Introduction to Software Engineering

...

By Er. Abinash Adhikari

Introduction to Software

- Software is a collection of instructions, data, or programs used to operate computers and execute specific tasks.
- It is a critical component of the digital world, allowing hardware to perform useful functions.
- Software can be broadly categorized into two main types:
- System software
- Application software

System Software

System software provides the basic functions necessary for the computer to operate and includes the following:

- Operating Systems (OS):
    - Manages the hardware and software resources of the computer. Examples include Windows, macOS, Linux, and Android.
- Device Drivers:
    - Allow the OS to communicate with hardware devices like printers, graphics cards, and network cards.
- Utilities:
    - Perform maintenance tasks such as disk cleanup, antivirus scanning, and system diagnostics.

Application Software

Application software is designed to help users perform specific tasks and includes:

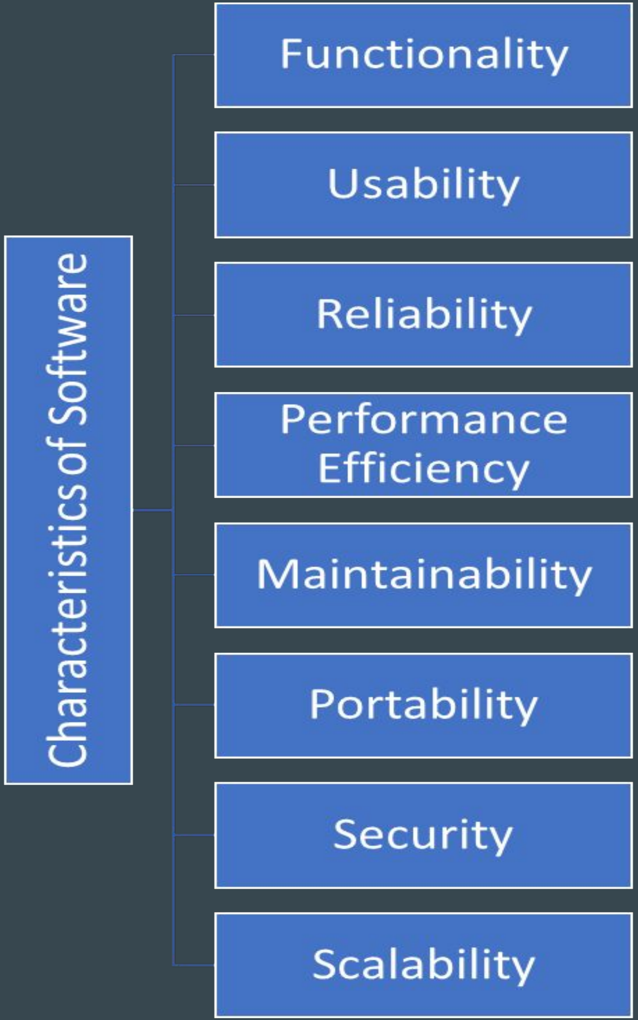
- ❖ Productivity Software: Includes word processors, spreadsheets, and presentation software (e.g., Microsoft Office, Google Workspace).
- ❖ Web Browsers: Tools for accessing the internet (e.g., Chrome, Firefox, Safari).
- ❖ Media Players: For playing audio and video files (e.g., VLC Media Player, Windows Media Player).
- ❖ Graphics Software: For creating and editing images (e.g., Adobe Photoshop, GIMP).
- ❖ Games: Interactive entertainment software (e.g., Fortnite, Minecraft).

ASPECT	SOFTWARE	PROGRAM
Definition	Software is a set of program that enables the hardware to perform specific task.	A computer program is a set of instruction that is used as a process of creating a software program using a programing language.
Categories	Various software categories include application software, system software, computer programming tools, etc.	There are no such categories of programs.
Size	The size of software generally ranges from megabytes (Mb) to gigabytes (Gb).	The program size generally ranges from kilobytes (Kb) to megabytes (Mb).
Developer Expertise	Software is usually developed by people having expert knowledge and experience as well as are trained in developing software and are also referred to as software developers.	Programs are usually developed by the person who is a beginner and has no prior experience.
Nature	Software’s can be a program that generally runs on computer.	Programs cannot be a software.

Software Components

- User Interface (UI) Components:
  - These are responsible for displaying information to the user and taking user input. Examples include buttons, forms, and menus.
- Business Logic Components:
  - These handle the core operations and rules of the software, such as calculations, decision-making, or data processing.
- Data Access Components:
  - These manage the interaction between the software and databases or data sources, handling data storage, retrieval, and manipulation.
- Service Components:
  - These provide services to other parts of the system, such as authentication, logging, or notifications.
- Utility Components:
  - These provide common functionalities that are used throughout the system, like error handling, file management, or encryption.

Characteristics Of Software



1. **Functionality**
   - ➢ Ensures the software performs its intended tasks accurately and meets user requirements. It focuses on what the software does.
2. **Usability**
   - ➢ Refers to how easy and intuitive the software is for users.
   - ➢ A user-friendly interface reduces the learning curve and enhances the experience.
3. **Reliability**
   - ➢ Indicates how consistently the software performs without failures.
   - ➢ Reliable software minimizes downtime and errors.
4. **Performance Efficiency**
   - ➢ Measures how efficiently the software responds and completes tasks.
   - ➢ It includes aspects like speed and resource management.

5. **Maintainability**
   - ➢ Refers to the ease of updating and fixing software. Well-structured code allows developers to add features or correct bugs quickly.
6. **Portability**
   - ➢ Ensures that the software can be used on different platforms or environments without modification. It provides deployment flexibility.
7. **Security**
   - ➢ Protects the software from unauthorized access and vulnerabilities, safeguarding user data and system integrity.
8. **Scalability**
   - ➢ The ability of software to grow and handle more users or data without requiring major changes. It supports business growth.

Generic View of Software Engineering

- ❖ Software engineering is the application of engineering principles to software development in a systematic way.
- ❖ Key Phases:
  - ➢ Requirement Analysis: Understand the problem and needs.
  - ➢ Design: Planning the architecture of the system.
  - ➢ Development: Writing code and constructing the software.
  - ➢ Testing: Verifying that the software works correctly.
  - ➢ Maintenance: Ongoing support and updates.

Importance of Software Engineering

- Reduces Complexity:
  - Breaks down large projects into manageable tasks.
- Ensures Quality:
  - Uses testing and validation to produce reliable software.
- Promotes Scalability:
  - Designs software that can grow and adapt.
- Enhances Collaboration:
  - Follows standards and methodologies that support teamwork.

1. Reduces Complexity

- ❖ Software engineering applies systematic approaches to handle the inherent complexity of large software projects. By breaking down a project into smaller, manageable components (also known as modularization), developers can focus on solving individual parts of the problem. This method, often called divide and conquer, makes it easier to develop, test, and maintain software systems.
- ❖ **Example:**
  - ➢ In a large-scale e-commerce platform, the system can be divided into modules like user authentication, product catalog, order processing, and payment gateway. Each module can be worked on by different teams without affecting the overall system.

2. Ensures Quality

- Quality assurance is a critical aspect of software engineering. Through rigorous testing, validation, and verification, software engineers ensure that the product meets specified requirements and performs reliably under different conditions. This reduces the chances of errors, bugs, and failures in the final product, ultimately leading to higher customer satisfaction.
- **Example:**
  - Before releasing a software application, it goes through multiple stages of testing such as unit testing, integration testing, and user acceptance testing (UAT) to identify and fix issues early.

3. Promotes Scalability

- Scalability refers to a software system's ability to handle an increasing amount of work, or its potential to expand. Software engineering practices, such as using scalable architecture designs and efficient algorithms, ensure that the system can grow as user demands increase, without a complete redesign.
- **Example:**
  - A small social media platform may start with a few thousand users, but with proper software engineering, it can be scaled to support millions of users by optimizing data storage, server handling, and load balancing.

4. Enhances Collaboration

- Software engineering promotes collaboration by introducing standardized processes, methodologies (like Agile or Waterfall), and tools that allow multiple team members to work on the same project efficiently. This involves using version control systems, coding standards, and documentation that help in managing contributions from different team members and keeping track of changes.
- **Example:**
  - In an Agile development environment, teams work together through sprints (short, time-boxed work phases), and use tools like Git for version control, ensuring that everyone can contribute while maintaining the integrity of the codebase.