



SOFTWARE TESTING

By Dr Rubeena Doomun




LO1 - Understand the stages of testing from testing, during development to acceptance testing by system customers.

LO2 - Introduce to techniques that help you choose test cases that are geared to discovering program defects.

LO3 - Understand test-first development, where you design tests before writing code and run these tests automatically.

LO4 - Know the important differences between component, system, and release testing and be aware of user testing processes and techniques.





Barry Boehm, a pioneer of software engineering, succinctly expressed the difference between validation and verification (Boehm, 1979):

Validation: Are we building the right product?

Verification: Are we building the product right?

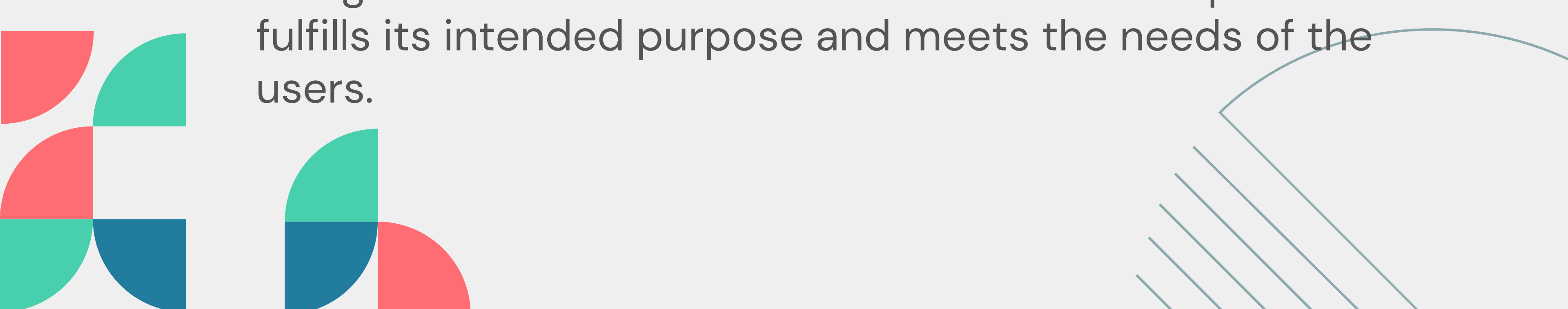


VALIDATION



Validation is the process of evaluating the software during or at the end of the development process to ensure that it meets the business needs and expectations of the customer.

The goal of validation is to ensure that the final product fulfills its intended purpose and meets the needs of the users.

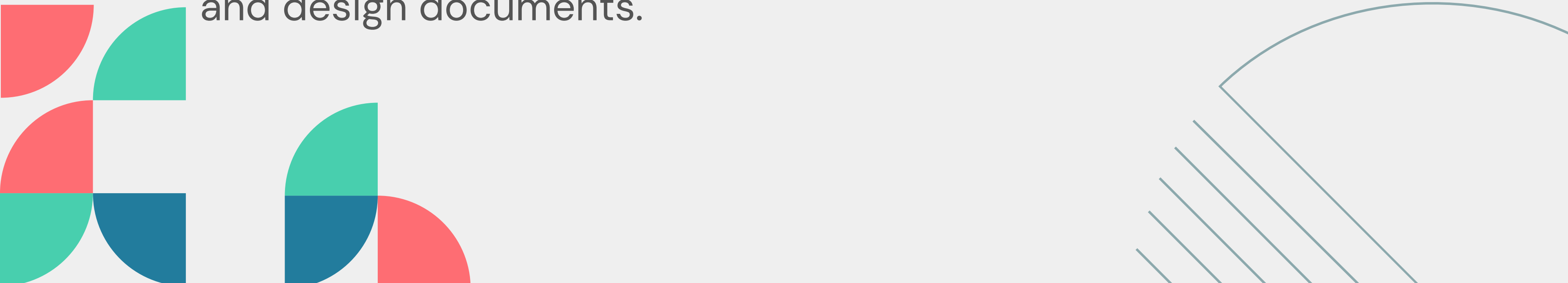


VERIFICATION



Verification is the process of evaluating software to ensure that it complies with the specified requirements and design specifications.

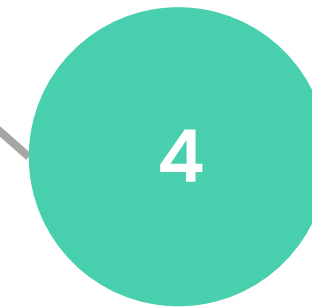
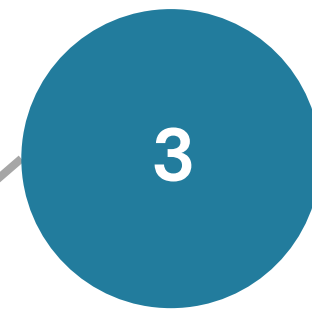
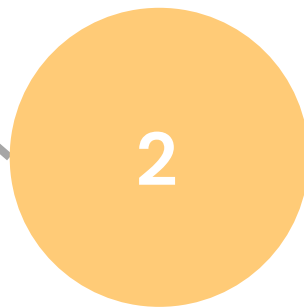
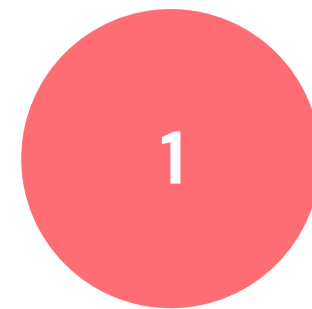
The goal of verification is to ensure that the software is being developed according to the technical specifications, standards, and design documents.



TYPES OF TESTING

Unit Testing

White Box Testing



Black Box Testing

Integration Testing



UNIT TESTING

Unit testing (also known as module testing) is the process of testing individual components (or modules) of a system in isolation.

To test a single module, you'll need a complete environment that includes everything you'll need to run the module.



BLACK BOX TESTING

Black Box Testing is a software testing technique in which the functionality of the application is tested without looking into the internal code, structure, or implementation.

The main focus is on input and output rather than the process that generates the output.

There are two basic techniques to create black box test case scenarios.



EQUIVALENCE CLASS PROPORTIONING

In this approach, the domain of input values to a program is partitioned into a set of equivalence classes. This partitioning is done such that the behaviour of the program is similar for every input data belonging to the same equivalence class. The main idea behind defining the equivalence classes is that testing the code with any one value belonging to an equivalence class is as good as testing the software with any other value belonging to that equivalence class.

BOUNDARY VALUE ANALYSIS

A type of programming error frequently occurs at the boundaries of different equivalence classes of inputs. The reason behind such errors might purely be due to psychological factors. Programmers often fail to see the special processing required by the input values that lie at the boundary of the different equivalence classes.

For example, programmers may improperly use `<` instead of `<=`, or conversely `<=` for.



WHITE BOX TESTING

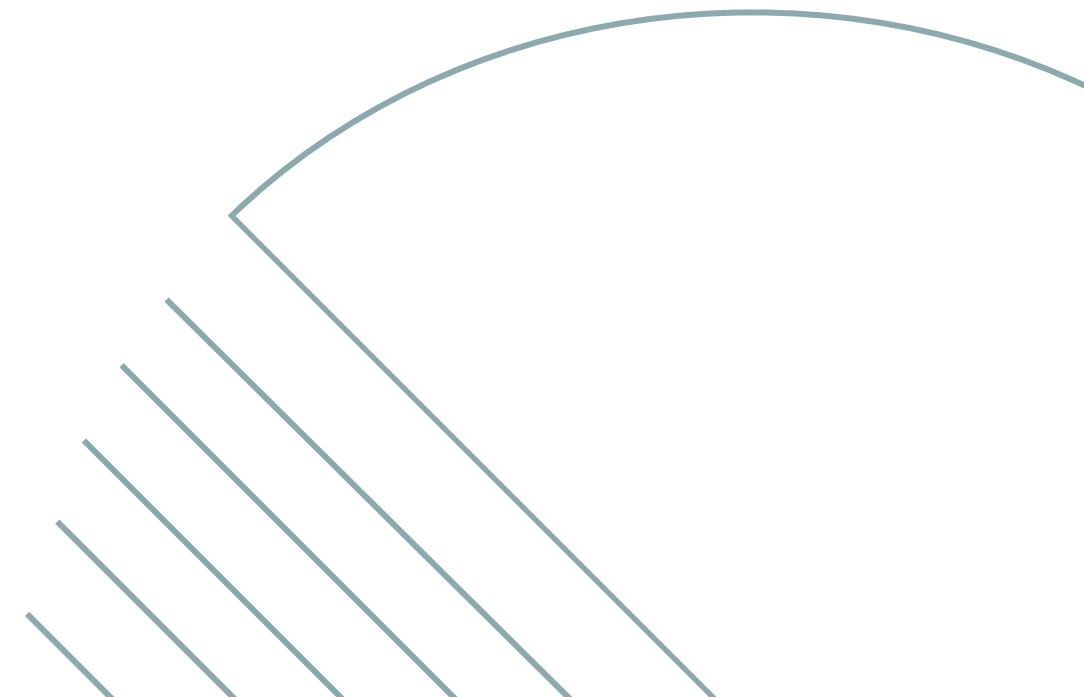
White Box Testing is a method of software testing that examines the internal structure, design, and code of the software.

The tester has knowledge of the internal workings of the application, including its source code, algorithms, and data flow.



TECHNIQUES

- **Statement and Branch Coverage:** Ensures all lines of code and branches (conditional paths) are covered in the tests.
- **Data Flow Testing:** Testing the data flow and ensuring variables are correctly initialized and used.
- **Loop Testing:** Specifically tests loops (e.g., for, while) to check how they behave under various conditions.
- **Control Flow Testing:** Testing the control flow of the application, such as loops, conditional statements, and branches.
- **Path Testing:** Analyzes all paths within the code to find potential logic errors.



INTEGRATION TESTING

Integration Testing is a level of software testing where individual units or components are combined and tested as a group. The primary purpose is to detect faults in the interaction between integrated modules. It ensures that the modules or services that make up a system work together as expected.

01 - BIG BANG APPROACH

02 - BOTTOM UP APPROACH

03 - TOP DOWN APPROACH

04 - MIXED APPROACH

APPROACHES

Big Bang Approach

In the Big Bang approach, all modules are integrated simultaneously after all components have been individually tested. Once everything is integrated, the entire system is tested as a whole.

Bottom Up Approach

The Bottom-Up approach involves testing from the lower-level or fundamental modules upward. Modules are integrated from the bottom (lowest level of hierarchy) to the top until the complete system is tested.

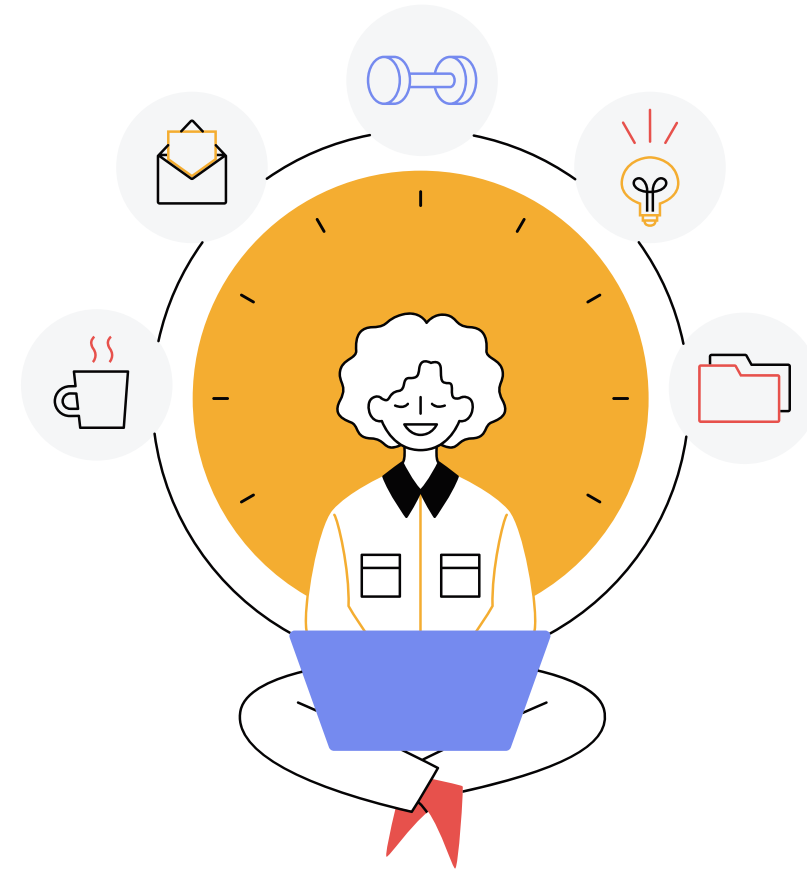
APPROACHES

Top Down Approach

The Top-Down approach starts by testing the higher-level, main modules and progressively integrates and tests the lower-level modules in the hierarchy.

Mixed Approach

The Mixed, or Hybrid, approach combines both Top-Down and Bottom-Up testing. Testing and integration occur both from the top and bottom simultaneously, meeting in the middle.



ACTIVITY TIME

The background features four decorative geometric patterns in the corners. The top-left corner has a series of parallel diagonal lines in a light blue-grey color. The top-right corner contains a cluster of overlapping semi-circles in yellow, red, teal, and dark blue. The bottom-left corner also features a cluster of overlapping semi-circles in red, teal, and dark blue. The bottom-right corner has a series of parallel diagonal lines in a light blue-grey color, mirroring the top-left pattern.

THANK YOU