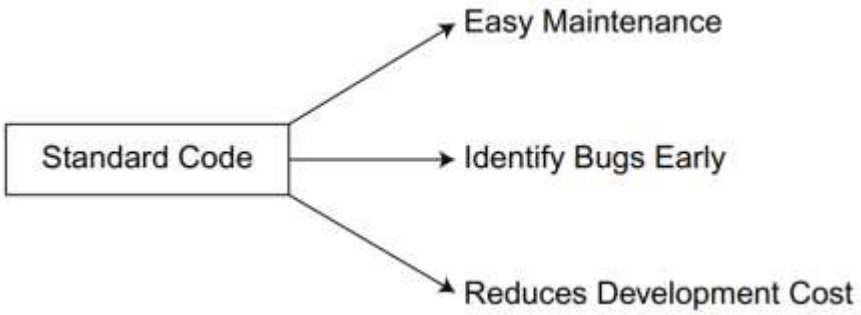
Software Engineering

Unit 4: Software Coding and Testing

INTRODUCTION

- Software coding is the core aspect of software engineering, where Software is actually made.
- Conversion of detailed design specification into the actual software code is called as coding.
- It is performed by people called as coders or programmers.
- The coding phase usually takes lesser time than the design phase.
- Writing standardized code for software not only helps to maintain it properly once it is completely deployed into production but also helps to enhance the code easily at a later point of time.
- It also helps to identify and fix the bugs quickly and easily.
- So writing the proper working code alone is not enough, but focus should also be on writing a ***standard code***.
- Standard code helps to ***reuse the same code*** for some other similar purposes, thereby ***reducing the cost of development***.



Programming Principles

- Computer programming is directly associated with writing a standard code.
- Computer programming means writing a ***standard code, testing the written code, and debugging and maintaining the code***.
- Programming can be considered as both ***art*** *and* ***science***.
- Programming is a science as it needs to ***follow a set of engineering principles and guidelines.***
- It is also being classified as art as there are lots of possibilities of ***using creative and innovative minds.***
- Following are the ***five general programming principles:***

  (1). Validity (2). Consistency (3). Maintainability (4). Readability (5). Usability

1). Validity:

- The program must give the correct result which is valid.
- ***For example***, let us consider a program intended to add two numbers say add (x, y).
- When we pass the value (4, 5), it should give the value 9 as output and when we pass the value (–4, 5), it should give the value 1 as output.

2). Consistency:

- The program must do ***repeatedly what it intends to do***. The program should give the output consistently.
- For example, if add (4.2, 5.4) gives the output as 10, add (5.4, 4.2) also should give the output as 10.

Programming Principles…

3). Maintainability:

- The program must be easily changeable (addition and modification) and should have proper documentations.

4). Readability:

- The program must be easily readable so that it is easily maintainable.

5). Usability:

- The program must be usable for the specific purpose without any trouble.

PROGRAMMING GUIDELINES

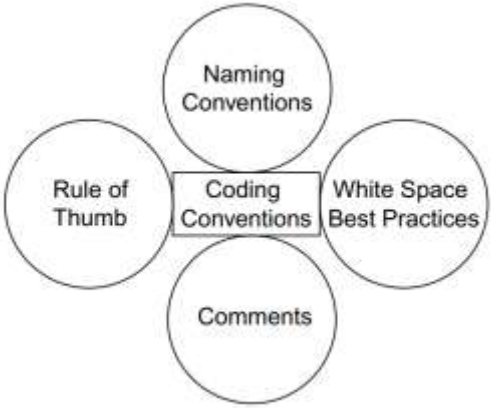
- Guideline means best practices.
- Programming best practices (guidelines) includes programming practices (coding conventions), naming conventions, comments, and programming principles rule of thumb.
- Programming guidelines will vary across programming languages, but still there are general guidelines that can be followed across the languages.
- Writing a standardized code for software not only helps to maintain it properly once it is completely deployed into production but also helps to enhance the code easily at later point of time.
- Standard coding also helps to identify and fix the bugs quickly and easily and ***so writing the proper working code alone is not enough.***
- Focus should also be on writing a standard code.
- Coding conventions can be documented in a central place so that the team can follow it; it can also be informal.
- Not following some or all of the rules (guidelines) will not impact the output (functionality) of the code.

CODING CONVENTIONS (PROGRAMMING BEST PRACTICES)

- Coding conventions are a set of best practices that helps to write the software code in an efficient manner.
- Coding conventions cover the aspects of programming style, programming practices, and methods.
- This will vary from one programming language to another.
- For example, Java programming language has a set of programming practices and C# has its own set of programming practices and guidelines.
- Advantages of coding convention:
  1. Code becomes reusable saving money for all.
  2. It is easy to maintain.
  3. Enhancing the code becomes easy.
  4. It saves cost for the company.
  5. It helps to identify coding problems quickly.
  6. Knowledge transfer becomes easy.

CODING CONVENTIONS (PROGRAMMING BEST PRACTICES)

- In general, coding conventions cover the following:
  1). Naming convention
  2). Rule of thumb
  3). Comment writing within the code
  4). Declarations and statements best practices
  5). White space best practices
  6). Bad practices to be avoided



1). Naming convention:

- Naming convention is a set of rules for choosing the character sequence to be used for identifiers (names).
- Naming convention can be applied to:
  1. File name
  2. Folder name
  3. Variable name
  4. Function (method) name
- Many companies nowadays define their own coding conventions to be used across all projects.
- A = B × C is syntactically correct, but the purpose of this code snippet is not known.
- We can write the same code as below, which has more meaning and conveys the purpose.
- Simple Interest = (PxRxN)/100

1). Naming convention

i). Length of the Identifiers :

- A fundamental element of naming conventions is the rule related to the length of the identifier and the combination of characters allowed in the identifier.
- Some considerations in deciding the length of the identifiers are as follows:
  - Shorter identifiers are preferred over lengthy ones (difficult to identify and remember).
  - Longer identifiers are preferred because it is easy to encode them.
  - Extremely short identifiers (example: a, b, c) are very difficult to distinguish each other.
  - Longer identifiers may not be preferred because of visual clutter.

ii). Letter Case and Numerals:

- Some naming conventions limit the letter case of the variable to be only in uppercase or lowercase (WEIGHT or weight).
- Some naming conventions do not limit the letter case of the variable but attach a defined meaning with the letter case (Chair Weight).
- Some naming conventions specify that the variable name should start with an upper case and should use at least one numeric within (password characters have this kind of restrictions).

1). Naming convention…

iii). Multiple Word Identifiers:

- A single word may not be enough to specify the clear purpose and meaning of an identifier, and we may need to use a combination of identifiers to make it meaningful (e.g., Chair Weight).
- When we use a combination of words for a single identifier, it is called as “compound identifier” (which contains more than one word).
- Most programming languages do not allow white spaces in the identifiers and so we need a delimiter in between for that purpose.
  - **Delimiter-separated Words** Using delimiters in between to separate the words (in an identifier) helps to understand the meaning easier. Underscore (“_”) and Hyphen (“-”) are most commonly used delimiters (e.g.,: Chair_Weight or Chair-Weight)
  - **Letter-case-separated Words** Without using delimiters in between (in an identifier), we can use uppercase characters for words that help to understand the meaning easier. It is also called as CamalCase (e.g.,: Chair Weight).

Benefits of Naming Convention

- **Benefits of Naming Convention**
- Following are some of the benefits of naming convention:
  - It provides additional information (i.e., metadata) about the use.
  - It promotes consistency within a development team.
  - It enhances clarity during ambiguity.
  - It helps to avoid naming collisions (two or more people using the same name for different purposes).
  - It provides a better understanding in case of code reuse.

2). Programming Principles and Rule of Thumb

- Senior software engineers in various IT organizations do follow various programming principles and rules of thumb which are derived based on their experience and skills.
- Choose appropriate and sufficient data type for a variable ensuring the size is not large (use integer data type, short data type and fl oat data type appropriately).
- Try to avoid declaring all variables as global variable.
- Keep specific name for variables and methods rather than using generalized name.
- Use and keep variables and methods for one and only purpose and try to avoid multipurpose variable and methods.
- Avoid writing “public class” for security purpose.
- While using database connection, create the connection object as late as possible and release it as early as possible.
- Avoid using database connection using specific user’s credentials. We cannot reuse those connections.

3). Comment writing within the code

- Properly commented code serves no purpose for the compilation as well as executing the code, but it improves the readability of the code.
- It also helps to understand the purpose and business logic behind the code.
- All software engineers write the comments on the first version of the code, but it is really a challenge to keep these comments up to date with further changes in the code.
- Following are some of the recommendations of commenting techniques:
  - Keep the comment always up to date while modifying the code.
  - Write the comments first before changing the actual code.
  - Write the comments in the beginning of every method, indicating the purpose of the method, assumptions, and limitations.
  - Try to avoid end-of-line comments.
  - Prior to deployment, remove all unnecessary and temporary comments to avoid confusion.
  - Before variable declarations, write the purpose of the variable in comments which will enhance the readability and understanding of the code.
  - Avoid writing series of asterisks in the comments that may look good, but maintaining it is difficult.
  - Use complete sentences while writing comments.
  - Comments should help to understand the code, not add confusion.
  - Use uniform style throughout the code for comments, with consistent punctuation and structure.

4). White Space Best Practices

- Blank lines and white spaces improve the readability of the code.
- It logically sub-divides the codes.
- **Blank Lines**

  1. Two blank lines can be used between sections within the source code
  2. Two blank lines can be used between class definitions within the source Code
  3. Two blank lines can be used between interface definitions within the source code
  4. One blank line can be used between methods before a block within the source code
- **Blank Spaces**

1. A blank space should always be placed after comma in any argument list.
2. Any keyword followed by a parenthesis must be separated by a space.

KEY CONCEPTS IN SOFTWARE CODING

- In the world of software development, coding is considered as the most ***critical part*** because at this stage the real software gets ***written.***
- Unless proper focus is maintained for keeping the standard of this ***stage high***, ***the software will break at the later stages***, causing huge ***effort in fixing the defects*** called bugs.

1). Structured Programming:

- It represents the usage of a ***logical structure*** on the program being written to make it more ***readable, efficient, reliable, and easily maintained***.
- In a typical structured programming language, the overall program is divided into logical structure for meaningful implementation.
- ***Common functions across the structures are taken out and coded in a separate module*** (or separate program) itself so that the ***functions can be reused and memory usage also minimized.***
- ***Modules are tested separately*** before fitting into the overall programming structure.
- The most common methodology employed was developed by Dijkstra.
- He suggests ***avoiding "GO TO" statement***.
- In this model, ***programs are divided into sections and then subsections***.
- Each ***subsection has only one entry point and one exit point*** and in which control is passed downward through the structure from the top.

KEY CONCEPTS IN SOFTWARE CODING…

- There are three types of control namely i). sequence, ii). selection, and iii). repetition.
- ***Sequence*** refers to ***the orderly execution*** of statements one by one.
- ***Selection*** refers to the ***selection of statements based on some conditions***. This is usually expressed with keywords such as ***if..then..else..endif, switch, or case***.
- ***Repetition*** refers to the same ***statement executed repeatedly depending on a condition*** such as ***for loop, while loop, and do loop.***

2). Information Hiding:

- Users will be ***exposed to only the required details***.
- In a good OO design/code, the object should ***reveal its data only through the specified interfaces or methods***.
- The attributes which are ***only for the internal use of the object and not to be revealed are completely hidden from other objects by declaring it "private".***
- This also ***protects the object attributes getting changed erroneously*** while some other attributes are getting changed.
- Information hiding principle is used to ***prevent other programmers or other program to change this program – intentionally or unintentionally.***

KEY CONCEPTS IN SOFTWARE CODING…

3). Top-Down and Bottom-Up Coding

Top-down coding: Here the implementation starts with the top of the hierarchy.
The top main module is written first before writing the bottom sub-modules.

Testing top-down coding: Top main module is written first and hence the test cases are written completely for it.

A stub is written in place of bottom sub-modules.

Bottom-up coding: It is the reverse of top-down coding, and here the implementation starts with the bottom sub-modules.

The top main module is written after that until it reaches the top of the hierarchy.

Testing bottom-up coding: Bottom sub-modules are written first, and hence the test cases are written completely for it.

A ***driver module*** is written to involve those modules under testing.

4). Code Sharing

- In software engineering also, third-party codes are being used (integrated) while writing the code.
- For example, credit card payment gateway is a commonly used code share.
- PayPal integration is also another example of code share.

Software Testing

Introduction to Software Testing

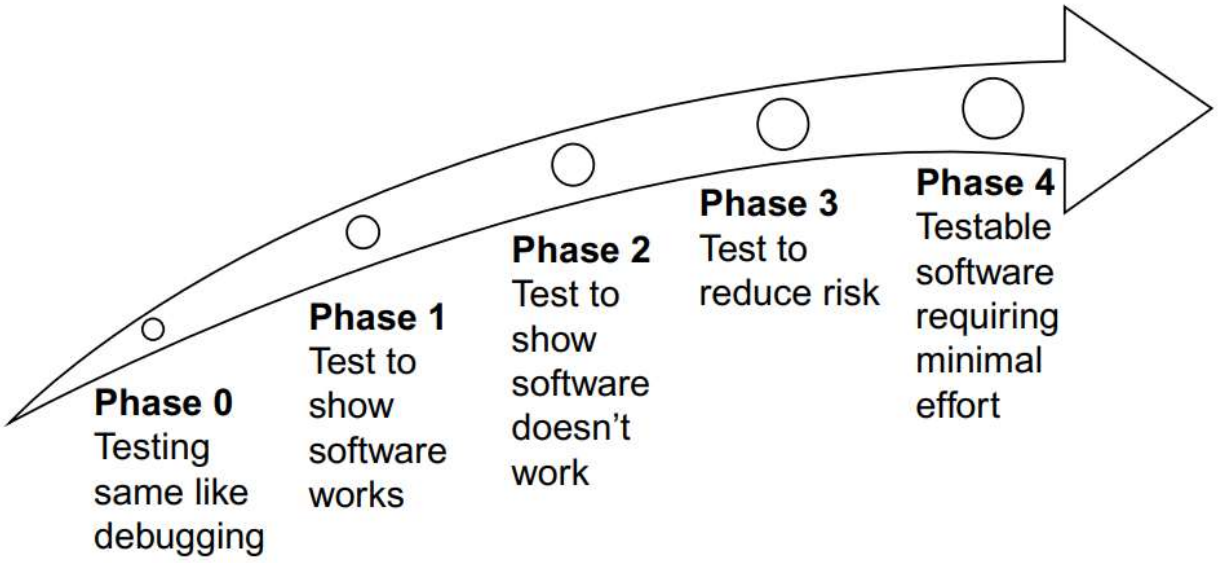
- Till 1970 testing was part and parcel of development, and no separate testing teams were involved till then.
- One may wonder if testing is really required if all the development was done as per the specification, and also if the developers do a good job then where and what is the necessity for software testing.
- Software systems were typically built with multiple evolving requirements and stake holder expectations.
- Because software is an evolving product, there is typically nothing called as complete requirements or requirements freeze.
- Requirements keep coming and it is up to the project team and stake holders to decide and plan.
- Considering these facts it is very important that software testing must be done using some specific methodologies and techniques.

PSYCHOLOGY OF TESTING

- Over the years as software testing has evolved from being synonymous with debugging to a separate independent competency aimed at improving the overall quality of the product.
- In the very beginning **(Phase 0),** the ***aim is to get a bug-free product without any coding errors.***
- ***Testing is like debugging*** so it is not really testing yet.
- A bug-free code does not mean that the software product will meet the expected requirements.
- The next level **(Phase 1)** is to show that the software works.
- This is similar to the development team goal, but testing is clearly different from debugging.
- But the problem with this is that ***no software can work under all cond***itions because this means that all the conditions have been tested.
- In a given time, ***only a subset of conditions can be tested due to the infinite possibilities.***
- The next level of thinking **(Phase 2**) is to find areas where the software does not work.
- This is completely opposite to the way of thinking earlier which was along with the developers' line of thinking.
- The Advanced level **(Phase 3)** of thinking evolved based on the ***level of confidence the tester develops on the software he is testing.***
- ***Testing cannot be done forever either positive or negative cases,*** a logical end needs to be met and the ***product should be released*** once there is a desired level of confidence in the product.
- This confidence would come if harsh or difficult test cases are consistently passed by the software.

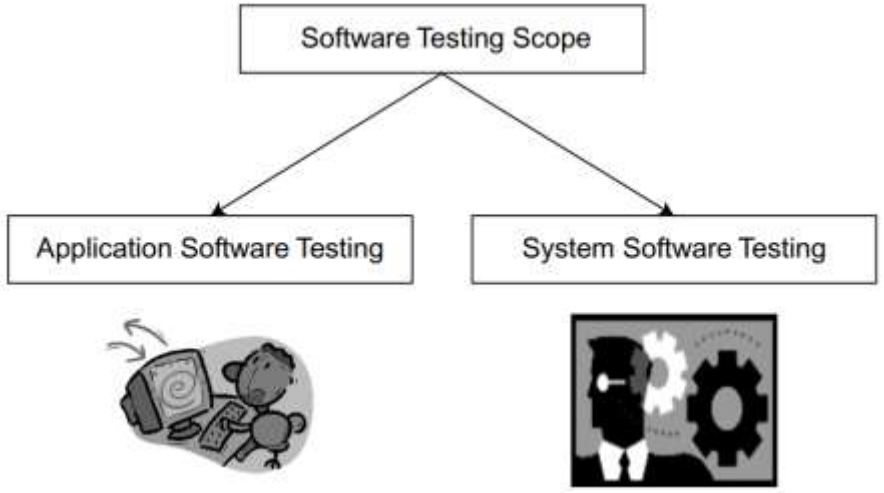
PSYCHOLOGY OF TESTING…

- Finally, **(Phase 4)** of ***thinking is based on a strong expertise in the areas to be tested and identification of what can be tested or not.***
- At this level the testing is optimized and it is most focused and precise.
- The tester has the expertise to identify the key problem areas and the skill to ***design test cases*** to capture them.
- This comes with the ***comprehensive knowledge of the multiple testing techniques*** and their practical applications.



SOFTWARE TESTING SCOPE

- Typically software testing scope is based on the ***functionality and purpose*** of the programs/software, that is, whether the programs are used for ***data processing applications*** (application software) or the ***programs are used to run the computer system itself*** (system software).
- These can be broadly classified into two main areas namely, ***applications software testing*** and ***systems software testing***.
- Application software testing is testing the common applications which is seen every day and mostly used for data processing, such as banking system, insurance, payroll, inventory management system, hospital management systems, and also mobile applications.
- Systems software testing is done on the programs written for running the computer/system itself which facilitate, enhance, and control various programs, such as the operating system, complier, and interpreter.



SOFTWARE TESTING OBJECTIVES

- The first objective of software testing is ***to prevent bugs*** from getting into the system.
- This type of testing is commonly ***called as verification testing.***
- The other obvious objective of testing is to find and ***finding bugs***.
- The basis of finding the bugs depends on the following:
  - Whether the ***software product meets the specifications***.
  - Whether the ***software product is fit for the purpose it was designed***.
  - Whether there are ***faults which would cause failure during the operations.***

Verification Testing and Validation Testing

[1]. Verification Testing

The fundamental theme of verification testing is—Is right product being built?

Following techniques helps for verification:

a). Reviews:

- It is a formal process in which the peers or stakeholders examine a piece of work to ensure correctness.
- It could be a specifications document or a technical design or a piece of code, and the purpose is to uncover errors, or lack of expected standards.

b). Walkthroughs:

- It is a formal process where the person responsible for a specific unit of work formally presents his work to a group of reviewers.
- The work has to be circulated to the group prior to the discussion giving ample time for the reviewers to examine, write comments, or prepare questions for the walkthrough meeting.
- The presenter will go through the document line by line during the walkthrough explaining what has been done and why.

c). Inspections:

- It is the most formal type of review. Each and every participant has to go through training to participate in the inspections.
- In the reviews and walkthroughs the presenter is the original owner of the materials presented, but in inspection the presenter has to learn from the owner along with his interpretations and then present it to other participants who are called as inspectors.
- There could be roles like moderators and recorders also assigned to the participants.
- Inspections have been found to be very effective in uncovering bugs for any software delivery specifically with respect to the design documents and the code.

Verification Testing and Validation Testing…

[2]. Validation Testing

- The testing activities that are done to evaluate the software in the executable mode can be called as validation testing.
- Bugs that are discovered during the validation phase are expensive to fix, than the bugs identified in the verification phase.
- Phases of testing along with the development can be planned in parallel.

a). Unit Testing

- The smallest testable units of software which could be individual programs or modules are tested during the unit testing phase.
- This is usually done by the developer, the tester can verify with the developer if this has been done and study the unit test cases and reports.

b). Integration Testing

- After the individual modules have been tested the next step is to ensure the interactions with in the software modules.
- This type of testing is called as integration testing; typically bugs could be found at the interfaces and the communication points between the modules.

Verification Testing and Validation Testing…

c). Functional Testing

- This is done to verify the end-to-end functionalities of the modules.
- This ensures whether the business requirements of the application have been satisfied.
- The tester prepares and executes the test cases based on the understanding of the business, and the business requirements.

d). System Testing

- In this, all the components of the system—software, hardware, external interfaces are all tested as per the specifications.
- This is done to ensure the delivery of a complete and fully integrated software product.

e). Acceptance Testing

- This is the final phase of testing where the customer will execute a set of test cases to evaluate and accept the product.
- This is executed during the final phase of testing to confirm acceptance of the product.

TYPES OF SOFTWARE TESTING

- There are two commonly used testing types used in the industry:

  1). White box testing

  2). Black box testing

1). White box testing

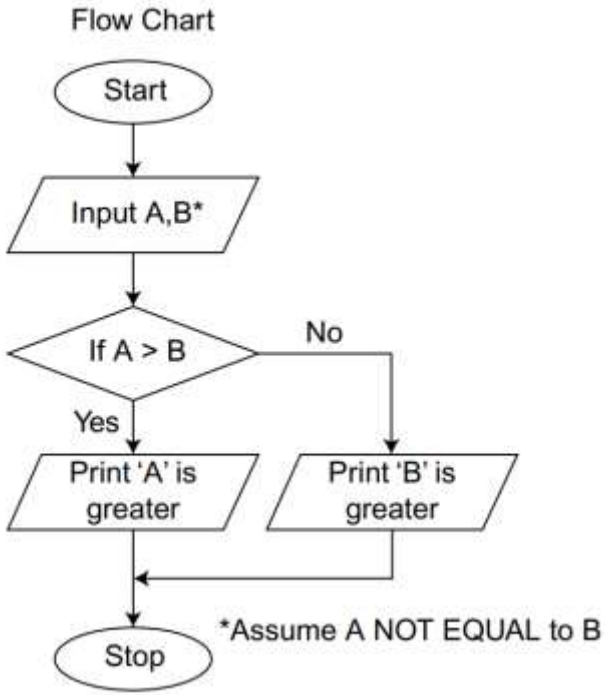
- Structural or white box testing is based on knowledge of internal structure and logic of programs.
- A small application that performs a few simple operations could be white box tested quickly in minutes, while larger programming applications take days, weeks, and even long.
  - White box testing ensures that all independent paths within a module have been exercised at least once.
  - It ensures flow for normal execution and when exceptions are encountered it should throw up meaningful error messages and exit gracefully.
  - No unhandled exceptions should be present under any circumstances.
  - The database interactions during the program flow should also be verified, such as when trying to update a table is not found or a field is not present in the table or if there is no space in the database to create new records.

TYPES OF SOFTWARE TESTING …

White Box Testing Techniques

a). Code coverage

- Each segment of code control structure is executed at least once in code coverage.
- Each branch in the software code is taken in each possible direction at least once.
- Each independent path through the code is taken in a pre-determined order.
- All possible paths through the code are defined and covered.



TYPES OF SOFTWARE TESTING …

White Box Testing Techniques…

b). Control Structure Testing

- ***Condition Testing:***
  - The value of an integer variable like the age can make the application behave in a certain manner.
  - For example, if age is 18 or above certain rules may be triggered and if age is less than 18 certain other conditions may be triggered.
- ***Data flow testing:***
  - When a variable is defined in a program, it is actually allocated a certain portion of the memory so that values could be assigned and reassigned to this memory location during the course of a program execution.
  - Data flow testing is analyzing the location and variables assigned or changed or impacted during the execution of a program.