One of the major drawback of [waterfall STLC model](https://www.softwaretestinghelp.com/what-is-sdlc-waterfall-model/) was that, defects were found at a very later state of the development process, since testing was done at the end of the development cycle. It became very challenging and costly to fix the defects since it were found at a very later stage. To overcome this problem, a new development model was introduced called the “V Model”

V model is now one of the most widely used software development process. Introduction of V model has actually proved the implementation of testing right from the requirement phase. V model is also called as verification and validation model.

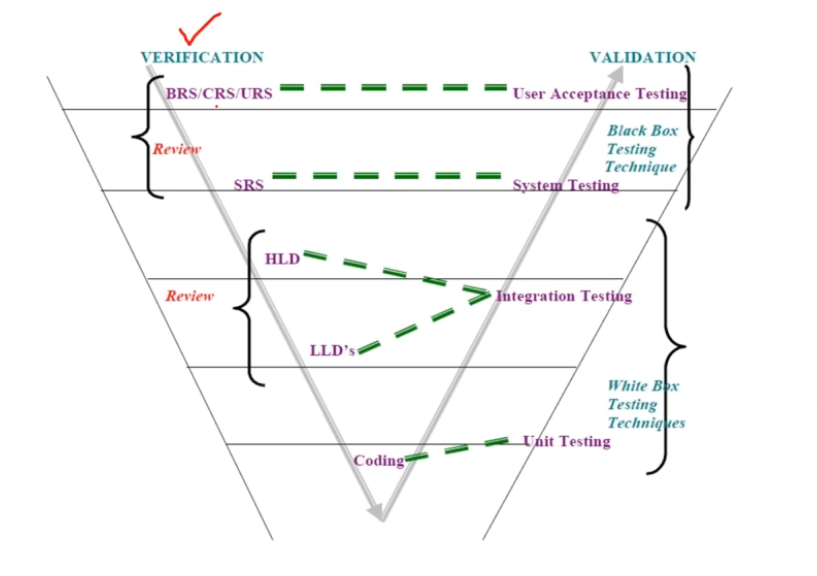
What is V-model- advantages, disadvantages and when to use it?

V- model means Verification and Validation model. Just like the [**waterfall model**](http://tryqa.com/what-is-waterfall-model-advantages-disadvantages-and-when-to-use-it/), the V-Shaped life cycle is a sequential path of execution of processes. Each phase must be completed before the next phase begins. **V-Model** is one of the [**many software development models**](http://tryqa.com/what-are-the-software-development-models/).

Testing of the product is planned in parallel with a corresponding phase of development in **V-model**.

**Diagram of V-model:**





The various phases of the V-model are as follows:

**Requirements** like BRS and SRS begin the life cycle model just like the waterfall model. But, in this model before development is started, a [**system test**](http://tryqa.com/what-is-system-testing/) plan is created.  The [**test plan**](http://tryqa.com/what-is-the-purpose-and-importance-of-test-plans/) focuses on meeting the functionality specified in the requirements gathering.

**The high-level design (HLD)** phase focuses on system architecture and design. It provide overview of solution, platform, system, product and service/process. An [**integration test**](http://tryqa.com/what-is-integration-testing/) plan is created in this phase as well in order to test the pieces of the software systems ability to work together.

**High Level Design** (HLD) is the overall system design - covering the system architecture and database design. It describes the relation between various modules and functions of the system. data flow, flow charts and data structures are covered under HLD.  
[High-level\_design](http://en.wikipedia.org/wiki/High-level_design)

**The low-level design** **(LLD)** phase is where the actual software components are designed. It defines the actual logic for each and every component of the system. Class diagram with all the methods and relation between classes comes under LLD. [**Component tests**](http://tryqa.com/what-is-component-testing/) are created in this phase as well.

**The implementation** phase is, again, where all coding takes place. Once coding is complete, the path of execution continues up the right side of the V where the test plans developed earlier are now put to use.

**Coding:** This is at the bottom of the V-Shape model. Module design is converted into code by developers. [**Unit Testing**](http://tryqa.com/what-is-unit-testing/) is performed by the developers on the code written by them.

**Advantages of V-model:**

* Simple and easy to use.
* Testing activities like planning, [**test designing**](http://tryqa.com/what-is-test-design-or-how-to-specify-test-cases/) happens well before coding. This saves a lot of time. Hence higher chance of success over the waterfall model.
* Proactive defect tracking – that is defects are found at early stage.
* Avoids the downward flow of the defects.
* Works well for small projects where requirements are easily understood.

**Disadvantages of V-model:**

* Very rigid and least flexible.
* Software is developed during the implementation phase, so no early prototypes of the software are produced.
* If any changes happen in midway, then the test documents along with requirement documents has to be updated.

**When to use the V-model:**

* The V-shaped model should be used for small to medium sized projects where requirements are clearly defined and fixed.
* The V-Shaped model should be chosen when ample technical resources are available with needed technical expertise.

## **VERIFICATION: (Static Testing)**

Verification is the process, to ensure that whether we are building the product right i.e., to verify the requirements which we have and to verify whether we are developing the product accordingly or not.

Defining, Monitoring, and optimizing the process is called QA and is prevention oriented-QA

Activities involved here are Inspections, Reviews, Walkthroughs

BRS, SRS,FRS, Design documents are called work products and verification of work products is called verification

Example: Building plan

## What is Testing Review?

A review in a Static Testing is a process or meeting conducted to find the potential defects in the design of any program. Another significance of review is that all the team members get to know about the progress of the project and sometimes the diversity of thoughts may result in excellent suggestions. Documents are directly examined by people and discrepancies are sorted out.

Reviews can further be classified into four parts:

* Informal reviews
* Walkthroughs
* Technical review
* Inspections

Note: Depending on the company feasibility they choose the review types

During the Review process four types of participants that take part in testing are:

* **Moderator**: Performs entry check, follow up on rework, coaching team member, schedule the meeting.
* **Author**: Takes responsibility for fixing the defect found and improves the quality of the document
* **Scribe**: It does the logging of the defect during a review and attends the review meeting
* **Reviewer**: Check material for defects and inspects
* **Manager**: Decide on the execution of reviews and ensures the review process objectives are met.

Types of defects which can be easier to find during static testing are:

* Deviations from standards
* Non-maintainable code
* Design defects
* Missing requirements
* Inconsistent interface specifications

## What is tested in Static Testing

In Static Testing, following things are tested

* Unit Test Cases
* Business Requirements Document (BRD)
* Use Cases
* System/Functional Requirements
* Prototype
* Prototype Specification Document
* DB Fields Dictionary Spreadsheet
* Test Data
* Traceability Matrix Document
* User Manual/Training Guides/Documentation
* Test Plan Strategy Document/Test Cases
* Automation/Performance Test Scripts

The various activities for performing Static Testing are:

1. **Use Cases Requirements Validation:**It validates that all the end-user actions are identified, as well as any input and output associated with them. The more detailed and thorough the use cases are, the more accurate and comprehensive the test cases can be.
2. **Functional Requirements Validation**: It ensures that the Functional Requirements identify all necessary elements. It also looks at the database functionality, interface listings, and hardware, software, and network requirements.
3. **Architecture Review**: All business level process like server locations, network diagrams, protocol definitions, load balancing, database accessibility, test equipment, etc.
4. **Prototype/Screen Mockup Validation**: This stage includes validation of requirements and use cases.
5. **Field Dictionary Validation**: Every field in the UI is defined well enough to create field level validation test cases. Fields are check for min/max length, list values, error messages, etc.

# Inspection:

* It is the most formal review type
* It is led by the trained moderators
* During inspection the documents are prepared and checked thoroughly by the reviewers before the meeting
* It involves peers to examine the product
* A separate preparation is carried out during which the product is examined and the [**defects are found**](http://tryqa.com/what-is-defect-or-bugs-or-faults-in-software-testing/)
* The defects found are documented in a logging list or issue log
* A formal follow-up is carried out by the moderator applying exit criteria

**The goals of inspection are:**

1. It helps the author to improve the quality of the document under inspection
2. It removes defects efficiently and as early as possible
3. It improve product quality
4. It create common understanding by exchanging information
5. It learn from defects found and prevent the occurrence of similar defects

# Walkthrough:

* It is not a formal process/review
* It is led by the authors
* Author guide the participants through the document according to his or her thought process to achieve a common understanding and to gather feedback.
* Useful for the people if they are not from the software discipline, who are not used to or cannot easily understand software development process.
* Is especially useful for higher level documents like requirement specification, etc.

**The goals of a walkthrough:**

1. To present the documents both within and outside the software discipline in order to gather the information regarding the topic under documentation.
2. To explain or do the knowledge transfer and evaluate the contents of the document
3. To achieve a common understanding and to gather feedback.
4. To examine and discuss the validity of the proposed solutions

## **VALIDATION: (Dynamic Testing)**

Validation is the process, whether we are building the right product i.e., to validate the product which we have developed is right or not.

Activities involved in this is Testing the software application

Actual testing, testing a system component/system with respective to requirements and is detection oriented-QC

In simple words, Validation is to validate the actual and expected output of the software

Checking the final product is called Validation

Example: checking the actual Building (final outcome)

What is White Box Testing?

White Box Testing is defined as the testing of a software solution's internal structure, design, and coding. In this type of testing, the code is visible to the tester. It focuses primarily on verifying the flow of inputs and outputs through the application, improving design and usability, strengthening security. White box testing is also known as Clear Box testing, Open Box testing, Structural testing, Transparent Box testing, Code-Based testing, and Glass Box testing. It is usually performed by developers.

## What do you verify in White Box Testing?

White box testing involves the testing of the software code for the following:

* Internal security holes
* Broken or poorly structured paths in the coding processes
* The flow of specific inputs through the code
* Expected output
* The functionality of conditional loops
* Testing of each statement, object, and function on an individual basis

Unit testing: is a level of software testing where individual units/ components of software are tested and confirms those units are as per LLD. The purpose is to validate that each unit of the software performs as designed. A unit is the smallest testable part of any software.

## **Unit Testing Method**

It is performed by using the [White Box Testing](http://softwaretestingfundamentals.com/white-box-testing/) method.

**When is it performed?**

Unit testing is the first [level of software testing](http://softwaretestingfundamentals.com/software-testing-levels/) and is performed prior to [Integration Testing](http://softwaretestingfundamentals.com/integration-testing/).

**Who performs it?**

It is normally performed by software developers themselves or their peers.

To see code is efficient logic, coding standard, no dead code, no unused variables, code coverage etc

### Integration Testing:

Integration Testing is defined as a type of testing where software modules are integrated logically and tested as a group.

A typical software project consists of multiple software modules, coded by different programmers. Integration Testing focuses on checking data communication amongst these modules.

Although each software module is unit tested, defects still exist for various reasons like

* A Module, in general, is designed by an individual software developer whose understanding and programming logic may differ from other programmers. Integration Testing becomes necessary to verify the software modules work in unity
* At the time of module development, there are wide chances of change in requirements by the clients. These new requirements may not be unit tested and hence system integration Testing becomes necessary.
* Interfaces of the software modules with the database could be erroneous
* External Hardware interfaces, if any, could be erroneous
* Inadequate exception handling could cause issues.

### What is Black Box Testing?

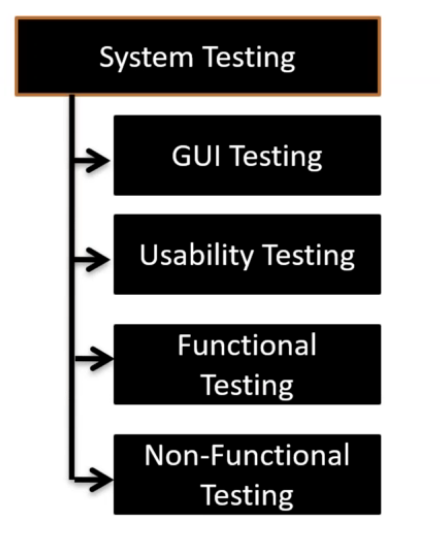
Black box testing is defined as a testing technique in which functionality of the Application under Test (AUT) is tested without looking at the internal code structure, implementation details and knowledge of internal paths of the software. This type of testing is based entirely on software requirements and specifications.

In Black Box Testing we just focus on inputs and output of the software system without bothering about internal knowledge of the software program.

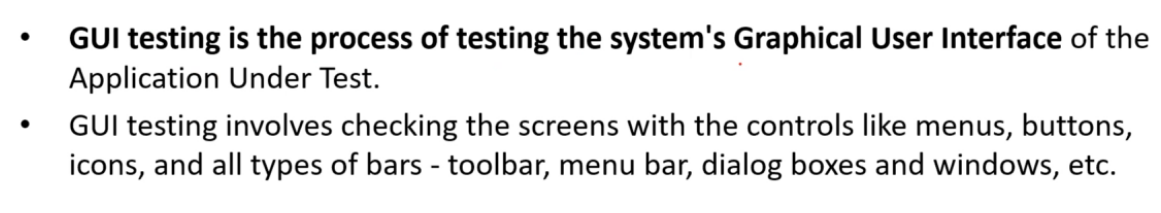
The above Black-Box can be any software system you want to test. For Example, an operating system like Windows, a website like Google, a database like Oracle or even your own custom application. Under Black Box Testing, you can test these applications by just focusing on the inputs and outputs without knowing their internal code implementation.

### System Level Testing

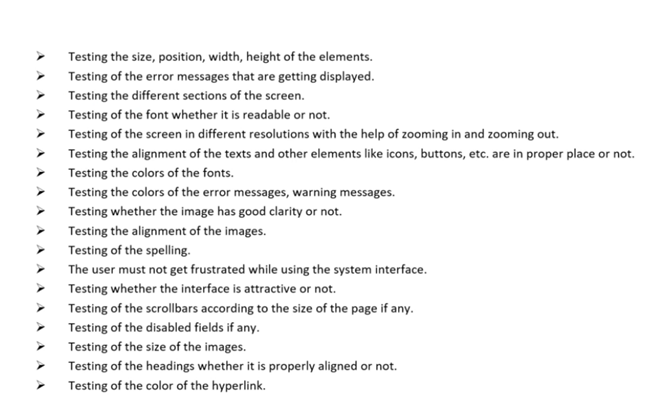
System Testing is the testing of a complete and fully integrated software product. Usually, software is only one element of a larger computer-based system. Ultimately, software is interfaced with other software/hardware systems. System Testing is actually a series of different tests whose sole purpose is to exercise the full computer-based system.System test falls under the **black box testing** category of software testing.



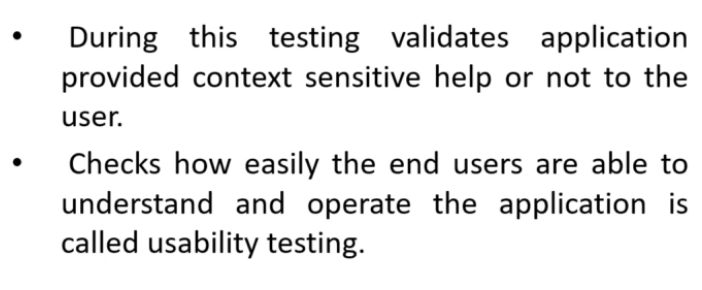
GUI Testing: Focus on look and feel (Spellings, Alignment, Fonts, Size)



GUI Testing Test cases:



**Usability Testing -** [Usability Testing](https://www.guru99.com/usability-testing-tutorial.html) mainly focuses on the user's ease to use the application, flexibility in handling controls and ability of the system to meet its objectives (Context sensitive help-help menu, user guides, tutorial guides, and user manuals)



**Functional Testing -** Also known as functional completeness testing,[Functional Testing](https://www.guru99.com/functional-testing.html) involves trying to think of any possible missing functions. Testers might make a list of additional functionalities that a product could have to improve it during functional testing.



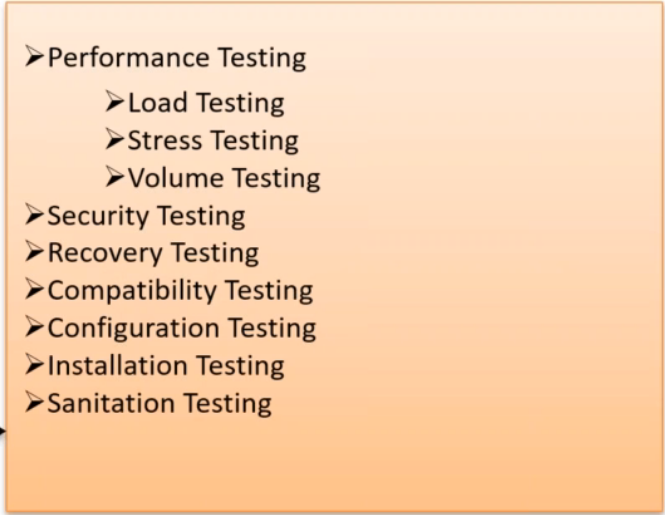
Object properties:

Link get enabled by giving input in textboxes

Links color changing upon clicking

Radio button only one should be clickable or checkboxes

Dropdown list when selected show remaining only

Nonfunctional Testing: 

## What do you verify in System Testing?

System Testing involves testing the software code for following

* Testing the fully integrated applications including external peripherals in order to check how components interact with one another and with the system as a whole. This is also called End to End testing scenario.
* Verify thorough testing of every input in the application to check for desired outputs.
* Testing of the user's experience with the application.

## Different Types of System Testing

There are more than 50 types of System Testing.Below we have listed types of system testing a large software development company would typically use

1. **Usability Testing -** [Usability Testing](https://www.guru99.com/usability-testing-tutorial.html) mainly focuses on the user's ease to use the application, flexibility in handling controls and ability of the system to meet its objectives
2. **Load Testing -** [Load Testing](https://www.guru99.com/load-testing-tutorial.html) is necessary to know that a software solution will perform under real-life loads.
3. **Regression Testing-** - [Regression Testing](https://www.guru99.com/regression-testing.html) involves testing done to make sure none of the changes made over the course of the development process have caused new bugs. It also makes sure no old bugs appear from the addition of new software modules over time.
4. **Recovery Testing -** Recovery testing is done to demonstrate a software solution is reliable, trustworthy and can successfully recoup from possible crashes.
5. **Migration Testing -** Migration testing is done to ensure that the software can be moved from older system infrastructures to current system infrastructures without any issues.
6. **Functional Testing -** Also known as functional completeness testing,[Functional Testing](https://www.guru99.com/functional-testing.html) involves trying to think of any possible missing functions. Testers might make a list of additional functionalities that a product could have to improve it during functional testing.
7. **Hardware/Software Testing -** IBM refers to Hardware/Software testing as "HW/SW Testing". This is when the tester focuses his/her attention on the interactions between the hardware and software during system testing.

### Alpha Testing

This is a form of internal acceptance testing performed mainly by the in-house software QA and testing teams. Alpha testing is the last testing done by the test teams at the development site after the acceptance testing and before releasing the software for beta test.

Alpha testing can also be done by the potential users or customers of the application. But still, this is a form of in-house acceptance testing.

### **What is Beta Testing?**

This is a testing stage followed by the internal full alpha test cycle. This is the final testing phase where the companies release the software to few external user groups outside the company test teams or employees. This initial software version is known as the beta version. Most companies gather user feedback in this release.