1. EXPLAIN TOP DOWN AND BOTTOM UP APPROACH ?

Top Down

* A top-down approach is also known as step-wise design approach.

* A top-down approach simply refers to the 'decomposition' means breaking down a component into sub-components.

* In the top down approach, testing takes place from top to the bottom which follows the control flow or architectural structure of the software system.

* Top down approach is a integration testing where top-level modules tested first and then lower-level modules are tested in a step manner until all the modules are not tested.

* In this approach only the top most module is tested separately after that all the lower level modules are combined one by one stand tested. That the system is working as expected.

Bottom - Up

* This approach is also known as "inductive reasoning" and the term refers to the Synthesis.

* Bottom-up Approach is an integration testing in which testing takes place from bottom to up that is lower-level modules are tested first when higher level modules then upper-level modules are tested.

* This testing takes help of drivers for testing the software which are the temporary modules for integration testing.

* Bottom up approach is opposite of top down approach.

* It is a type of testing which is used to analyze the risks in the software system or to evaluate that the system works as expected or not.

* The bottom-up approach is being utilized when off-shelf or existing components are selected.

* ADVANTAGES OF TOP-DOWN APPROACH:

* Fault Localization is easier.

* In the top-down approach, critical modules are evaluated on priority and major design flaws could be detect and fixed first.

* Top down approach requires few or no drivers.

* It is still very successful for functional programming.

* Easy to maintain.

* ADVANTAGES OF BOTTOM-UP APPROACH.

* Fault localization is easy in this approach.

* This testing does not waste time waiting for all modules to be developed.

* It is easy to develop test conditions.

* Interface defects are detected at early stage.

* It provides higher accuracy

2. COMPARE FUNCTIONAL TESTING AND STRUCTURAL TESTING:

STRUCTURAL TESTING

STRUCTURAL Testing is a black-box testing method or Software testing technique in which Software's Structural elements such as modules, components and objects are tested for proper interface and interaction. Structural test is performed without seeing the internal implementation details of the code structure.

* STATEMENT COVERAGE: Statement Coverage is a white box testing technique. It involves executing each line of executable code at least once to ensure that all possible are explored through the code.

* DATA FLOW TESTING: Data flow testing is a method for checking the internal logical consistency of a complex System. It is white box testing technique that involves analyzing the control-flow diagram of the code or its equivalent.

* PATH COVERAGE: path coverage that every branch in the control flow graph be executed at least once. In general, branch coverage is easier to obtain than Statement Coverage because not all Statements.

* BRANCH COVERAGE: Branch coverage or Decision coverage is a white box testing technique.

Functional Testing:

Functional Testing is a type of black-box testing that bases its test cases on the specifications of the software component being tested. It determines if the component under test behaves as specified in the functions specification of user requirements document.

* UNIT TESTING: In unit testing it is verified whether each module works as expected by performing a set of test cases on the individual modules. It focuses mainly on individual components.

* Integration Testing: In integration testing all the modules are tested to verify whether they operate correctly when they are combined.

* SYSTEM TESTING: In System testing it is verified whether the entire integrated application meets the requirements of its users.

* ACCEPTANCE TESTING: In Regression testing it is verified whether changes to the software components have not affected the existing functionalities and features.

* Regression Testing:

In regression testing it is verified whether changes to the software components have not affected the exist.

DIFFERENCE B/W STRUCTURAL AND FUNCTIONAL

STRUCTURAL	FUNCTIONAL TESTING
* Structural testing is done in manual mode	* Functional testing can be performed either manually or automatically.
* Structural test cases are based on input / output conditions.	* Functional test cases are based on actions that a component can perform.
* Structural test case case do not depend on data values	* Functional test cases may have to use some specific value for a test case to pass.
* Structural testing addresses error detection and repair	* Functional Testing may address quality by attributes data.
* Structural testing mainly focuses on logical errors or bugs.	* Functional testing tool is based on the Event Analysis methodology.

3.

Classification of Metrics

Software metrics is a measure of software characterstics which are measurable or countable. Software metrics are valuable for many reasons, including measuring software performance planning work items.

Classification of Software Metrics

* Product Metrics
* Process Metrics.

* PRODUCT METRICS:

These are the measures of various characteristics of the software product. The two important software characteristics are:

* Size and Complexity of Software
* Quality and reliability of Software

* These metrics can be computered for different stages of SDLC.

* PROCESS METRICS:

These are the metrics of various characteristics of the software development process. The efficiency of fault detection. They are used to measure the characteristics of method, techniques and tools are used.

* TYPES OF METRICS:

* INTERNAL METRICS:

Internal metrics are used to measuring properties that are viewed to be a greater importance to a software developer.

* EXTERNAL METRICS:

External metrics are the metrics used for measuring properties they are viewed to be of greater importance to the user.

* HYBRID METRICS:

Hybrid metrics are the metrics used by the combine product, process and resource metrics. For example cost per FP where FP stands for function.

* PROJECT METRICS:

Project metrics used by the project manager to check the project's progress. Data from the past projects are used to collect various metrics, like time and cost. These estimates are used as a base of new software.

* ADVANTAGES OF SOFTWARE METRICS:

* In the preparation of software quality specifications.

* In the verification of compliance of software requirements and specifications.

* In making inference about the effort to be put in the design and development of software system.

* In getting an idea about the complexity of the code

* In guiding resource manager.

* DISADVANTAGES OF SOFTWARE METRICS:

* The application of software metrics is not always easy and in some cases.

* The verification and justification of software metrics are based on historical.

* These are useful for manage Software products but not for evaluating the performance of the tech.

* Most of the predictive models rely on estimates of certain variables which are often not known precisely.

4.

Software Reliability:

Software reliability means operational reliability. It is described as the ability of a system or component to perform its required functions under static conditions for a specific period.

* Software reliability is also defined as the probability that a software system fulfills its assigned task in a given environment for a predefined.

* Software reliability is an essential connect to Software quality, composed with functionality, usability, performance, serviceability, capability, installability, maintainability and documentation.

* Software Reliability is hard to achieve because the complexity of software turn to be high. While any system with a high degree of complexity, containing software will be hard to reach a certain level.

Basic Concepts:

* There are three phases in the life of any hardware component i.e. burn-in, useful life & wear-out.

* In burn in phase failure rate is quite high initially, and it starts decreasing gradually as the time progresses.

* Failure are primarily due to design faults, repairs are made by modifying the design.

* No "wear out" phenomena. Errors can occur without wearing.

* No corrective preventive maintenance for software.

* Reliability is not time dependent. failures occur when the logic path the program takes contains an error.

* Due to the nature of software, no general accepted mechanisms exist to predict software reliability.

* Importank empirical observation and experience. and Good methods can largely improve Software reliability.

* Software testing serves as a way to measure and improve Software reliability

* unfeability of completely testing a Software module: defect-free Software products cannot be assured.

* Database with Software failure rates are available but numbers should be used with caution and adjusted based on observation and experience.

* Reliability is usually more important than efficiency and no need to utilise hardware to full extent as computers are cheap and fast