
* **Schedule:** Decide that the project can be completed within the given schedule or not.

### Phase 3: Design:

In this third phase, the system and software design documents are prepared as per the requirement specification document. This helps define overall system architecture.

This design phase serves as input for the next phase of the model.

There are two kinds of design documents developed in this phase:

High-Level Design (HLD)

* Brief description and name of each module
* An outline about the functionality of every module
* Interface relationship and dependencies between modules
* Database tables identified along with their key elements
* Complete architecture diagrams along with technology details

Low-Level Design(LLD)

* Functional logic of the modules
* Database tables, which include type and size
* Complete detail of the interface
* Addresses all types of dependency issues
* Listing of error messages
* Complete input and outputs for every module

### Phase 4: Coding:

Once the system design phase is over, the next phase is coding. In this phase, developers start build the entire system by writing code using the chosen programming language. In the coding phase, tasks are divided into units or modules and assigned to the various developers. It is the longest phase of the Software Development Life Cycle process.

In this phase, Developer needs to follow certain predefined coding guidelines. They also need to use programming tools like compiler, interpreters, debugger to generate and implement the code

### Phase 5: Testing:

Once the software is complete, and it is deployed in the testing environment. The testing team starts testing the functionality of the entire system. This is done to verify that the entire application works according to the customer requirement.

During this phase, QA and testing team may find some bugs/defects which they communicate to developers. The development team fixes the bug and send back to QA for a re-test. This process continues until the software is bug-free, stable, and working according to the business needs of that system.

### Phase 6: Installation/Deployment:

Once the software testing phase is over and no bugs or errors left in the system then the final deployment process starts. Based on the feedback given by the project manager, the final software is released and checked for deployment issues if any.

### Phase 7: Maintenance:

Once the system is deployed, and customers start using the developed system, following 3 activities occur

* Bug fixing - bugs are reported because of some scenarios which are not tested at all
* Upgrade - Upgrading the application to the newer versions of the Software
* Enhancement - Adding some new features into the existing software

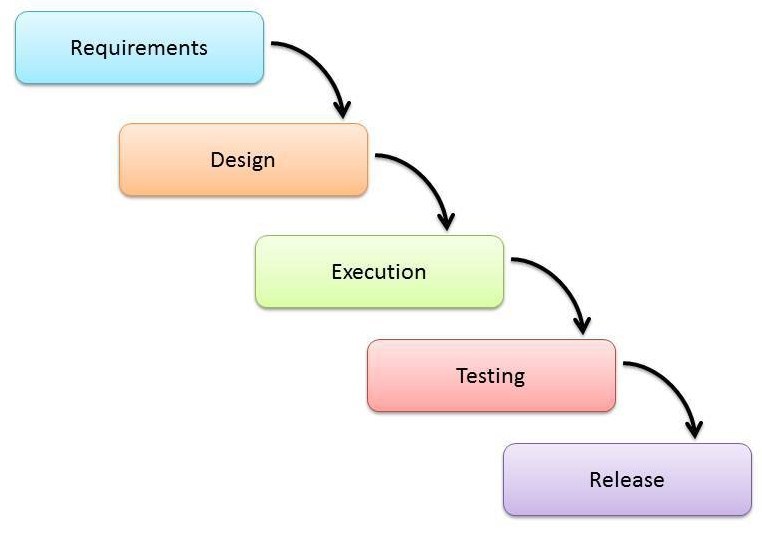
The main focus of this SDLC phase is to ensure that needs continue to be met and that the system continues to perform as per the specification mentioned in the first phase.

## Popular SDLC models

Here, are some most important phases of SDLC life cycle:

**Waterfall model**

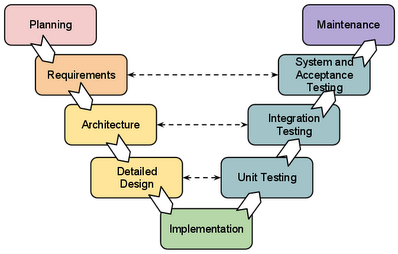
The [Waterfall Model](http://melsatar.blog/2018/02/16/the-waterfall-model-a-different-perspective/) is a linear sequential flow. In which progress is seen as flowing steadily downwards (like a waterfall) through the phases of software implementation. This means that any phase in the development process begins only if the previous phase is complete. The waterfall approach does not define the process to go back to the previous phase to handle changes in requirement. The waterfall approach is the earliest approach and most widely known that was used for software development.



|  |  |
| --- | --- |
| Advantages | Disadvantages |
| * Easy to explain to the users. * Structures approach. * Stages and activities are well defined. * Helps to plan and schedule the project. * Verification at each stage ensures early detection of errors/misunderstanding. * Each phase has specific deliverables. | * Assumes that the requirements of a system can be frozen. * Very difficult to go back to any stage after it finished. * A little flexibility and adjusting scope is difficult and expensive. * Costly and required more time, in addition to the detailed plan. |

**V-Model**

It is an extension of the waterfall model, Instead of moving down in a linear way, the process steps are bent upwards after the implementation and coding phase, to form the typical V shape. The major difference between the V-shaped model and waterfall model is the early test planning in the V-shaped model.



|  |  |
| --- | --- |
| Advantages | Disadvantages |
| * Simple and easy to use * Each phase has specific deliverables. * Higher chance of success over the waterfall model due to the development of test plans early on during the life cycle. * Works well for where requirements are easily understood. * Verification and validation of the product in the early stages of product development. | * Very inflexible, like the waterfall model. * Adjusting scope is difficult and expensive. * The software is developed during the implementation phase, so no early prototypes of the software are produced. * The model doesn’t provide a clear path for problems found during testing phases. * Costly and required more time, in addition to a detailed plan |

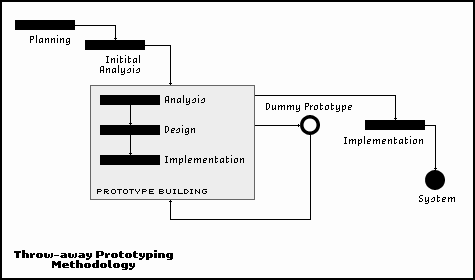
### **Prototyping Model**

#### **Description**

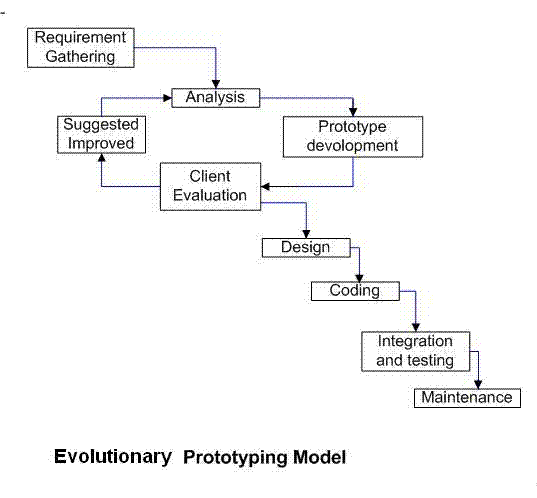
It refers to the activity of creating prototypes of software applications, for example, incomplete versions of the software program being developed. It is an activity that can occur in software development and It used to visualize some component of the software to limit the gap of misunderstanding the customer requirements by the development team. This also will reduce the iterations may occur in the waterfall approach and hard to be implemented due to the inflexibility of the waterfall approach. So, when the final prototype is developed, the requirement is considered to be frozen.

It has some types, such as:

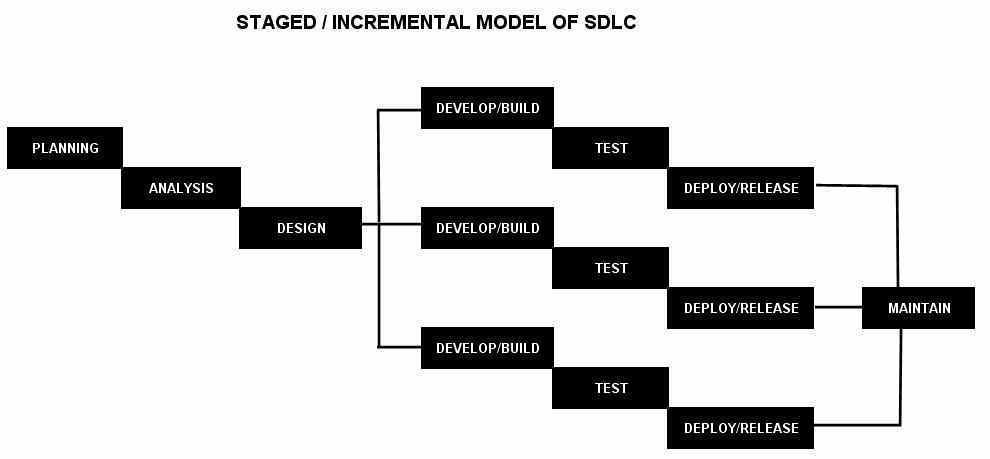
* Throwaway prototyping: Prototypes that are eventually discarded rather than becoming a part of the finally delivered software



* Evolutionary prototyping: prototypes that evolve into the final system through an iterative incorporation of user feedback.



* Incremental prototyping: The final product is built as separate prototypes. In the end, the separate prototypes are merged in an overall design.



* Extreme prototyping: used in web applications mainly. Basically, it breaks down web development into three phases, each one based on the preceding one. The first phase is a static prototype that consists mainly of HTML pages. In the second phase, the screens are programmed and fully functional using a simulated services layer. In the third phase, the services are implemented

#### **The usage**

* This process can be used with any software developing life cycle model. While this shall be chosen when you are developing a system has user interactions. So, if the system does not have user interactions, such as a system does some calculations shall not have prototypes.

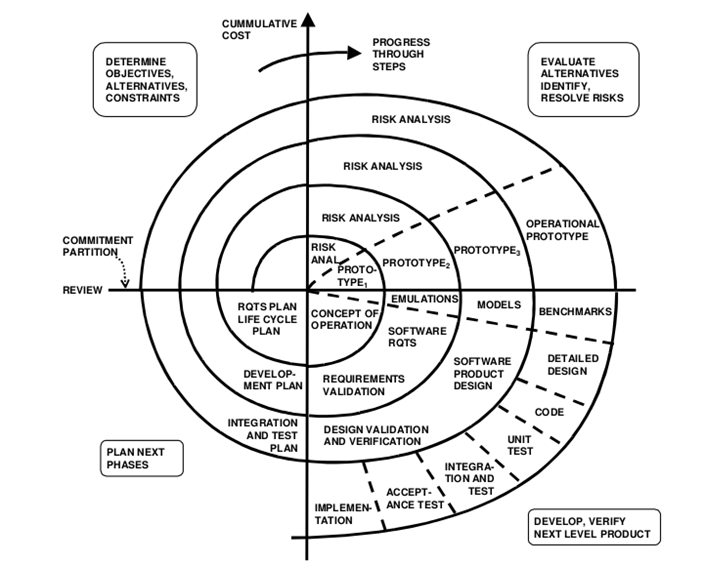
#### **Advantages and Disadvantages**

|  |  |
| --- | --- |
| Advantages | Disadvantages |
| * Reduced time and costs, but this can be a disadvantage if the developer loses time in developing the prototypes. * Improved and increased user involvement. | * Insufficient analysis. User confusion of prototype and finished system. * Developer misunderstanding of user objectives. * Excessive development time of the prototype. * It is costly to implement the prototypes |

### **Spiral Model (SDM)**

#### **Description**

It is combining elements of both design and prototyping-in-stages, in an effort to combine advantages of top-down and bottom-up concepts. This model of development combines the features of the prototyping model and the waterfall model. The spiral model is favored for large, expensive, and complicated projects. This model uses many of the same phases as the waterfall model, in essentially the same order, separated by planning, risk assessment, and the building of prototypes and simulations.



#### **The usage**

It is used in the large applications and systems which built-in small phases or segments.

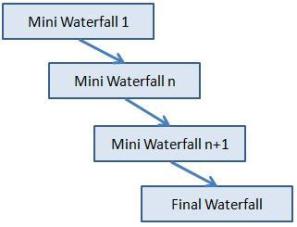
#### **Advantages and Disadvantages**

|  |  |
| --- | --- |
| Advantages | Disadvantages |
| * Estimates (i.e. budget, schedule, etc.) become more realistic as work progressed because important issues are discovered earlier. * Early involvement of developers. * Manages risks and develops the system into phases. | * High cost and time to reach the final product. * Needs special skills to evaluate the risks and assumptions. * Highly customized limiting re-usability |

### **Iterative and Incremental Model**

#### **Description**

It is developed to overcome the weaknesses of the waterfall model. It starts with an initial planning and ends with deployment with the cyclic interactions in between. The basic idea behind this method is to develop a system through repeated cycles (iterative) and in smaller portions at a time (incremental), allowing software developers to take advantage of what was learned during the development of earlier parts or versions of the system. It can consist of mini waterfalls or mini V-Shaped model



#### **The usage**

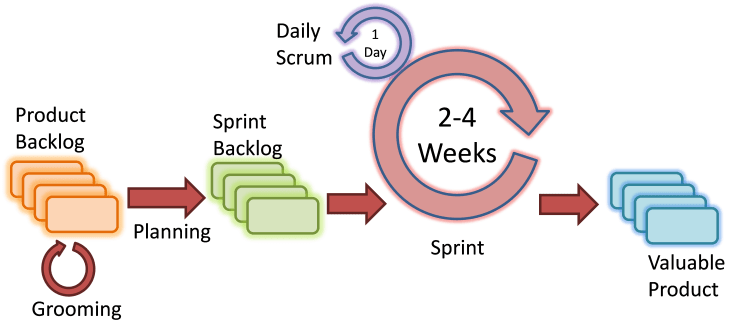
It is used in shrink-wrap application and large system which built-in small phases or segments. Also, can be used in a system has separated components, for example, ERP system. Which we can start with the budget module as a first iteration and then we can start with the inventory module and so forth.

|  |  |
| --- | --- |
| Advantages | Disadvantages |
| * Produces business value early in the development lifecycle. * Better use of scarce resources through proper increment definition. * Can accommodate some change requests between increments. * More focused on customer value than the linear approaches. * We can detect project issues and changes earlier. | * Requires heavy documentation. * Follows a defined set of processes. * Defines increments based on function and feature dependencies. * Requires more customer involvement than the linear approaches. * Partitioning the functions and features might be problematic. * Integration between the iterations can be an issue if it is not considered during the development and project planning. |

### **Agile Model**

#### **Description**

It is based on iterative and incremental development, where requirements and solutions evolve through collaboration between cross-functional teams.



Scrum Agile Model

#### **The usage**

It can be used with any type of the project, but it needs more engagement from the customer and to be interactive. Also, we can use it when the customer needs to have some functional requirement ready in less than three weeks and the requirements are not clear enough. This will enable more valuable and workable piece for software early which also increase the customer satisfaction.

#### **Advantages and Disadvantages**

|  |  |
| --- | --- |
| Advantages | Disadvantages |
| * Decrease the time required to avail some system features. * Face to face communication and continuous inputs from customer representative leaves no space for guesswork. * The end result is the high-quality software in the least possible time duration and satisfied customer. | * Scalability. * The ability and collaboration of the customer to express user needs. * Documentation is done at later stages. * Reduce the usability of components. * Needs special skills for the team. |

Scrum is a part of agile frameworks that has taken many industries by storm during the last few years

The main objective of Scrum is breaking down the work in such a way that it can maximize efficiency and reduce bottlenecks while moving towards project completion and customer satisfaction. The different Scrum roles within an organization include Scrum Master, Product Owner, Scrum Team etc.

The Scrum team is defined as the set of individuals working on a project, the product owner is the person who is responsible for designing different sections of the workflows and the Scrum Master facilitates both scrum team and the product owner in implementing the established work process.

Scrum framework makes sure that everyone is working in sync with the project deliverables and completely understands the milestones to be achieved. It encourages customer involvement at every stage, sets the project timelines in form of Sprints or we can say Daily Scrums. Each Sprint is given a time period when particular tasks assigned by Product Owners should be completed. The average time span for a Sprint is 7 days to one month or may also depend on the client requirements.

A few development teams focus more on daily scrums, stand-up meetings among the team members, product owner, scrum master, customers, and management etc. It helps to evaluate tasks completed on daily basis with hindrances and potential risks.



### The Scrum Events

Prescribed events are used in Scrum to create regularity and to minimize the need for meetings not defined in Scrum. All events are time-boxed. Once a Sprint begins, its duration is fixed and cannot be shortened or lengthened. The remaining events may end whenever the purpose of the event is achieved, ensuring an appropriate amount of time is spent without allowing waste in the process.  The Scrum Events are:

* 1. Sprint
  2. Sprint Planning
  3. Daily Scrum
  4. Sprint Review
  5. Sprint Retrospective

**Sprint**, a time-box of one month or less during which a “Done”, useable, and potentially releasable product Increment is created. Sprints have consistent durations throughout a development effort. A new Sprint starts immediately after the conclusion of the previous Sprint.

During the Sprint:

* No changes are made that would endanger the Sprint Goal;
* Quality goals do not decrease; and,
* Scope may be clarified and re-negotiated between the Product owner and development team as more is learned.

**Sprint Planning** is time-boxed to a maximum of eight hours for a one-month Sprint. For shorter Sprints, the event is usually shorter. The Scrum master ensures that the event takes place and that attendants understand its purpose. The Scrum Master teaches the Scrum Team to keep it within the time-box.

Sprint Planning answers the following:

* What can be delivered in the Increment resulting from the upcoming Sprint?
* How will the work needed to deliver the Increment be achieved?

Work is selected from the Product Backlog and pulled into the Sprint Backlog.  Now remember that the work in the Sprint Backlog is not a commitment, it is a forecast.  The only container of a Sprint is its time box, not the work planned for the Sprint.

### **Sprint Goal**

The Sprint Goal is an objective set for the Sprint that can be met through the implementation of Product Backlog. It provides guidance to the Development Team on why it is building the Increment. It is created during the Sprint Planning meeting. The Sprint Goal gives the Development Team some flexibility regarding the functionality implemented within the Sprint. As the Development Team works, it does so with the Sprint Goal always in mind.

**Daily Scrum** is a 15-minute time-boxed event for the Development Team to synchronize activities and create a plan for the next 24 hours. The Daily Scrum is held every day of the Sprint. At it, the Development Team plans work for the next 24 hours. This optimizes team collaboration and performance by inspecting the work since the last Daily Scrum and forecasting upcoming Sprint work. The Daily Scrum is held at the same time and place each day to reduce complexity.

The Development Team uses the Daily Scrum to inspect progress toward the Sprint Goal and to inspect how progress is trending toward completing the work in the Sprint Backlog. The Daily Scrum optimizes the probability that the Development Team will meet the Sprint Goal. Every day, the Development Team should understand how it intends to work together as a self-organizing team to accomplish the Sprint Goal and create the anticipated Increment by the end of the Sprint. The Development Team or team members often meet immediately after the Daily Scrum for detailed discussions, or to adapt, or replan, the rest of the Sprint’s work.

Daily Scrums improve communications, eliminate other meetings, identify impediments to development for removal, highlight and promote quick decision-making, and improve the Development Team’s level of knowledge. This is a key inspect and adapt meeting.

The structure of the meeting is set by the Development Team and can be conducted in different ways if it focuses on progress toward the Sprint Goal. Some Development Teams will use questions, some will be more discussion based.

The Scrum Master ensures that the Development Team has the meeting, but the Development Team is responsible for conducting the Daily Scrum.

#### The Daily Scrum is Not a Status Meeting

**Sprint Review** is held at the end of the Sprint to inspect the Increment and adapt the Product Backlog if needed

During the Sprint Review, the Scrum Team and stakeholders collaborate about what was done in the Sprint. Based on that and any changes to the Product Backlog during the Sprint, attendees collaborate on the next things that could be done to optimize value. This is an informal meeting, not a status meeting, and the presentation of the Increment is intended to elicit feedback and foster collaboration.

This is at most a four-hour meeting for one-month Sprints. For shorter Sprints, the event is usually shorter. The Scrum Master ensures that the event takes place and that attendees understand its purpose. The Scrum Master teaches everyone involved to keep it within the time-box.

The Sprint Review includes the following elements:

* Attendees include the Scrum Team and key stakeholders invited by the Product Owner;
* The Product Owner explains what Product Backlog items have been “Done” and what has not been “Done”;
* The Development Team discusses what went well during the Sprint, what problems it ran into, and how those problems were solved;
* The Development Team demonstrates the work that it has “Done” and answers questions about the Increment;
* The Product Owner discusses the Product Backlog as it stands. He or she projects likely target and delivery dates based on progress to date (if needed);
* The entire group collaborates on what to do next, so that the Sprint Review provides valuable input to subsequent Sprint Planning.
* Review of how the marketplace or potential use of the product might have changed what is the most valuable thing to do next; and,
* Review of the timeline, budget, potential capabilities, and marketplace for the next anticipated releases of functionality and capability of the product.

The result of the Sprint Review is a revised Product Backlog that defines the probable Product Backlog items for the next Sprint. The Product Backlog may also be adjusted overall to meet new opportunities.

**Sprint Retrospective** is an opportunity for the Scrum Team to inspect itself and create a plan for improvements to be enacted during the next Sprint.

The Sprint Retrospective occurs after the Sprint Review and prior to the next Sprint Planning. This is at most a three-hour meeting for one-month Sprints. For shorter Sprints, the event is usually shorter. The Scrum Master ensures that the event takes place and that attendants understand its purpose. This is the opportunity for the Scrum Team to improve and all member should be in attendance.

During the Sprint Retrospective, the team discusses:

* What went well in the Sprint
* What could be improved
* What will we commit to improve in the next Sprint

The Scrum Master encourages the Scrum Team to improve its development process and practices to make it more effective and enjoyable for the next Sprint. During each Sprint Retrospective, the Scrum Team plans ways to increase product quality by improving work processes or adapting the definition of “Done” if appropriate and not in conflict with product or organizational standards.

By the end of the Sprint Retrospective, the Scrum Team should have identified improvements that it will implement in the next Sprint. Implementing these improvements in the next Sprint is the adaptation to the inspection of the Scrum Team itself. Although improvements may be implemented at any time, the Sprint Retrospective provides a formal opportunity to focus on inspection and adaptation

### **Scrum Artifacts**

Scrum’s artifacts represent work or value to provide transparency and opportunities for inspection and adaptation. Artifacts defined by Scrum are specifically designed to maximize transparency of key information so that everybody has the same understanding of the artifact. The Scrum Artifacts are:

* Product Backlog
* Sprint Backlog
* Increment

**Product Backlog** is an ordered list of everything that is known to be needed in the product. It is the single source of requirements for any changes to be made to the product. The Product Owner is responsible for the Product Backlog, including its content, availability, and ordering.

**Sprint Backlog** is the set of Product Backlog items selected for the Sprint, plus a plan for delivering the product Increment and realizing the Sprint Goal.

**Increment** is the sum of all the Product Backlog items completed during a Sprint and the value of the increments of all previous Sprints. At the end of a Sprint, the new Increment must be “Done,” which means it must be in useable condition and meet the Scrum Team’s definition of “Done.”  An increment is a body of inspectable, done work that supports empiricism at the end of the Sprint. The increment is a step toward a vision or goal.  The increment must be in useable condition regardless of whether the Product Owner decides to release it.

### What is Software Testing Life Cycle (STLC)?

Software Testing Life Cycle (STLC) is defined as a sequence of activities conducted to perform Software Testing.

**Software Test Life Cycle** has the following stages.



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

### What is Entry and Exit 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

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 RTM
* 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

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. In this phase, Test Strategy is also determined.

**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/ strategy document.
* Effort Estimation document.

## Test Case Development

This phase involves the creation, verification and rework of test cases & test scripts. Test data 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, the testers 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 the 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 the 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 the 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

**Verification & Validation:**

|  |  |  |
| --- | --- | --- |
| **Sr.No.** | **Verification** | **Validation** |
| 1 | Verification addresses the concern: "Are you building it right?" | Validation addresses the concern: "Are you building the right thing?" |
| 2 | Ensures that the software system meets all the functionality. | Ensures that the functionalities meet the intended behaviour. |
| 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 | It has static activities, as it includes collecting reviews, walkthroughs, and inspections to verify a software. | It has 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 a software. | It is a subjective process and involves subjective decisions on how well a software works. |

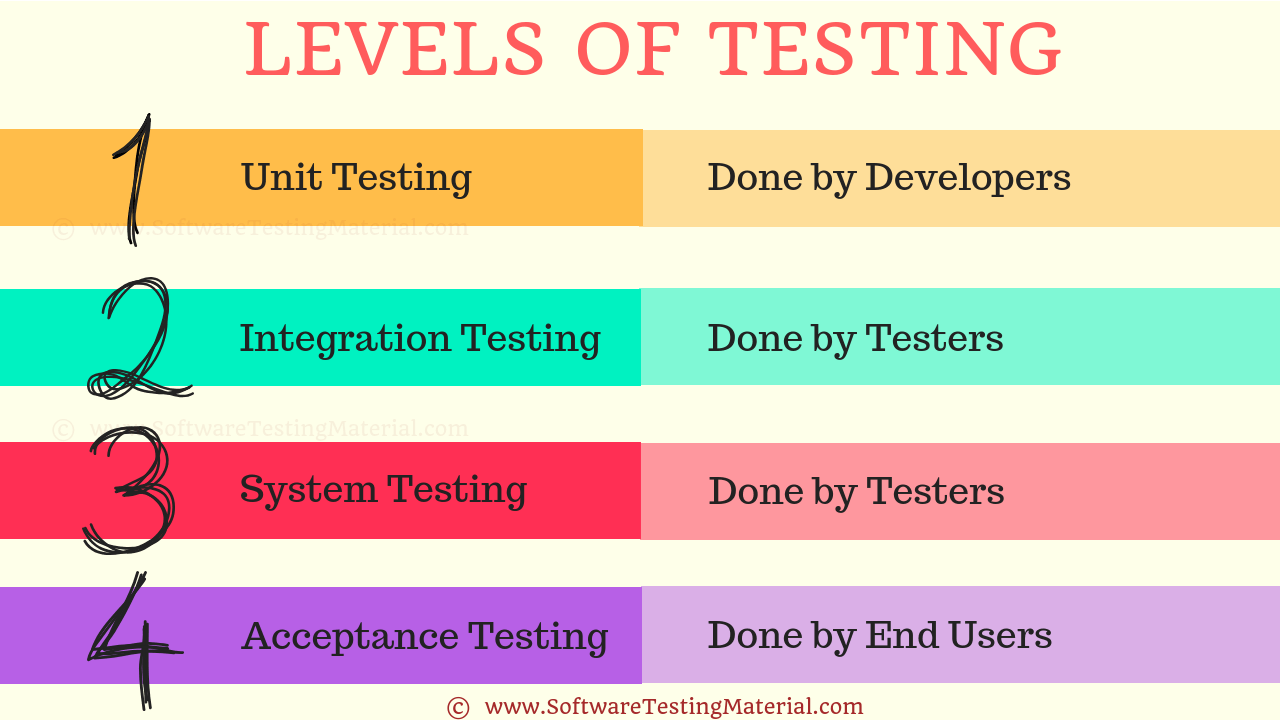
## When to Stop Testing?

It is difficult to determine when to stop testing, as testing is a never-ending process and no one can claim that a software is 100% tested. The following aspects are to be considered for stopping the testing process −

* Testing Deadlines
* Completion of test case execution
* Completion of functional and code coverage to a certain point
* Bug rate falls below a certain level and no high-priority bugs are identified
* Management decision

1. **Levels Of Testing**

|  |  |
| --- | --- |
| **Testing Category** | **Types of Testing** |
| Functional Testing |  [Unit Testing](https://www.guru99.com/unit-testing-guide.html)   [Integration Testing](https://www.guru99.com/integration-testing.html)   Smoke   UAT ( User Acceptance Testing)   Localization   Globalization   Interoperability   So on |
| Non-Functional Testing |  Performance   Endurance   Load   Volume   Scalability   Usability   So on |
| Maintenance |  Regression   Maintenance |



### **UNIT TESTING:**

Unit Testing is done to check whether the individual modules of the source code are working properly. i.e. testing each and every unit of the application separately by the developer in the developer’s environment.

### **INTEGRATION TESTING:**

Integration Testing is the process of testing the connectivity or data transfer between a couple of unit tested modules

# **Levels of Testing | Software Testing Material**

Last Updated on May 21, 2019 by [Rajkumar](https://www.softwaretestingmaterial.com/author/smrajkumar27gmail-com/) [2 Comments](https://www.softwaretestingmaterial.com/levels-of-testing/#comments)

Levels of Testing!! Before starting the post on Levels of Testing, let’s see what is Software Testing.

In Software development, both developers and testers work together to release a high-quality product. To release a high-quality product, every product goes through various testing processes. Coming to testing, testers use various levels of testing in the process of releasing a quality product. There are different levels of software testing. Each of these levels of software testing has a specific purpose. We will see each software testing level in detail.

### **What is Software Testing?**

**Software testing** is a process, to evaluate the functionality of a software application with an intent to find whether the developed software met the specified requirements or not and to identify the defects to ensure that the product is defect free in order to produce the quality product.

Learn more:

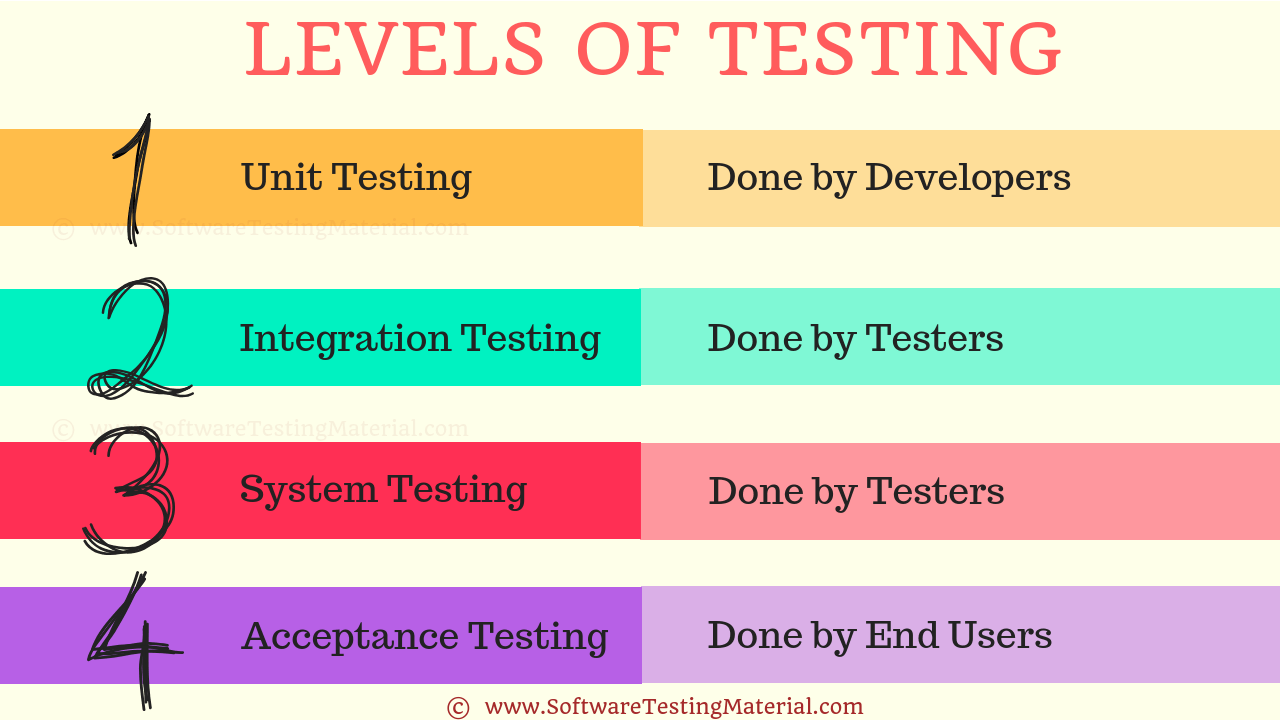
[Software Testing – Definition, Types, Methods & Approach](https://www.softwaretestingmaterial.com/software-testing/)

### **Levels of Software Testing:**

Let’s see what are the levels of software testing:

Different levels of software testing are as follows.

1. Unit Testing  
2. Integration Testing  
3. System Testing  
4. Acceptance Testing



### **Let’s see the levels of testing in detail.**

### **UNIT TESTING:**

Unit Testing is done to check whether the individual modules of the source code are working properly. i.e. testing each and every unit of the application separately by the developer in the developer’s environment. It is AKA Module Testing or Component Testing

### **INTEGRATION TESTING:**

Integration Testing is the process of testing the connectivity or data transfer between a couple of unit tested modules. It is AKA I&T Testing or String Testing

It is subdivided into the Top-Down Approach, Bottom-Up Approach and Sandwich Approach (Combination of Top Down and Bottom Up). This process is carried out by using dummy programs called Stubs and Drivers. Stubs and Drivers do not implement the entire programming logic of the software module but just simulate data communication with the calling module.

#### **Big Bang Integration Testing:**

In Big Bang Integration Testing, the individual modules are not integrated until all the modules are ready. Then they will run to check whether it is performing well. In this type of testing, some disadvantages might occur like, defects can be found at the later stage. It would be difficult to find out whether the defect arouses in an interface or in a module.

#### **Top-Down Integration Testing**

In Top-Down Integration Testing, the high-level modules are integrated and tested first. i.e Testing from the main module to the submodule. In this type of testing, Stubs are used as a temporary module if a module is not ready for integration testing.

#### **Bottom-Up Integration Testing**

In Bottom Up Integration Testing, the low-level modules are integrated and tested first i.e Testing from sub-module to the main module. Same like Stubs, here drivers are used as a temporary module for integration testing.

#### **Stub:**

It is called by the Module under Test.

#### **Driver:**

It calls the Module to be tested.

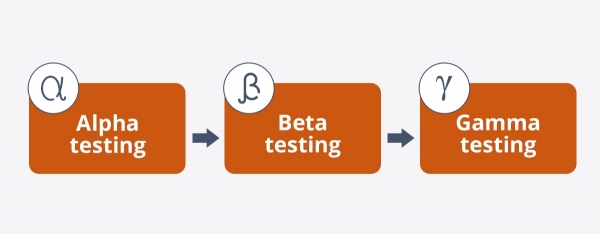
### **SYSTEM TESTING (END TO END TESTING):**

It’s a black box testing. Testing the fully integrated application this is also called as an end to end scenario testing. To ensure that the software works in all intended target systems. Verify thorough testing of every input in the application to check for desired outputs. Testing of the users’ experiences with the application.

### **ACCEPTANCE TESTING:**

To obtain customer sign-off so that software can be delivered and payments received.

There are three phases of software testing - alpha, beta, and gamma. They are performed one after another, and together ensures a release of high-quality software.



## Alpha Testing

**Alpha testing** is an internal checking done by in-house development or QA team, rarely, by the customer himself. Its main purpose is to discover software bugs that were not found before. At the stage of alpha testing, software behavior is verified under real-life conditions by imitating the end users’ actions. It enables to get the fast approval from the customer before proceeding to product delivery.

The alpha phase includes the following testing types: smoke, sanity, integration, systems, usability, UI (user interface), acceptance, regression, and [functional testing](https://qatestlab.com/services/manual-testing/functional-testing/#functional%20testing). If an error is detected, then it is immediately addressed to the development team. Alpha testing helps to discover issues missed at the stage of requirement gathering. Alpha release is the software version that has passed alpha testing. The next stage is beta testing.

## Beta Testing

**Beta testing** can be called pre-release testing. It can be conducted by a limited number of end users called [beta testers](https://qatestlab.com/resources/knowledge-center/characteristics-of-beta-tester-profession/) before the official product delivery. The main purpose of beta testing is to verify software compatibility with different software and hardware configurations, types of network connection, and to get the users’ feedback on software usability and functionality.

There are two types of beta testing:

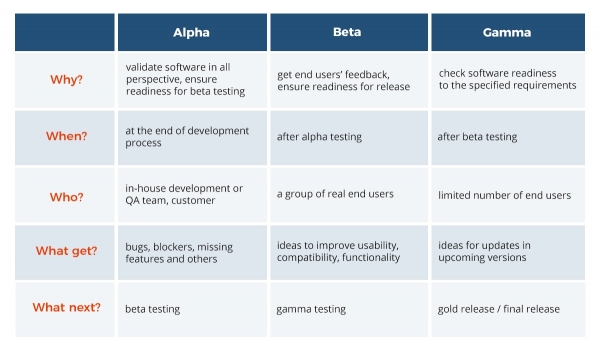
* **open beta** is available for a large group of end users or to everyone interested
* **closed beta** is available only to a limited number of users that are selected especially for beta testing.

During beta testing, end users detect and report bugs they have found. All the testing activities are performed outside the organization that has developed the product. Beta checking helps to identify the gaps between the stage of requirements gathering and their implementation. The product version that has passed beta testing is called beta release. After beta phase comes gamma testing.

Gamma Testing

**Gamma testing** is the final stage of the testing process conducted before software release. It makes sure that the product is ready for market release according to all the specified requirements. Gamma testing focuses on software security and functionality. But it does not include any in-house QA activities. During gamma testing, software does not undergo any modifications unless the detected bug is of a high priority and severity.

Only a limited number of users perform gamma testing, and testers do not participate. The checking includes the verification of certain specification, not the whole product. Feedback received after gamma testing are considered as updates for upcoming software versions. But, because of a limited development cycle, gamma testing is usually skipped.



### **Different Types of Software Testing**

**Functional testing types include:**

* Unit testing
* Integration testing
* System testing
* Sanity testing
* Smoke testing
* Interface testing
* Regression testing
* Beta/Acceptance testing

**Non-functional testing types include:**

* Performance Testing
* Load testing
* Stress testing
* Volume testing
* Security testing
* Compatibility testing
* Install testing
* Recovery testing
* Reliability testing
* Usability testing
* Compliance testing
* Localization testing

#### **#1) Alpha Testing**

It is the most common type of testing used in the Software industry. The objective of this testing is to identify all possible issues or defects before releasing it into the market or to the user.

Alpha testing is carried out at the end of the software development phase but before the Beta Testing. Still, minor design changes may be made as a result of such testing. [Alpha testing](https://www.softwaretestinghelp.com/what-is-alpha-testing-beta-testing/) is conducted at the developer’s site. In-house virtual user environment can be created for this type of testing.

#### **#2) Acceptance Testing**

An [acceptance test](https://www.softwaretestinghelp.com/what-is-acceptance-testing/) is performed by the client and verifies whether the end to end the flow of the system is as per the business requirements or not and if it is as per the needs of the end user. Client accepts the software only when all the features and functionalities work as expected.

It is the last phase of the testing, after which the software goes into production. This is also called User Acceptance Testing (UAT).

#### **#3) Ad-hoc Testing**

The name itself suggests that this testing is performed on [an ad-hoc](https://www.softwaretestinghelp.com/ad-hoc-testing/) basis i.e. with no reference to the test case and also without any plan or documentation in place for such type of testing. The objective of this testing is to find the defects and break the application by executing any flow of the application or any random functionality.

Ad-hoc testing is an informal way of finding defects and can be performed by anyone in the project. It is difficult to identify defects without a test case but sometimes it is possible that defects found during ad-hoc testing might not have been identified using existing test cases.

#### **#4) Accessibility Testing**

The aim of [accessibility testing](https://www.softwaretestinghelp.com/what-is-web-accessibility-testing/) is to determine whether the software or application is accessible for disabled people or not. Here disability means deaf, color blind, mentally disabled, blind, old age and other disabled groups. Various checks are performed such as font size for visually disabled, color and contrast for color blindness etc.

#### **#5) Beta Testing**

[Beta Testing](https://www.softwaretestinghelp.com/beta-testing/) is a formal type of software testing which is carried out by the customer. It is performed in **the Real Environment**before releasing the product to the market for the actual end users.

Beta testing is carried out to ensure that there are no major failures in the software or product and it satisfies the business requirements from an end-user perspective. Beta testing is successful when the customer accepts the software.

Usually, this testing is typically done by end-users or others. It is the final testing done before releasing an application for commercial purpose. Usually, the Beta version of the software or product released is limited to a certain number of users in a specific area.

So end user actually uses the software and shares the feedback to the company. Company then takes necessary action before releasing the software to the worldwide.

#### **#6) Back-end Testing**

Whenever an input or data is entered on front-end application, it stores in the database and the testing of such database is known as Database Testing or Backend testing. There are different databases like SQL Server, MySQL, and Oracle etc. Database testing involves testing of table structure, schema, stored procedure, data structure and so on.

In back-end testing GUI is not involved, testers are directly connected to the database with proper access and testers can easily verify data by running a few queries on the database. There can be issues identified like data loss, deadlock, data corruption etc during this back-end testing and these issues are critical to fixing before the system goes live into the production environment

#### **#7) Browser Compatibility Testing**

It is a subtype of Compatibility Testing (which is explained below) and is performed by the testing team.

[Browser Compatibility Testing](https://www.softwaretestinghelp.com/how-is-cross-browser-testing-performed/) is performed for web applications and it ensures that the software can run with the combination of different browser and operating system. This type of testing also validates whether web application runs on all versions of all browsers or not.

#### **#8) Backward Compatibility Testing**

It is a type of testing which validates whether the newly developed software or updated software works well with older version of the environment or not.

Backward Compatibility Testing checks whether the new version of the software works properly with file format created by older version of the software; it also works well with data tables, data files, data structure created by older version of that software. If any of the software is updated then it should work well on top of the previous version of that software.

#### **#9) Black Box Testing**

Internal system design is not considered in this type of testing. Tests are based on the requirements and functionality.

Detailed information about the advantages, disadvantages, and [types of Black box testing](https://www.softwaretestinghelp.com/black-box-testing/) can be seen here.

#### **#10) Boundary Value Testing**

This type of testing checks the behavior of the application at the boundary level.

[Boundary value Testing](https://www.softwaretestinghelp.com/what-is-boundary-value-analysis-and-equivalence-partitioning/) is performed for checking if defects exist at boundary values. Boundary value testing is used for testing a different range of numbers. There is an upper and lower boundary for each range and testing is performed on these boundary values.

If testing requires a test range of numbers from 1 to 500 then Boundary Value Testing is performed on values at 0, 1, 2, 499, 500 and 501.

#### **#11) Branch Testing**

It is a type of white box testing and is carried out during unit testing. Branch Testing, the name itself suggests that the code is tested thoroughly by traversing at every branch.

#### **#12) Comparison Testing**

Comparison of a product's strength and weaknesses with its previous versions or other similar products is termed as Comparison Testing.

#### **#13) Compatibility Testing**

It is a testing type in which it validates how software behaves and runs in a different environment, web servers, hardware, and network environment. [Compatibility testing](https://www.softwaretestinghelp.com/software-compatibility-testing/) ensures that software can run on a different configuration, different database, different browsers, and their versions. Compatibility testing is performed by the testing team.

#### **#14) Component Testing**

It is mostly performed by developers after the completion of unit testing. [Component Testing](https://www.softwaretestinghelp.com/what-is-component-testing-or-module-testing/) involves testing of multiple functionalities as a single code and its objective is to identify if any defect exists after connecting those multiple functionalities with each other.

#### **#15) End-to-End Testing**

Similar to system testing, [End-to-end testing](https://www.softwaretestinghelp.com/what-is-end-to-end-testing/) involves testing of a complete application environment in a situation that mimics real-world use, such as interacting with a database, using network communications, or interacting with other hardware, applications, or systems if appropriate.

#### **#16) Equivalence Partitioning**

It is a testing technique and a type of Black Box Testing. During this [equivalence partitioning](https://www.softwaretestinghelp.com/what-is-boundary-value-analysis-and-equivalence-partitioning/), a set of group is selected and a few values or numbers are picked up for testing. It is understood that all values from that group generate the same output.

The aim of this testing is to remove redundant test cases within a specific group which generates the same output but not any defect.

Suppose, application accepts values between -10 to +10 so using equivalence partitioning the values picked up for testing are zero, one positive value, one negative value. So the Equivalence Partitioning for this testing is: -10 to -1, 0, and 1 to 10.

#### **#17) Example Testing**

It means real-time testing. Example testing includes the real-time scenario, it also involves the scenarios based on the experience of the testers.

#### **#18) Exploratory Testing**

Exploratory Testing is informal testing performed by the testing team. The objective of this testing is to explore the application and looking for defects that exist in the application. Sometimes it may happen that during this testing major defect discovered can even cause system failure.

During exploratory testing, it is advisable to keep a track of what flow you have tested and what activity you did before the start of the specific flow.

[An exploratory testing technique](https://www.softwaretestinghelp.com/what-is-exploratory-testing/) is performed without documentation and test cases.

#### **#20) Functional Testing**

This type of testing ignores the internal parts and focuses only on the output to check if it is as per the requirement or not. It is a Black-box type testing geared to the functional requirements of an application. For detailed information about Functional Testing click [here](https://www.softwaretestinghelp.com/guide-to-functional-testing/).

#### **#21) Graphical User Interface (GUI) Testing**

The objective of this GUI testing is to validate the GUI as per the business requirement. The expected GUI of the application is mentioned in the Detailed Design Document and GUI mockup screens.

The GUI testing includes the size of the buttons and input field present on the screen, alignment of all text, tables and content in the tables.

It also validates the menu of the application, after selecting different menu and menu items, it validates that the page does not fluctuate and the alignment remains same after hovering the mouse on the menu or sub-menu.

#### **#22) Gorilla Testing**

Gorilla Testing is a testing type performed by a tester and sometimes by developer the as well. In Gorilla Testing, one module or the functionality in the module is tested thoroughly and heavily. The objective of this testing is to check the robustness of the application.

#### **#23) Happy Path Testing**

The objective of Happy Path Testing is to test an application successfully on a positive flow. It does not look for negative or error conditions. The focus is only on the valid and positive inputs through which application generates the expected output.

#### **#24) Incremental Integration Testing**

[Incremental Integration Testing](https://www.softwaretestinghelp.com/incremental-testing/) is a Bottom-up approach for testing i.e continuous testing of an application when a new functionality is added. Application functionality and modules should be independent enough to test separately. This is done by programmers or by testers.

#### **#25) Install/Uninstall Testing**

[Installation and uninstallation testing](https://www.softwaretestinghelp.com/software-installationuninstallation-testing/) is done on full, partial, or upgrade install/uninstall processes on different operating systems under different hardware or software environment.

#### **#26) Integration Testing**

Testing of all integrated modules to verify the combined functionality after integration is [termed as Integration Testing](https://www.softwaretestinghelp.com/what-is-integration-testing/). Modules are typically code modules, individual applications, client and server applications on a network, etc. This type of testing is especially relevant to client/server and distributed systems.

#### **#27) Load Testing**

It is a type of non-functional testing and the objective of Load testing is to check how much of load or maximum workload a system can handle without any performance degradation.

[Load testing helps](https://www.softwaretestinghelp.com/introduction-to-performance-testing-loadrunner-training-tutorial-part-1/) to find the maximum capacity of the system under specific load and any issues that cause the software performance degradation. Load testing is performed using tools like[JMeter](https://www.softwaretestinghelp.com/jmeter-tutorials/), LoadRunner, WebLoad, Silk performer etc.

#### **#28) Monkey Testing**

[Monkey testing](https://www.softwaretestinghelp.com/what-is-monkey-testing-in-software-testing/) is carried out by a tester assuming that if the monkey uses the application then how random input, values will be entered by the Monkey without any knowledge or understanding of the application.

The objective of Monkey Testing is to check if an application or system gets crashed by providing random input values/data. Monkey Testing is performed randomly and no test cases are scripted and it is not necessary to

Monkey Testing is performed randomly and no test cases are scripted and it is not necessary to be aware of the full functionality of the system.

#### **#29) Mutation Testing**

[Mutation Testing](https://www.softwaretestinghelp.com/what-is-mutation-testing/) is a type of white box testing in which the source code of one of the program is changed and verifies whether the existing test cases can identify these defects in the system. The change in the program source code is very minimal so that it does not impact the entire application, only the specific area having the impact and the related test cases should able to identify those errors in the system.

#### **#30) Negative Testing**

Testers having the mindset of “attitude to break” and using negative testing they validate that if system or application breaks. [A negative testing technique](https://www.softwaretestinghelp.com/what-is-negative-testing/) is performed using incorrect data, invalid data or input. It validates that if the system throws an error of invalid input and behaves as expected.

#### **#31) Non-Functional Testing**

It is a type of testing for which every organization having a separate team which usually called as Non-Functional Test (NFT) team or Performance team.

[Non-functional testing involves](https://www.softwaretestinghelp.com/what-is-non-functional-testing/) testing of non-functional requirements such as Load Testing, Stress Testing, Security, Volume, Recovery Testing etc. The objective of NFT testing is to ensure whether the response time of software or application is quick enough as per the business requirement.

It should not take much time to load any page or system and should sustain during peak load.

#### **#32) Performance Testing**

This term is often used interchangeably with ‘stress' and ‘load' testing. [Performance Testing](https://www.softwaretestinghelp.com/introduction-to-performance-testing-loadrunner-training-tutorial-part-1/) is done to check whether the system meets the performance requirements. Different performance and load tools are used to do this testing.

#### **#33) Recovery Testing**

It is a type of testing which validates that how well the application or system recovers from crashes or disasters.

Recovery testing determines if the system is able to continue the operation after a disaster. Assume that application is receiving data through the network cable and suddenly that network cable has been unplugged.

Sometime later, plug the network cable; then the system should start receiving data from where it lost the connection due to network cable unplugged.

#### **#34) Regression Testing**

Testing an application as a whole for the modification in any module or functionality is termed as Regression Testing. It is difficult to cover all the system in [Regression Testing](https://www.softwaretestinghelp.com/regression-testing-tools-and-methods/), so typically [automation testing tools](https://www.softwaretestinghelp.com/automation-testing-tutorial-1/) are used for these types of testing.

#### **#35) Risk-Based Testing (RBT)**

In [Risk Based Testing](https://www.softwaretestinghelp.com/risk-management-during-test-planning-risk-based-testing/), the functionalities or requirements are tested based on their priority. Risk-based testing includes testing of highly critical functionality, which has the highest impact on business and in which the probability of failure is very high.

The priority decision is based on the business need, so once priority is set for all functionalities then high priority functionality or test cases are executed first followed by medium and then low priority functionalities.

The low priority functionality may be tested or not tested based on the available time.

The Risk-based testing is carried out if there is insufficient time available to test entire software and software needs to be implemented on time without any delay. This approach is followed only by the discussion and approval of the client and senior management of the organization.

#### **#36) Sanity Testing**

[Sanity Testing](https://www.softwaretestinghelp.com/smoke-testing-and-sanity-testing-difference/) is done to determine if a new software version is performing well enough to accept it for a major testing effort or not. If an application is crashing for the initial use then the system is not stable enough for further testing. Hence a build or an application is assigned to fix it.

#### **#37) Security Testing**

It is a type of testing performed by a special team of testers. A system can be penetrated by any hacking way.

[Security Testing](https://www.softwaretestinghelp.com/how-to-test-application-security-web-and-desktop-application-security-testing-techniques/) is done to check how the software or application or website is secure from internal and external threats. This testing includes how much software is secure from the malicious program, viruses and how secure and strong the authorization and authentication processes are.

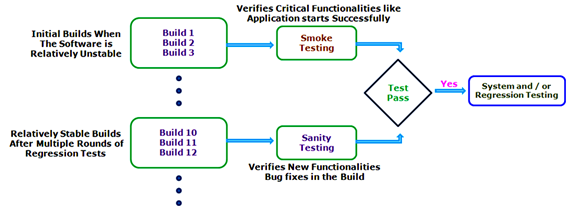
It also checks how software behaves for any hackers attack and malicious programs and how software is maintained for data security after such a hacker attack.

#### **#38) Smoke Testing**

Whenever a new build is provided by the development team then the software testing team validates the build and ensures that no major issue exists.

The testing team ensures that the build is stable and a detailed level of testing is carried out further. [Smoke Testing](https://www.softwaretestinghelp.com/smoke-testing-and-sanity-testing-difference/) checks that no show stopper defect exists in the build which will prevent the testing team to test the application in detail.

If testers find that the major critical functionality is broken down at the initial stage itself then testing team can reject the build and inform accordingly to the development team. Smoke Testing is carried out to a detailed level of any functional or regression testing.



#### **#39) Static Testing**

Static Testing is a type of testing which is executed without any code. The execution is performed on the documentation during the testing phase. It involves reviews, walkthrough, and inspection of the deliverables of the project. Static testing does not execute the code instead of the code syntax, naming conventions are checked.

The [static testing](https://www.softwaretestinghelp.com/static-testing-and-dynamic-testing-difference/) is also applicable for test cases, test plan, design document. It is necessary to perform static testing by the testing team as the defects identified during this type of testing are cost-effective from the project perspective.

#### **#40) Stress Testing**

This testing is done when a system is stressed beyond its specifications in order to check how and when it fails. This is performed under heavy load like putting large number beyond storage capacity, complex database queries, continuous input to the system or database load.

#### **#41) System Testing**

Under [System Testing technique](https://www.softwaretestinghelp.com/system-testing/), the entire system is tested as per the requirements. It is a Black-box type testing that is based on overall requirement specifications and covers all the combined parts of a system.

#### **#42) Unit Testing**

Testing of an individual software component or module is termed as [Unit Testing](https://www.softwaretestinghelp.com/unit-testing/). It is typically done by the programmer and not by testers, as it requires a detailed knowledge of the internal program design and code. It may also require developing test driver modules or test harnesses.

#### **#43) Usability Testing**

Under [Usability Testing](https://www.softwaretestinghelp.com/usability-testing-guide/), User-friendliness check is done. Application flow is tested to know if a new user can understand the application easily or not, Proper help documented if a user gets stuck at any point. Basically, system navigation is checked in this testing.

#### **#44) Vulnerability Testing**

The testing which involves identifying of weakness in the software, hardware and the network is known as Vulnerability Testing. Malicious programs, the hacker can take control of the system, if it is vulnerable to such kind of attacks, viruses, and worms.

So it is necessary to check if those systems undergo Vulnerability Testing before production. It may identify critical defects, flaws in the security.

#### **#45) Volume Testing**

[Volume testing](https://www.softwaretestinghelp.com/what-is-volume-testing/) is a type of non-functional testing performed by the performance testing team.

The software or application undergoes a huge amount of data and Volume Testing checks the system behavior and response time of the application when the system came across such a high volume of data. This high volume of data may impact the system’s performance and speed of the processing time.

#### **#46) White Box Testing**

[White Box testing](https://www.softwaretestinghelp.com/white-box-testing-techniques-with-example/) is based on the knowledge about the internal logic of an application's code.

It is also known as Glass box Testing. Internal software and code working should be known for performing this type of testing. Under these tests are based on the coverage of code statements, branches, paths, conditions etc.

## What is the Test Case?

A Test Case is a set of actions executed to verify a particular feature or functionality of your software application. The Test Case has a set test data, precondition, certain expected and actual results developed for specific test scenario to verify any requirement.

A test case includes specific variables or conditions, using which a test engineer can determine as to whether a software product is functioning as per the requirements of the client or the customer.

## What is a Test Scenario?

A Test Scenario is defined as any functionality that can be tested. It is a collective set of test cases which helps the testing team to determine the positive and negative characteristics of the project.

Test Scenario gives a high-level idea of what we need to test.

## Example of Test Scenario

For an eCommerce Application, a few test scenarios would be

**Test Scenario 1:**Check the Search Functionality

**Test Scenario 2:**Check the Payments Functionality

**Test Scenario 3:**Check the Login Functionality

## Example of Test Cases

Test cases for the **Test Scenario:** "Check the Login Functionality" would be

1. Check system behavior when valid email id and password is entered.
2. Check system behavior when invalid email id and valid password is entered.
3. Check system behavior when valid email id and invalid password is entered.
4. Check system behavior when invalid email id and invalid password is entered.
5. Check system behavior when email id and password are left blank and Sign in entered.
6. Check Forgot your password is working as expected
7. Check system behavior when valid/invalid phone number and password is entered.
8. Check system behavior when "Keep me signed" is checked

## Why do we write Test Cases?

Here, are some important reasons to create a Test Case-

* Test cases help to verify conformance to applicable standards, guidelines and customer requirements
* Helps you to validate expectations and customer requirements
* Increased control, logic, and data flow coverage
* You can simulate 'real' end user scenarios
* Exposes errors or defects
* When test cases are written for test execution, the test engineer's work will be organized better and simplified

## Why do we write Test Scenario?

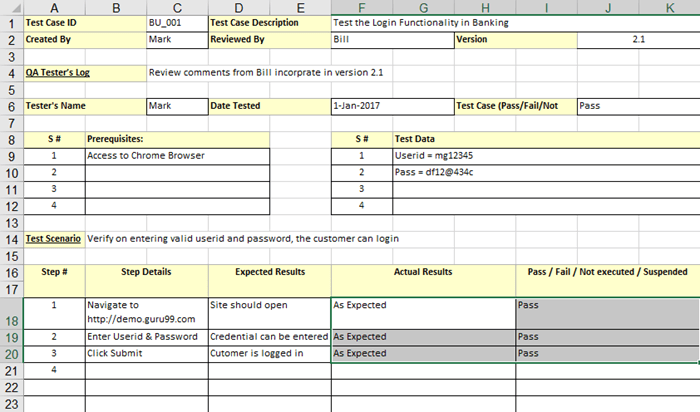
Here, are important reasons to create a Test Scenario:

* The main reason to write a test scenario is to verify the complete functionality of the software application
* It also helps you to ensure that the business processes and flows are as per the functional requirements
* Test Scenarios can be approved by various stakeholders like Business Analyst, Developers, Customers to ensure the Application Under Test is thoroughly tested. It ensures that the software is working for the most common use cases.
* They serve as a quick tool to determine the testing work effort and accordingly create a proposal for the client or organize the workforce.
* They help determine the most critical end-to-end transactions or the real use of the software applications.
* Once these Test Scenarios are finalized, test cases can be easily derived from the Test Cases.

Here, are significant differences between Test scenario and a Test Case

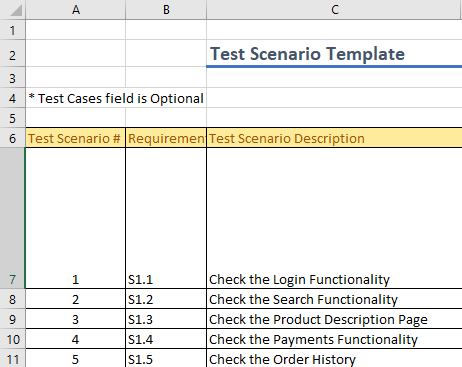
|  |  |
| --- | --- |
| **Test Scenario** | **Test Case** |
| A test scenario contains high-level documentation which describes an end to end functionality to be tested. | Test cases contain definite test steps, data, expected results for testing all the features of an application. |
| It focuses on more "what to test" **than** "how to test". | A complete emphasis on "what to test" **and**"how to test.". |
| Test scenarios are a one-liner. So, there is always the possibility of ambiguity during the testing. | Test cases have defined a step, pre-requisites, expected result, etc. Therefore, there is no ambiguity in this process. |
| Test scenarios are derived from test artifacts like BRS, SRS, etc. | Test case is mostly derived from test scenarios. Multiple Test case can be derived from a single Test Scenario |
| It helps in an agile way of testing the end to end functionality | It helps in exhaustive testing of an application |
| Test scenarios are high-level actions. | Test cases are low-level actions. |
| Comparatively less time and resources are required for creating & testing using scenarios. | More resources are needed for documentation and execution of test cases. |

## Best practices of Creating Test cases

[](https://www.guru99.com/images/1/011819_0751_TestCasevsT3.png)Test Case Example

* Test Cases should be transparent and straightforward
* Create Test Case by keeping the end user in the mind
* Avoid test case repetition
* You need to make sure that you will write test cases to check all software requirements mentioned in the specification document
* Never assume functionality and features of your software application while preparing a test case
* Test Cases must be readily identifiable

## Best practices of creating a Test Scenario

[](https://www.guru99.com/images/1/011819_0751_TestCasevsT4.png)Test Scenario Example

* Test scenarios are mostly single line statement that tells what should be tested
* Scenario description should be simple and easy to understand
* A careful assessment of the stated requirements should be done
* The required tools and resources for testing need to be accumulated before the beginning of the testing process

## Test case design techniques

The main purpose of test case design techniques is to test the functionalities and features of the software with the help of effective test cases. The test case design techniques are broadly classified into three major categories.

### 1.    Specification-Based techniques

### 2.    Structure-Based techniques

### 3.    Experience-Based techniques

## Specification-Based or Black-Box techniques

This technique leverages the external description of the software such as technical specifications, design, and client’s requirements to design test cases. The technique enables testers to develop test cases that provide full test coverage. The Specification-based or [black box test](https://reqtest.com/testing-blog/test-design-techniques-explained-1-black-box-vs-white-box-testing/) case design techniques are divided further into 5 categories. These categories are as follows:

### **Boundary Value Analysis (BVA)**

This technique is applied to explore errors at the boundary of the input domain. It catches any input errors that might interrupt with the proper functioning of the program.

### **Equivalence Partitioning (EP)**

In this technique, the test input data is partitioned into a number of classes having an equivalent number of data. The test cases are then designed for each class or partition.  This helps to reduce the number of test cases.

### **Decision Table Testing**

In this technique, test cases are designed on the basis of the decision tables that are formulated using different combinations of inputs and their corresponding outputs based on various conditions and scenarios adhering to different business rules.

### **State Transition Diagrams**

In this technique, the software under test is perceived as a system having a finite number of states of different types. The transition from one state to another is guided by a set of rules. The rules define the response to different inputs. This technique can be implemented on the systems which have certain workflows within them.

### **Use Case Testing**

A use case is a description of a particular use of the software by a user. In this technique, the test cases are designed to execute different business scenarios and end-user functionalities.  Use case testing helps to identify test cases that cover the entire system.

## Structure-Based or White-Box techniques

The structure-based or white-box technique design test cases based on the internal structure of the software.  This technique exhaustively tests the developed code. Developers who have complete information of the software code, its internal structure, and design help to design the test cases. This technique is further divided into five categories.

### **Statement Testing & Coverage**

This technique involves execution of all the executable statements in the source code at least once.  The percentage of the executable statements is calculated as per the given requirement. This is the least preferred metric for checking test coverage.

### **Decision Testing Coverage**

This technique is also known as branch coverage is a testing method in which each one of the possible branches from each decision point is executed at least once to ensure all reachable code is executed.  This helps to validate all the branches in the code. This helps to ensure that no branch leads to unexpected behavior of the application.

### **Condition Testing**

Condition testing also is known as Predicate coverage testing, each Boolean expression is predicted as TRUE or FALSE.  All the testing outcomes are at least tested once.  This type of testing involves 100% coverage of the code.  The test cases are designed as such that the condition outcomes are easily executed.

### **Multiple Condition Testing**

The purpose of Multiple condition testing is to test the different combination of conditions to get 100% coverage.  To ensure complete coverage, two or more test scripts are required which requires more efforts.

### **All Path Testing**

In this technique, the source code of a program is leveraged to find every executable path. This helps to determine all the faults within a particular code.

## Experience-Based techniques

These techniques are highly dependent on tester’s experience to understand the most important areas of the software.  The outcomes of these techniques are based on the skills, knowledge, and expertise of the people involved. The types of experience-based techniques are as follows:

### **Error Guessing**

In this technique, the testers anticipate errors based on their experience, availability of data and their knowledge of product failure.  Error guessing is dependent on the skills, intuition, and experience of the testers.

### **Exploratory Testing**

This technique is used to test the application without any formal documentation.  There is minimum time available for testing and maximum for test execution.  In this, the test design and test execution are performed concurrently.

Appendix:

Practise website link - <http://www.globalsqa.com/demo-site/>