Software Coding and Testing

4.1 Coding standards and guidelines

- Before understanding coding standard, let's first discuss about coding.

❖ Concept of Coding

- Coding is a computer programming language that helps to communicate with a computer.
- Computers do not understand human languages.
- Coding allows humans to communicate with the computer.
- Code instructs the computer which tasks to be performed and what things to do.

❖ Coding Standard

- Different modules specified in the design document are coded in the Coding phase according to the module specification.
- The main goal of the coding phase is to code from the design document prepared after the design phase through a high-level language and then to unit tests this code.
- Good software development organizations want their programmers to maintain to some well-defined and standard style of coding called coding standards.
- They usually make their own coding standards and guidelines depending on what suits their organization best and based on the types of software they develop.
- It is very important for the programmers to maintain the coding standards otherwise the code will be rejected during code review.

❖ Purpose of Having Coding Standards:

- A coding standard gives a uniform appearance to the codes written by different engineers.
- It improves readability, and maintainability of the code and it reduces complexity also.
- It helps in code reuse and helps to detect error easily.
- It promotes sound programming practices and increases efficiency of the programmers.

❖ Coding Guidelines

- Safe: It can be used without causing harm.
- Secure: It can't be hacked.
- Reliable: It functions as it should, every time.
- Testable: It can be tested at the code level.
- Maintainable: It can be maintained, even as your codebase grows.
- Portable: It works the same in every environment.

❖ Characteristics of good coding

- 7 Simple Attributes of Good Code. It comes down to one programming principle
- The Code Must Be Readable

- The Code Must Be Scalable
- The Code Must Be Testable.
- The Code Does What Is Asked For
- The Code Fails Gracefully
- The Code Is Easy to Extend
- The Code Is Reusable

❖ **Advantage of coding standard and guidelines**

- Coding standards help you develop less complex software and, therefore, reduce errors.
- Improvement of the bug fixing process: with coding standards, it becomes easier for developers to locate and correct bugs in the source code because it is written in a consistent manner.

4.2 Code review

- Code review for a model is carried out after all the syntax errors have been eliminated.
- This is the cost-effective strategies for reduction in coding errors and to produce high quality code.
- Two types code review techniques are code inspection and code walk through.

❖ **CODE WALK THROUGH**

- Code walk through is a code analysis technique.
- In this technique carried out after a module has been coded, successfully compiled and all syntax errors eliminated.
- A few 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 simulates execution of the code by hand (i.e. trace execution through each statement and function execution).
- The main objectives of the walk through are to find the algorithmic and logical errors in the code.
- The members note down their findings to discuss these in a walk through meeting where the coder of the module is present.
- Some of the guidelines for this technique:
  - ✓ It should consist of between three to seven members.
  - ✓ Discussion should focus on find the errors and not on how to fix the errors.
  - ✓ Managers should not attend the walk through meetings.

❖ **CODE INSPECTION**

- The aim of code inspection is to discover common types of errors caused due to improper programming.
- During code inspection the code is examined for the presence of certain kinds of errors.
- Error will be discovered by looking for these kinds of mistakes in the code.
- Coding standards is also checked during code inspection.

- Good software development companies collect different types of errors commonly committed by their engineers and identify the type of errors most frequently committed.
- List of commonly committed errors can be used during code inspection to look out for possible errors.
- Following is a list of some errors which can be checked during code inspection:
  - ✓ Use of uninitialized variables
  - ✓ No terminating 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

4.3 Software Documentation

- When various kinds of software products are developed then not only the executable files and the source code are developed but also various kinds of documents such as users' manual, software requirements specification (SRS) documents, design documents, test documents, installation manual, etc are also developed as part of any software engineering process.
- Good documents enhance understandability and maintainability of a software product.
- Documents help the users in effectively using of the system.
- Good documents help in effectively handling the manpower.
- Even when an engineer leaves the organization, and a new engineer comes in, he can build up the required knowledge easily.
- Different types of software documents can be classified into the following:
  - ✓ Internal documentation
  - ✓ External documentation

❖ INTERNAL DOCUMENTATION

- Documentation which focuses on the information that is used to determine the software code is known as internal documentation.
- It describes the data structures, algorithms, and control flow in the programs.
- It includes header comment blocks and program comments.
- Header comment blocks are useful in identifying the purpose of the code along with details such as how the functions are used in the program.
- Since software code is updated and revised several times, it is important to keep a record of the code information so that internal documentation reflects the changes made to the software code.
- Internal documentation should explain how each code section relates to user requirements in the software. Generally, internal documentation includes the following information.
  - ✓ Name, type, and purpose of each variable and data structure used in the code
  - ✓ Brief description of algorithms, logic, and error-handling techniques

- ✓ Information about the required input and expected output of the program
- ✓ Information on the up gradations in the program.

❖ **EXTERNAL DOCUMENTATION**

- Its detail is known as external documentation.
- It includes information such as function of code, name of the software developer who has written the code, dependency of code on programs, and format of the output produced by the software.
- Generally, external documentation describing the design of the program.
- It consists of information such as description of the problem along with the program written to solve it.
- It describes the approach used to solve the problem, operational requirements of the program, and user interface components.
- For the purpose of readability and proper understanding, the detailed description is given by figures and illustrations that how one component is related to another.
- External documentation explains why a particular solution is chosen and implemented in the software.
- It includes formulas, conditions, and references from where the documentation is derived.
- External documentation makes the user aware of the errors that occur while running the software code. For example, if an array of five numbers is used, it should be mentioned in the external documentation that the limit of the array is five.

4.4 Testing Fundamentals

- 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.
- If the program fails to behave as expected, then the conditions under which failure occurs are noted for later debugging and correction.
- Some commonly used terms associated with testing are:
  - ✓ **Error:** It is a mistake committed by the development team during any of the development phases. An error is also called fault, a bug, or a defect.
  - ✓ **Failure**: This is a form of an error (or defect or bug). Due to Error program can failure.
  - ✓ **Test case**: This is the triplet [I,S,O], where I is the data input to the system, S is the state of the system at which the data is input, and O is the expected output of the system.
  - ✓ **Test suite**: This is the set of all test cases with which a given software product is to be tested.

❖ **Verification**

- Verification is the process of checking that software achieves its goal without any bugs.
- It is the process to ensure whether the product that is developed is right or not.
- It verifies whether the developed product fulfills the requirements that we have.
  Verification is **Static Testing**.
- Activities involved in verification:
  1. Inspections

2. Reviews
3. Walkthroughs
4. Desk-checking

❖ **Validation**

- Validation is the process of checking whether the software product is up to the mark or in other words product has high level requirements.
- It is the process of checking the validation of product i.e. it checks what we are developing is the right product.
- Its validation of actual and expected product.
  Validation is the **Dynamic Testing**.
- Activities involved in verification:
  1. Black box testing
  2. White box testing
  3. Unit testing
  4. Integration testing

Verification	Validation
Are we implementing the system right?	Are we implementing the right system?
Evaluating products of a development phase	Evaluating products at the closing of the development process
The objective is making sure the product is as per the requirements and design specifications	The objective is making sure that the product meets user's requirements
Activities included: reviews, meetings, and inspections	Activities included: black box testing, white box testing, and grey box testing
Verifies that outputs are according to inputs or not	Validates that the users accept the software or not
Items evaluated: plans, requirement specifications, design specifications, code, and test cases	Items evaluated: actual product or software under test
Manual checking of the documents and files	Checking the developed products using the documents and files

4.5 Functional Testing – 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. The following are the two main approaches to designing black box test cases.
  - ✓ Equivalence class portioning
  - ✓ Boundary value analysis

Black box - we do not know anything

❖ Equivalence Class Partitioning

- 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 behavior of the program is similar for every input data belonging to the same equivalence class.
- In this approach 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.
- Equivalence classes can be designed by examining the input data and output data. The following are some general guidelines for designing the equivalence classes:
  - ✓ If the input data values to a system can be specified by a range of values, then one valid and two invalid equivalence classes should be defined. For example if the equivalence class is the set of integers in the range 1 to 10 (e.g. [1, 10]) then the invalid equivalence classes are [-∞, 0], [11, +∞]
  - ✓ If the input data assumes values from a set of discrete members of some domain, then one equivalence class for valid input values and another equivalence class for invalid input values should be defined. For example if the valid equivalence classes are {A,B,C} then the invalid equivalence class is U-{A,B,C} where U is the universe of possible input values.
- In short it is the process of taking all possible test cases and placing them into classes. One test value is picked from each class while testing.
- For example, Test cases for input box accepting numbers between 1 and 1000 using Equivalence Partitioning:
  1. One input data class with all valid inputs. Pick a single value from range 1 to 1000 as a valid test case. If you select other values between 1 and 1000 then result is going to be same. So one test case for valid input data should be sufficient.
  2. Input data with any value less than 1 (Lower Limit) to represent invalid input class.
  3. Input data with any value greater than 1000 (Upper Limit) to represent invalid input class.
- So using equivalence partitioning you has categorized all possible test cases into three classes. Test cases with other values from any class should give you the same result.

❖ Boundary value analysis

- It's widely recognized that input values at the extreme ends of input domain cause more errors in system. More application errors occur at the boundaries of input domain.
- Boundary value analysis testing technique is used to identify errors at boundaries rather than finding those exist in center of input domain.
- Boundary value analysis is a next part of Equivalence partitioning for designing test cases where test cases are selected at the edges of the equivalence classes.
- Test cases for input box accepting numbers between 1 and 1000 using Boundary value analysis:
  1. Test cases with test data exactly as the input boundaries of input domain i.e. values 1 and 1000.
  2. Test data with values just below the extreme edges of input domains i.e. values 0 and 999.
  3. Test data with values just above the extreme edges of input domain i.e. values 2 and 1001.

Advantages and disadvantages of "Black Box" testing

Advantages	Disadvantages
Penetration testing speed. It can be finished faster than other types of testing due to the restricted scope.	A full comprehensive system testing cannot be performed because the penetration testing is carried out externally.
Ability to get the cheaper price. The cost of the work is based on the difficulty of the system and the scope.	The time of the testing is limited. For this reason, the testing shows only the level of security at the moment of performing it.

4.6 Structural Testing – White box testing

- In this method of testing the test cases are calculated based on analysis internal structure of the system based on Code coverage, branches coverage, paths coverage, condition Coverage etc.
- White box testing involves the testing by looking at the internal structure of the code & when you completely aware of the internal structure of the code then you can run your test cases & check whether the system meet requirements mentioned in the specification document.
- Based on derived test cases the user exercised the test cases by giving the input to the system and checking for expected outputs with actual output.

White box – we know everything

❖ Statement coverage

- The statement coverage strategy aims to design test cases so that every statement in a program is executed at least once.
- The principal idea governing the statement coverage strategy is that unless a statement is executed, it is very hard to determine if an error exists in that statement.
- Unless a statement is executed, it is very difficult to observe whether it causes failure due to some illegal memory access, wrong result computation, etc.
- For ex.

```
If(a>b)
printf("a is greater")
else
printf("b is greater than")
```

- Test cases for above example: {a=5,b=10}, {a=10,b=5}

❖ 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.
- Branch testing is also known as edge testing as in this testing scheme, each edge of a program's control flow graph is traversed at least once.
- It is obvious that branch testing guarantees statement coverage and thus is a stronger testing strategy compared to the statement coverage-based testing.
- For example, find maximum from three number:
-

```
If(a>b && a>c)
{
max=a;
}
else if
```

```
(b>c)
{
max=b;
}
else
{
max=c;
}
```

Test cases for above example: {a=5,b=10,c=15}, {a=5,b=15,c=10}, {a=15, b=5, c=10}

❖ 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.
- For example, in the conditional expression ((c1.and.c2).or.c3), the components c1, c2 and c3 are each made to assume both true and false values.
- Branch testing is probably the simplest condition testing strategy where only the compound conditions appearing in the different branch statements are made to assume the true and false values.
- Thus, condition testing is a stronger testing strategy than branch testing and branch testing is stronger testing strategy than the statement coverage-based testing.
- For a composite conditional expression of n components, for condition coverage, 2^n test cases are required. Thus, for condition coverage, the number of test cases increases exponentially with the number of component conditions.
- Therefore, a condition coverage-based testing technique is practical only if n (the number of conditions) is small.

❖ 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.
- A linearly independent path can be defined in terms of the control flow graph (CFG) of a program.
- Path testing is used for module or unit testing.
- It requires complete knowledge of the program structure.

Advantages and disadvantages of "White Box" testing

Advantages	Disadvantages
Allows performing a comprehensive system or application analysis.	Duration and difficulty of the testing.
Finding conceptual errors at the architecture level.	Functional which is accessible on the Internet. "Black Box" testing is required anyway.
Performing analysis during the development.	The high cost of the work.

Differences between Black Box Testing vs White Box Testing:

S. No.	Black Box Testing	White Box Testing
1.	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.
2.	Implementation of code is not needed for black box testing.	Code implementation is necessary for white box testing.
3.	It is mostly done by software testers.	It is mostly done by software developers.
4.	No knowledge of implementation is needed.	Knowledge of implementation is required.

4.7 Overview of Alpha & Beta 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 potential users or customers of the application. Still, this is a form of in-house acceptance testing.

❖ Beta Testing

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

Alpha Testing	Beta Testing
The employees of the companies perform alpha testing.	Beta testing is performed by the end-users who are not the employees of the companies.
Alpha testing is performed at the developer's site.	Beta testing is performed at the end-users place.
It uses both black box and white box testing techniques.	It uses the black box testing technique.
Alpha testing is often carried out following the system testing phase and when the product is 70-90 percent complete.	Beta testing is carried out following alpha testing and when the product is 90-95 percent complete.
The main goal of alpha testing is to evaluate the quality of the product.	The main goal of beta testing is to evaluate customer satisfaction.
It requires a specific environment for testing.	It does not require any specific environment for testing.
It is done before the product is launched in the market.	It is done at the time of marketing of the product.

4.8 Overview of Unit testing & Integration testing

❖ UNIT TESTING

- Unit testing is undertaken after a module has been coded and successfully reviewed.
- Unit testing (or module testing) is the testing of different units (or modules) of a system in separately.
- When developer is coding the software it may happen that the dependent modules are not completed for testing, in such cases developers use stubs and drivers to simulate the called (stub) and caller (driver) units. Unit testing requires stubs and drivers, stub represent the called unit and driver represent the calling unit.

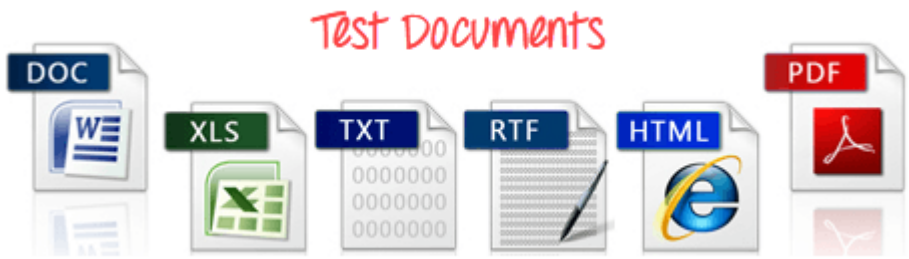
Strategy	Advantage	Disadvantage
Top-Down	Early integration of units Allow for overlaps with top-down design Early demonstration of functionality Redundant lower-level functionality identified	Cost to develop/maintain stub Harder to achieve structural coverage Changes to a unit can impact sibling and lower-level unit tests
Bottom-up	Early integration of units Do not need to know structural design details Compatible with object oriented software	Becomes complicated towards the top of the hierarchy Changes to a unit can impact sibling and higher-level unit tests Can cause bottlenecks in the test schedule

❖ INTEGRATION TESTING

- Integration testing is the second level of the software testing process comes after unit testing.
- In this testing, units or individual components of the software are tested in a group.
- The focus of the integration testing level is to expose defects at the time of interaction between integrated components or units.

Advantages	Disadvantages
Developers who want to learn what a unit does and how to use it might look at the unit tests to better understand the API.	By definition, unit testing focuses on a single piece of software. As a result, it is incapable of detecting integration or system-wide issues.
Unit testing enables the programmer to improve code and confirm that the module functions correctly.	Evaluating all execution paths is not possible in unit testing, so unit testing cannot catch every error in a program.

4.9 Test Documentation – test case templates



- Test documentation is documentation of artifacts created before or during the testing of software.
- It helps the testing team to estimate testing effort needed, test coverage, resource tracking, execution progress, etc.
- It is a complete suite of documents that allows you to describe and document test planning, test design, test execution, test results that are drawn from the testing activity.

❖ Types of Testing Documents

Test policy	It is a high-level document which describes principles, methods and all the important testing goals of the organization.
Test strategy	A high-level document which identifies the Test Levels (types) to be executed for the project.
Test plan	A test plan is a complete planning document which contains the scope, approach, resources, schedule, etc. of testing activities.
Requirements Traceability Matrix	This is a document which connects the requirements to the test cases.
Test Scenario	Test scenario is an item or event of a software system which could be verified by one or more Test cases.
Test case	It is a group of input values, execution preconditions, expected execution postconditions and results. It is developed for a Test Scenario.
Test Data	Test Data is a data which exists before a test is executed. It used to execute the test

❖ Advantages of Test Documentation

- The main reason behind creating test documentation is to either reduce or remove any uncertainties about the testing activities. Helps you to remove ambiguity which often arises when it comes to the allocation of tasks
- Documentation not only offers a systematic approach to software testing, but it also acts as training material to fresher's in the software testing process
- It is also a good marketing & sales strategy to showcase Test Documentation to exhibit a mature testing process
- Test documentation helps you to offer a quality product to the client within specific time limits
- In Software Engineering, Test Documentation also helps to configure or set-up the program through the configuration document and operator manuals
- Test documentation helps you to improve transparency with the client

❖ Disadvantages of Test Documentation

- The cost of the documentation may surpass its value as it is very time-consuming
- Many times, it is written by people who can't write well or who don't know the material
- Keeping track of changes requested by the client and updating corresponding documents is tiring.
- Poor documentation directly reflects the quality of the product as a misunderstanding between the client and the organization can occur

❖ Summary

- Test documentation is documentation of artifacts created before or during the testing of software.
- The degree of test formality depends on 1) the type of application under test 2) standards followed by your organization 3) the maturity of the development process.
- Important types of Test Documents are Test policy, Test strategy, Test plan, Test case etc.
- QA team needs to be involved in the initial phase of the project so that Test Documentation is created in parallel
- The main reason behind creating test documentation is to either reduce or remove any uncertainties about the testing activities.
- The cost of the documentation may surpass its value as it is very time-consuming