Class Test #1

1. *Compare between Hardware & Software.*

Difference Between Hardware and Software

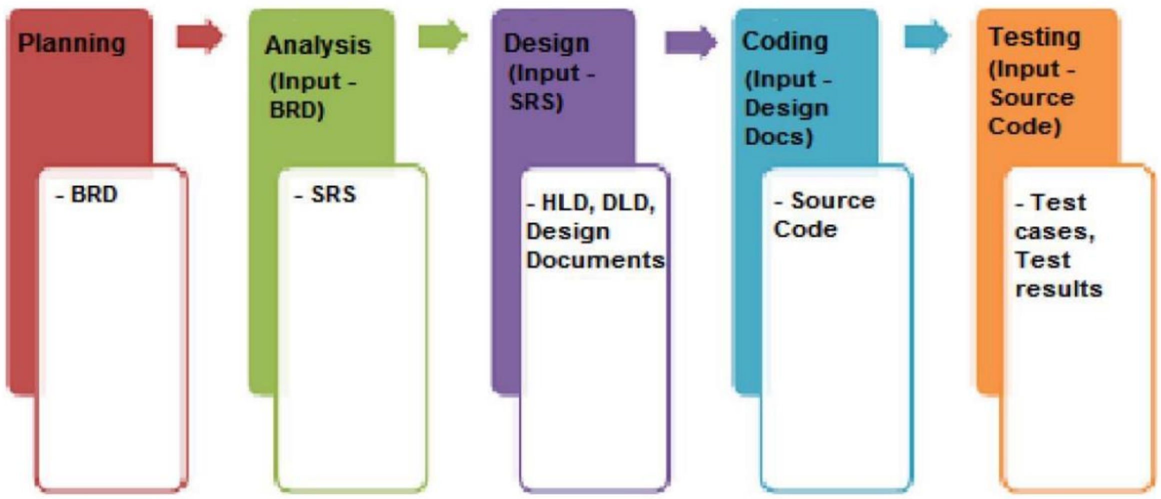
Parameters	**Hardware**	**Software**
Basic Definition	Hardware is a physical part of the computer that causes the processing of data.	Software is a set of instructions that tells a computer exactly what to do.
Development	It is manufactured.	It is developed and engineered.
Dependency	Hardware cannot perform any task without software.	The software can not be executed without hardware.
Process of creating	Electronic and other materials are used to create hardware.	Created by utilizing a computer language to write instructions.
Tangible	Hardware is tangible as hardware is a physical electronic device, that can be touched.	Software is intangible as we can see and also use the software but can't touch them.
Durability	Hardware typically wears out over time.	The software does not wear out with time. However, it may contain flaws and glitches.
Types	It has four main categories: 1. Input Devices 2. Output Devices 3. Storage Devices 4. Internal Components.	It is mainly divided into 1. System software 2. Application software.

Parameters	Hardware	Software
Virus effect	Hardware is not affected by computer viruses.	Software is affected by computer viruses.
Transfer	It cannot be transferred from one place to another electrically through the network.	It can be transferred via a network means.
Machine-Level language	Only machine-level language is known to be understood by hardware.	The program accepts human-readable input, interprets it in machine-level language, and sends it to hardware for additional processing.
Replacement	If the hardware is damaged, it is replaced with a new one.	If the software is damaged, its backup copy can be reinstalled.
Failures	Dust, overheating, humidity, and other factors are commonly responsible for hardware failures.	Overloading, systematic error, major-minor version error, and other factors are commonly responsible for software failures.
Examples	Ex: Keyboard, Mouse, Monitor, Printer, CPU, Hard disk, RAM, ROM, etc.	Ex: MS Word, Excel, PowerPoint, Photoshop, MySQL, etc.

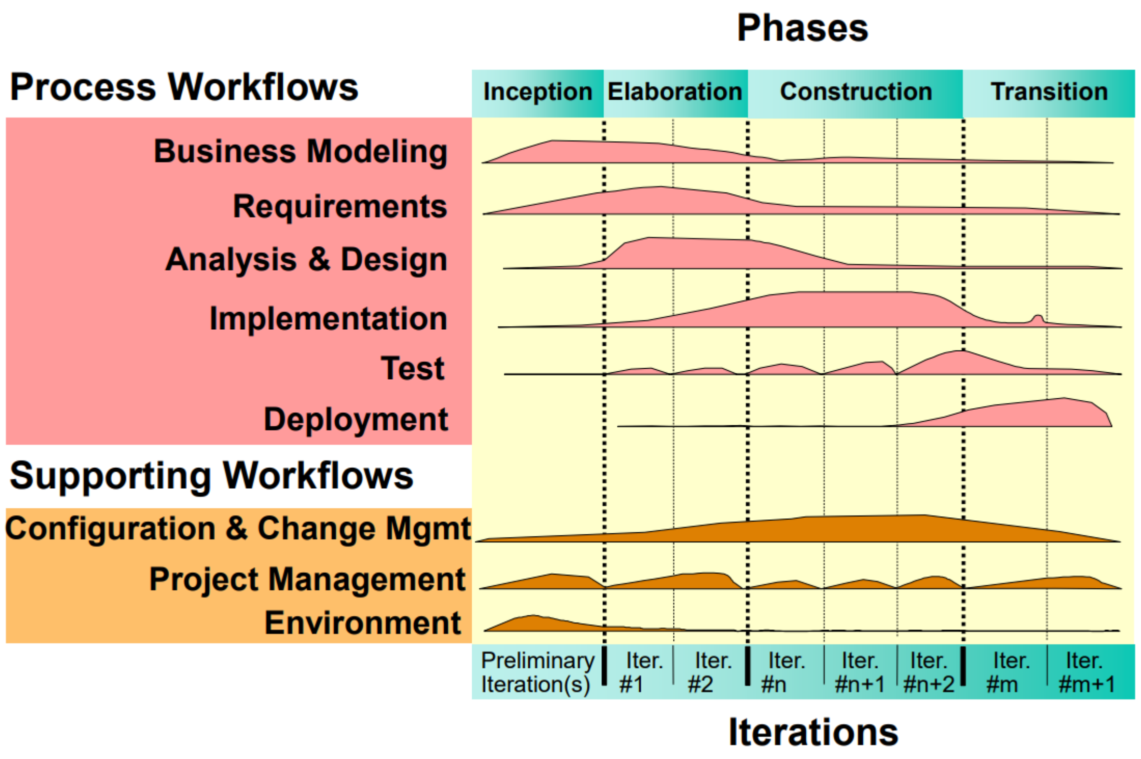
2. *State SDLC phases & their outcome documents.*

https://www.softwaretestinghelp.com/software-development-life-cycle-sdlc/

SDLC Stages & Documents


Planning
- BRD
Analysis (Input - BRD)
- SRS
Design (Input - SRS)
- HLD, DLD, Design Documents
Coding (Input - Design Docs)
- Source Code
Testing (Input - Source Code)
- Test cases, Test results

3. *State the workflow & phases of the RUP.*



4. *Explain the characteristics of a good requirement.*

Characteristics of a Good Requirement

- Clear and Unambiguous
  - standard structure
  - has only one possible interpretation
  - Not more than one requirement in one sentence
- Correct
  - A requirement contributes to a real need
- Understandable
  - A reader can easily understand the meaning of the requirement
- Verifiable
  - A requirement can be tested
- Complete
- Consistent
- Traceable

CUCUVCCT

Class Test #2

1. **Write the characteristics of software testability
   Answer:

   Characteristics of s/w Testability:

   • **Operability** - “The better it works, the more efficiently it can be tested”

   – Relatively few bugs will block the execution of tests.

   – Allowing testing progress without fits and starts.

   • **Observability** - "What you see is what you test.“

   – Distinct output is generated for each input.

   – System states and variables are visible or queriable during execution.

   – Incorrect output is easily identified.

   – Internal errors are automatically detected & reported.

   – Source code is accessible.

• **Controllability** - "The better we can control the software, the more the testing can be automated and optimized.“
– Software and hardware states and variables can be controlled directly by the test engineer.
– Tests can be conveniently specified, automated, and reproduced.

• **Decomposability** - By controlling the scope of testing, we can
more quickly isolate problems and perform smarter retesting.
– Independent modules can be tested independently.

• **Simplicity** - The less there is to test, the more quickly we can test it."
– Functional simplicity (e.g., the feature set is the minimum
necessary to meet requirements).
– Structural simplicity (e.g., architecture is modularized to limit the propagation of faults).
– Code simplicity (e.g., a coding standard is adopted for ease of inspection and maintenance).

• **Stability** - "The fewer the changes, the fewer the disruptions to
testing."
– Changes to the software are infrequent.
– Changes to the software are controlled.
– Changes to the software do not invalidate existing tests.

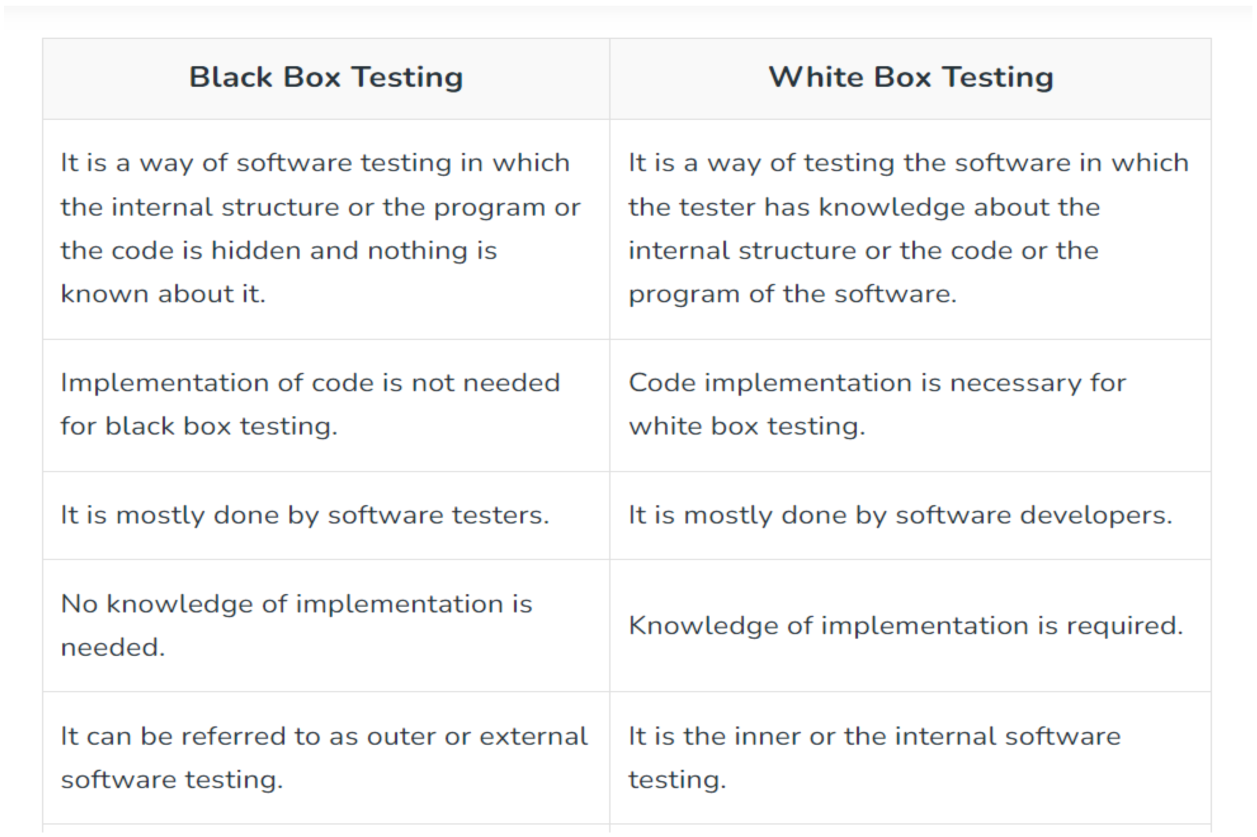
• **Understandability** – "The more information we have, the smarter we will test."
– Dependencies between internal, external, and shared components are well understood.
– Changes to the design are communicated to testers.
– Technical documentation is instantly accessible, well organized,
specific and detailed, and accurate.

2. ***Define cyclomatic complexity.***

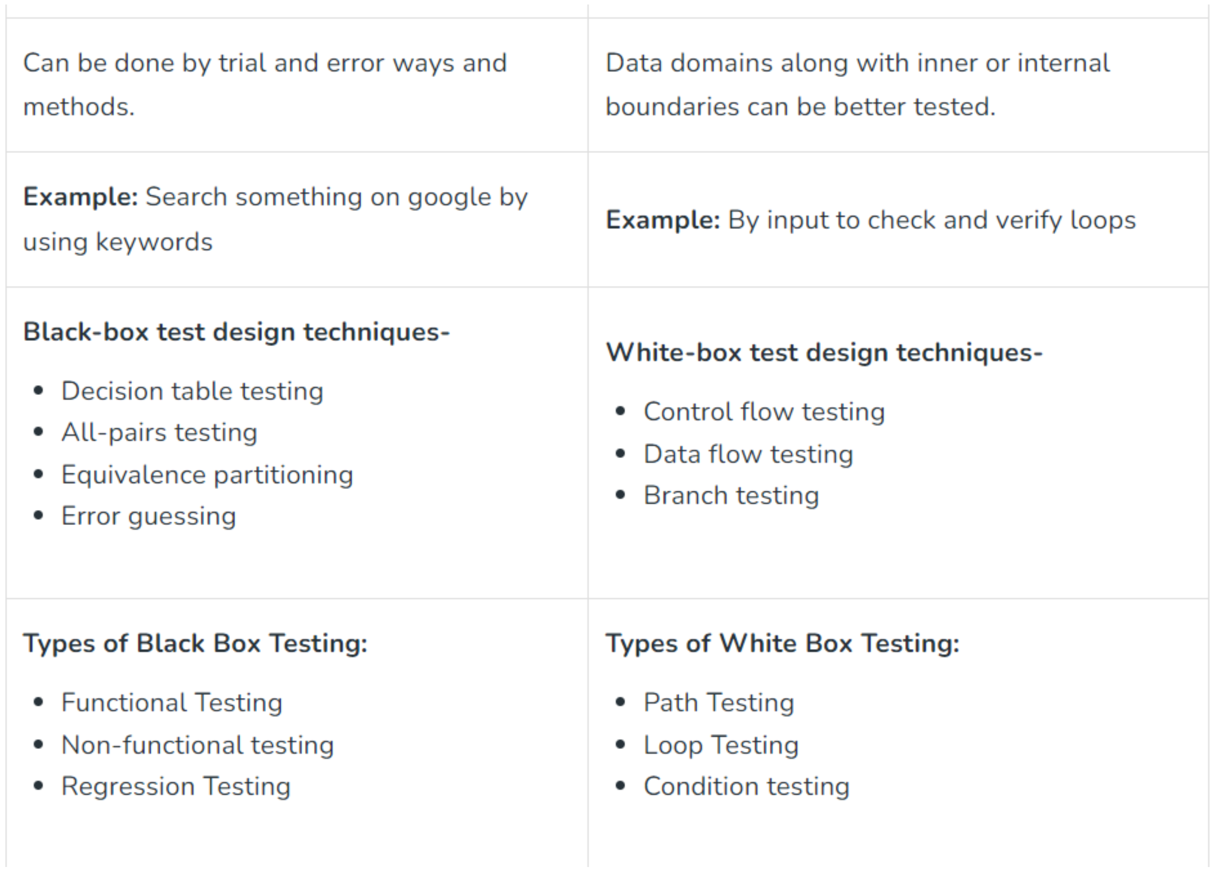
- How do we know how many paths to looks for ?
- *Cyclomatic complexity* is a software metrics that provides a quantitative measure of the logical complexity of a program.
- It defines no. of independent paths in the basis set and also provides number of test that must be conducted.
- One of three ways to compute cyclomatic complexity:
  1. The *no. of regions* corresponds to the cyclomatic complexity.
  2. Cyclomatic complexity, V(G), for a flow graph, G, is defined as
     $V(G) = E - N + 2$
     where E is the number of flow graph edges, N is the number of flow graph nodes.
  3. Cyclomatic complexity, V(G), for a flow graph, G, is also defined as
     $V(G) = P + 1$
     where P is the number of predicate nodes edges

  So the value of V(G) provides us with upper bound of test cases.

3. **Distinguish between white-box & black-box testing.**
   https://www.geeksforgeeks.org/differences-between-black-box-testing-vs-white-box-testing/

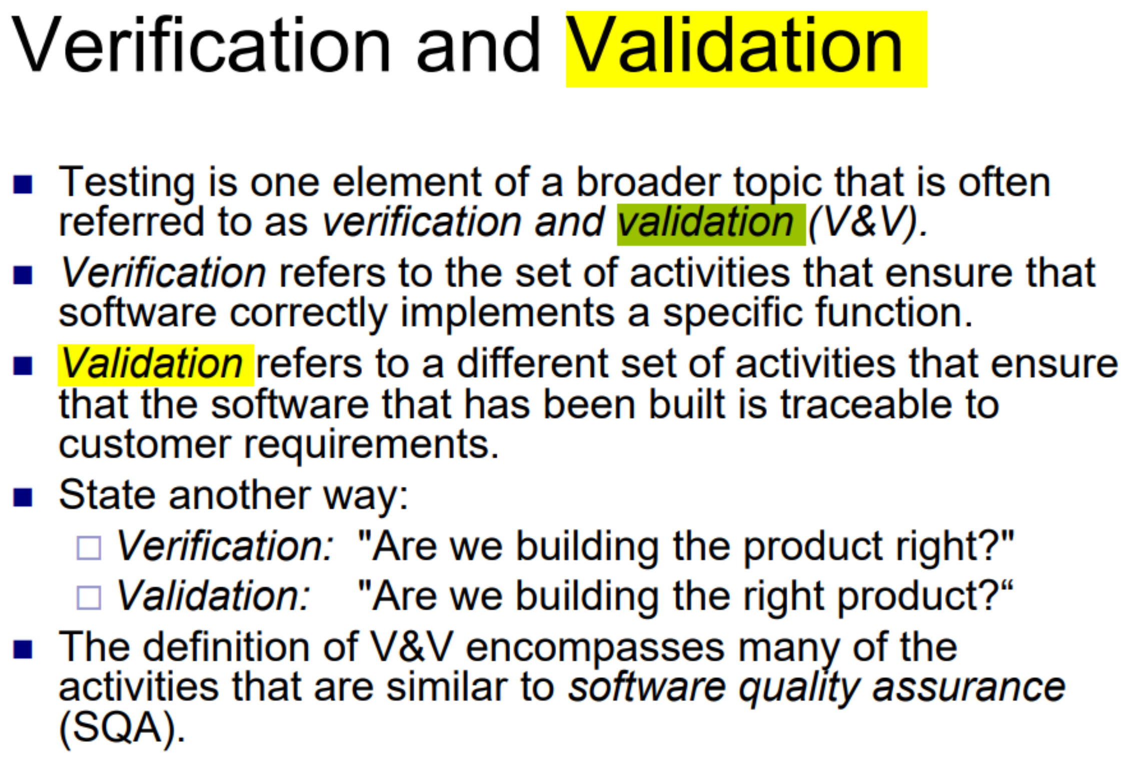


Black Box Testing	White Box Testing
It is a way of software testing in which the internal structure or the program or the code is hidden and nothing is known about it.	It is a way of testing the software in which the tester has knowledge about the internal structure or the code or the program of the software.
Implementation of code is not needed for black box testing.	Code implementation is necessary for white box testing.
It is mostly done by software testers.	It is mostly done by software developers.
No knowledge of implementation is needed.	Knowledge of implementation is required.
It can be referred to as outer or external software testing.	It is the inner or the internal software testing.



Can be done by trial and error ways and methods.	Data domains along with inner or internal boundaries can be better tested.
Example: Search something on google by using keywords	**Example:** By input to check and verify loops
Black-box test design techniques- • Decision table testing • All-pairs testing • Equivalence partitioning • Error guessing	**White-box test design techniques-** • Control flow testing • Data flow testing • Branch testing
Types of Black Box Testing: • Functional Testing • Non-functional testing • Regression Testing	**Types of White Box Testing:** • Path Testing • Loop Testing • Condition testing

4. **What is meant by software validation & verification. Name some SQA activities that are encompassed by V&V process.**



Verification and Validation

- Testing is one element of a broader topic that is often referred to as *verification and validation (V&V).*
- *Verification* refers to the set of activities that ensure that software correctly implements a specific function.
- *Validation* refers to a different set of activities that ensure that the software that has been built is traceable to customer requirements.
- State another way:
  - *Verification:* "Are we building the product right?"
  - *Validation:* "Are we building the right product?"
- The definition of V&V encompasses many of the activities that are similar to *software quality assurance* (SQA).

- V&V encompasses a wide array of SQA activities that include
  - Formal technical reviews,
  - quality and configuration audits,
  - performance monitoring,
  - simulation,
  - feasibility study,
  - documentation review,
  - database review,
  - algorithm analysis,
  - development testing,
  - qualification testing, and installation testing
- Testing does provide the last bastion from which quality can be assessed and, more pragmatically, errors can be uncovered.
- Quality is not measure only by no. of error but it is also measure on application methods, process model, tool, formal technical review, etc will lead to quality, that is confirmed during testing.