NAME: MGBEZE EMMANUEL

MATRIC NUMBER: 15/ENG05/013

**CODING:**

This is converting the design of a system to codes in a high level language. When coding, one must follow the standard style of coding.

Coding is done in programming languages.

Characteristics of programming languages:

* Cost
* Efficiency
* Error checking
* Brevity
* Readability
* Familiar notation
* Widely available, e.t.c

**Code Review**

Reviewing codes for a model is done after the module is successfully compiled and all the syntax errors have been eliminated. They reduce coding errors and produce high quality codes.

**Code Walk Throughs**

This is am informal code analysis technique. The main objectives of the walk through are to discover the algorithmic and logical errors in the code

Some members of the development team are given the code few days before the walk through meeting to read and understand code. Each member selects some test cases and trace execution through each statement and function execution.

The code walk through should consist of 3-7 persons and the focus should be on discovery of errors and not on how to fix the errors.

**Code Inspection**

This is similar to code walk through. In code inspection, emphasis are laid on discovering some common types of errors caused due to oversight and improper programming. During code inspection the code is examined for the presence of certain kinds of errors, in contrast to the hand simulation of code execution done in code walk throughs.

A list of commonly committed errors can be used during code inspection to look out for possible errors.

**Some examples are;**

Use of uninitialized variables, Jumps into loops, Nonterminating loops, Incompatible assignments, Array indices out of bounds, Improper storage allocation and deallocation, Mismatches between actual and formal parameter in procedure calls, Use of incorrect logical operators or incorrect precedence among operators, Improper modification of loop variables, Comparison of equally of floating point variables, etc.

**Clean Room Testing**

This type of testing relies heavily on walk throughs, inspection, and formal verification by using a rigorous inspection process. The programmers are not allowed to test any of their code by executing the code other than doing some syntax testing using a compiler.

The clean room approach to software development is based on five characteristics:

* Formal specification
* Incremental development
* Structured programming
* Static verification
* Statistical testing of the system

**Software Documentation**

This involves user’s manual, software requirements specification (SRS) documents, design documents, test documents, installation manual which are all part of the software engineering process.

Different types of software documents can broadly be classified into the following:

Internal documentation:

Internal documentation is provided through the useful variable names, module and function headers, code indentation, code structuring, use of enumerated types and constant identifiers, use of user-defined data types. Internal documentation is very useful in understanding a code.

External documentation:

External documentation is provided through various types of supporting documents such as user’s manual, software requirements specification document, design document, test documents, etc.

**Program Testing**

Testing a program consists of providing the program with a set of test inputs (or test cases) and observing if the program behaves as expected.

Terms associated with program testing are; failure, test case(ISO), test suite

Aim of Testing

The aim of the testing process is to identify all defects existing in a software product. Testing provides a practical way of reducing defects in a system and increasing the user’s confidence in a developed system.

Verification

Verification is the process of determining whether the output of one phase of software development conforms to that of its previous phase

Validation

is the process of determining whether a fully developed system conforms to its requirements specification.

Testing in small

Software products are normally tested first at the individual component (or unit) level.

Testing in large

After testing all the components individually, the components are slowly integrated and tested at each level of integration (integration testing). Finally, the fully integrated system is tested (called system testing).

Unit testing

Unit testing (or module testing) is the testing of different units (or modules) of a system in isolation.

Black Box testing

In the black-box testing, test cases are designed from an examination of the input/output values only and no knowledge of design or code is required

Statement Coverage

The statement coverage strategy aims to design test cases so that every statement in a program is executed at least once.

Branch Coverage

In the branch coverage-based testing strategy, test cases are designed to make each branch condition to assume true and false values in turn

Condition Coverage

In this structural testing, test cases are designed to make each component of a composite conditional expression to assume both true and false values

Path Coverage

The path coverage-based testing strategy requires us to design test cases such that all linearly independent paths in the program are executed at least once.

Control Flow Graph (CFG)

A control flow graph describes how the control flows through the program. The CFG for any program can be easily drawn by knowing how to represent the sequence, selection, and iteration type of statements in the CFG.

Debugging

Identifying errors in a program code and then fix them up are known as debugging.

Debugging Guidelines

* Many times debugging requires a thorough understanding of the program design. Trying to debug based on a partial understanding of the system design and implementation may require an inordinate amount of effort to be put into debugging even simple problems.
* Debugging may sometimes even require full redesign of the system. In such cases, a common mistake that novice programmers often make is attempting not to fix the error but its symptoms.
* One must be beware of the possibility that an error correction may introduce new errors. Therefore, after every round of error-fixing, regression testing must be carried out.

Program Analysis

Tools A program analysis tool means an automated tool that takes the source code or the executable code of a program as input and produces reports regarding several important characteristics of the program, such as its size, complexity, adequacy of commenting, adherence to programming standards, etc.

Static Analysis Tool

It assesses and computes various characteristics of a software product without executing it.

Dynamic program analysis tools

They require the program to be executed and its actual behaviour recorded.

Integration Testing

The integration plan specifies the steps and the order in which modules are combined to realize the full system.

There are four types of integration testing approaches. Any one (or a mixture) of the following approaches can be used to develop the integration test plan.

Those approaches are the following:

* Big bangapproach
* Bottom-up approach
* Top-down approach
* Mixed-approach

Phased Testing

In phased integration, a group of related modules are added to the partial system each time.

Phased integration requires lesser number of integration steps.

Incremental testing

In incremental integration testing, only one new module is added to the partial system each time.

when failures are detected, it is easier to debug the system in the incremental testing approach since it is known that the error is caused by addition of a single module.

Performance Testing

 Performance testing is carried out to check whether the system needs the non-functional requirements identified in the SRS document. There are several types of performance testing listed below

* Stress testing: Stress testing evaluates system performance when it is stressed for short periods of time
* Volume testing: -It is especially important to check whether the data structures have been designed to successfully extraordinary situations
* Configuration testing: This is used to analyze system behavior in various hardware and software configurations specified in the requirements
* Compatibility testing: -This type of testing is required when the system interfaces with other types of systems.
* Regression testing: This type of testing is required when the system being tested is an upgradation of an already existing system to fix some bugs
* Recovery testing: Recovery testing tests the response of the system to the presence of faults, or loss of power, devices, services, data, etc
* Maintenance testing: This testing addresses the diagnostic programs, and other procedures that are required to be developed to help maintenance of the system
* Documentation testing: It is checked that the required user manual, maintenance manuals, and technical manuals exist and are consistent
* Usability testing: Usability testing concerns checking the user interface to see if it meets all user requirements concerning the user interface

Regression Testing

 Regression testing does not belong to either unit test, integration test, or system testing. Instead, it is a separate dimension to these three forms of testing

**Software Maintenance**

Types of software maintenance

There are basically three types of software maintenance. These are:

* Corrective: Corrective maintenance of a software product is necessary to rectify the bugs observed while the system is in use.
* Adaptive: A software product might need maintenance when the customers need the product to run on new platforms, on new operating systems, or when they need the product to interface with new hardware or software.
* Perfective: A software product needs maintenance to support the new features that users want it to support, to change different functionalities of the system according to customer demands, or to enhance the performance of the system

Problems associated with software maintenance

Software maintenance work typically is much more expensive than what it should be and takes more time than required. Software maintenance has a very poor image in industry. Therefore, an organization often cannot employ bright engineers to carry out maintenance work. Another problem associated with maintenance work is that the majority of software products needing maintenance are legacy products.

Software Reverse Engineering

 Software reverse engineering is the process of recovering the design and the requirements specification of a product from an analysis of its code.

Reverse engineering is becoming important, since legacy software products lack proper documentation, and are highly unstructured. Even well designed products become legacy software as their structure degrades through a series of maintenance efforts.