Need for Software Engineering

Need for Software Engineering

- **Handling Big Projects:** in order to handle large projects without any issues.
- **Manage the cost:** plan everything and reduce all those things that are not required.
- **Decrease time:** It will save a lot of time if you are developing software using a software engineering technique.

Reliable software: It is the company's responsibility to deliver software products on schedule and to address any defects that may exist.

Effectiveness: Effectiveness results from things being created in accordance with the standards.

Reduces complexity: Large challenges are broken down into smaller ones and solved one at a time in software engineering. Individual solutions are found for each of these issues.

Productivity: Because it contains testing systems at every level, proper care is done to maintain software productivity.

Quality Management: Better procedure of software development provides a better and quality software product.

Maintainability: Clear documentation and well-organized code make it easier to update and fix issues later.

Software developed without a software engineering process

Unstructured development:

Developers might jump straight into coding without clearly defining requirements, system architecture / design, leading to haphazard code structure.

- **Poor quality:**

  Without rigorous testing and quality checks, the software is likely to have many bugs and may not function reliably under different conditions.

- **Maintenance challenges:**

  Modifying or adding features to the software later becomes difficult due to the lack of documentation and poorly organized code.

- **Potential for project failure:**

  Without proper planning and risk management, the project might exceed deadlines, budget, and fail to meet user expectations.

Source: NMG Technologies.com

Software Charateristics

Software Characteristics

1. Software is developed or engineered; it is not manufactured in the classical sense:

- Although some similarities exist between software development and hardware manufacturing, few activities are fundamentally different.
- In both activities, high quality is achieved through good design, but the manufacturing phase for hardware can introduce quality problems than software.

2. The software doesn't "wear out.":

- Software is not susceptible to the environmental maladies
- When a hardware component wears out, it is replaced by a spare part. There are no software spare parts.
- Every software failure indicates an error in design or in the process through which the design was translated into machine-executable code.
- Implication is clear—the software doesn't wear out. But it does deteriorate.

Software Characteristics

1. Software is developed/engineered; it is not manufactured in the classical sense:

- Although some similarities exist between software development and hardware manufacturing, few activities are fundamentally different.
- In both activities, high quality is achieved through good design, but the manufacturing phase for hardware can introduce quality problems than software.

2. The software doesn't "wear out.":

- Software is not susceptible to the environmental maladies
- When a hardware component wears out, it is replaced by a spare part. There are no software spare parts.
- Every software failure indicates an error in design or in the process through which the design was translated into machine-executable code.
- Implication is clear—the software doesn't wear out. But it does deteriorate.

3. Functionality

The functionality of software refers to its ability to perform and function according to design specifications. In simple terms, software systems should function correctly, i.e. perform all the functions for which they are designed.

4. Usability (User-friendly)

The user-friendliness of the software is characterized by its ease of use. In other words, learning how to use the software should require less effort or time.

5. Efficiency

Essentially, it refers to the software's ability to utilize human and system resources such as time, effort, CPU, memory, computation power, network bandwidth, files, databases, etc., as effectively and efficiently as possible.

6. Flexibility

Software Flexibility refers to the ability of the software solution to adapt to potential or future changes in its requirements. When evaluating the flexibility of software, look at how simple it is to add, modify, or remove features without interfering with the current operation.

7. Reliability

The reliability of a software product describes the likelihood it will operate without failure over a specified period of time under certain conditions. It determines the ability of software to maintain its level of performance (provide desired functionality) under specified conditions for a specified period of time.

8. Maintainability

Maintainability refers to how easily you can repair, improve and comprehend software code. In some ways, maintaining is similar to being flexible.

9. Portability

Software portability is a critical factor that cannot be ignored. Portability refers to the ability to use software in different environments. This is the ease with which software can be ported from one platform to another without (or with minimal) changes, while obtaining similar results.

10. Integrity

There are multiple interpretations of software integrity. Some people tend to associate integrity with security, believing it is resistant to hacks and privacy violations. To others, high integrity means that the software cannot be modified without authorization

Assessment

1. Software is Manufactured(True / False)
2. _____matches the bath tub curve?.
3. Hardware Wears out! How?
4. Software reliability is _____
5. If we do not use Software Engineering process. It ends in__

   a. Quality Software
   b. Potential Failure
   c. Structured Development
   d. Maintenance Challenge