FREE EBOOK

INTRODUCTION TO SOFTWARE

PHASE-II STUDY NOTES

FOR NABARD GR. A IT OFFICER EXAM

FOR NABARD GR. A IT OFFICER EXAM

Introduction to Software

Phase-II Study Notes for NABARD Gr. A IT Officer Exam

Introduction

- Software is a set of instructions, data or programs used **to operate computers and execute specific tasks.**
- It is the opposite of hardware, **which describes the physical aspects of a computer.**
- Software is a generic term used to refer **to applications, scripts and programs that run on a device.**
- It can be thought of as the variable part of a computer, while the hardware is the invariable part.
- The two main categories of software are **application software and system software.**
- **An application is software that fulfils a specific need or performs tasks.** System software is designed to run a computer's hardware and provides a platform for applications to run on.
- Other types of software include programming software, which provides the programming tools software developers need; **middleware, which sits between system software and applications; and driver software**, which operates computer devices and peripherals.

Examples and Types of Software

Among the various categories of software, the most common types include the following:

1. Application Software

The most common type of software, application software is a computer software package that performs a specific function for a user, or in some cases, for another application. An application can be self-contained, or it can be a group of programs that run the application for the user.

Examples of modern applications include office suites, graphics software, databases and database management programs, web browsers, word processors, software development tools, image editors and communication platforms.

2. System Software

These software programs are designed to run a **computer's application programs and hardware**. System software coordinates the activities and functions of the hardware and software.

In addition, it controls the operations of the computer hardware and provides an environment or platform for all the other types of software to work in. The OS is the best example of system software; it manages all the other computer programs.

Other examples of system software include firmware, computer language translators and system utilities.

3. Driver Software

Also known as device drivers, this software is often considered a type of system software.

Device drivers control the devices and peripherals connected to a computer, enabling them to perform their specific tasks. Every device that is connected to a computer needs at least one device driver to function.

Examples include software that comes with any nonstandard hardware, including special game controllers, as well as the software that enables standard hardware, such as USB storage devices, keyboards, headphones, and printers.

4. Middleware

The term middleware describes software that mediates between application and system software or between two different kinds of application software.

For example, middleware enables Microsoft Windows to talk to Excel and Word. It is also used to send a remote work request from an application on a computer that has one kind of OS, to an application on a computer with a different OS. It also enables newer applications to work with legacy ones.

5. Programming Software

Computer programmers use programming software to write code. Programming software and programming tools enable developers to develop, write, test and debug other software programs.

Examples of programming software include assemblers, compilers, debuggers, and interpreters.

How Does a Software Work?

All software provides the directions and data computers need to work and meet users' needs. **However, the two different types -- application software and system software -- work in distinctly different ways.**

- **Application Software**

  Application software consists of many programs that perform specific functions for end-users, such as writing reports and navigating websites.

  Applications can also perform tasks for other applications. Applications on a computer cannot run on their own; they require a computer's OS, along with other supporting system software programs, to work.

  These desktop applications are installed on a user's computer and use the computer memory to carry out tasks. They take up space on the computer's hard drive and do not need an internet connection to work.

  However, desktop applications must adhere to the requirements of the hardware devices they run on.

  Web applications, on the other hand, only require internet access to work; they do not rely on the hardware and system software to run. Consequently, users can launch web applications from devices that have a web browser.

  Since the components responsible for the application functionality are on the server, users can launch the app from Windows, Mac, Linux, or any other OS.

- **System Software**

  System software sits between the computer hardware and the application software. Users do not interact directly with system software as it runs in the background, handling the basic functions of the computer.

  This software coordinates a system's hardware and software so users can run high-level application software to perform specific actions.

  System software executes when a computer system boots up and continues running as long as the system is on.

Design and Implementation

- The software development lifecycle is a framework that project managers use **to describe the stages and tasks associated with designing software.**

- The first steps in the design lifecycle are planning the effort and then analysing the needs of the individuals who will use the software and creating detailed requirements. **After the initial requirements analysis, the design phase aims to specify how to fulfil those user requirements.**

- The next step is implementation, where development work is completed, and then software testing happens. **The maintenance phase involves any tasks required to keep the system running.**

- The software design includes a description of the structure of the software that **will be implemented, data models, interfaces between system components and potentially the algorithms the software engineer will use.**

- The software design process transforms user requirements into a form that computer programmers can use to do the software coding and implementation. **The software engineers develop the software design iteratively, adding detail and correcting the design as they develop it.**

The Different Types of Software Design include the following:

- **Architectural design:** This is the foundational design, which identifies the overall structure of the system, **its main components, and their relationships with one another using architectural design tools.**

- **High-level design:** This is the second layer of design that focuses on how the system, **along with all its components, can be implemented in form of modules supported by a software stack.** A high-level design describes the relationships between data flow and the various modules and functions of the system.

- **Detailed design:** This third layer of design focuses **on all the implementation details necessary for the specified architecture.**

How to Maintain Software Quality?

- Software quality measures if the software meets **both its functional and non-functional requirements.**

- **Functional requirements identify what the software should do.** They include technical details, data manipulation and processing, **calculations or any other specific function that specifies what an application aims to accomplish.**

- **Non-functional requirements** -- also known as quality attributes -- determine how the system should work. **Non-functional requirements include portability, disaster recovery, security, privacy, and usability.**

- Software testing detects and solves technical issues **in the software source code and assesses the overall usability, performance, security, and compatibility of the product to ensure it meets its requirements.**

The Dimensions of Software Quality include the Following Characteristics:

- **Accessibility:** The degree to which a diverse group of people, including individuals who require adaptive technologies **such as voice recognition and screen magnifiers, can comfortably use the software.**
- **Compatibility:** The suitability of the software for use in a variety of environments, **such as with different OSes, devices, and browsers.**
- **Efficiency:** The ability of the software to perform well without wasting energy, resources, effort, time, or money.
- **Functionality:** Software's ability to carry out its specified functions.
- **Installability:** The ability of the software to be installed in a specified environment.
- **Localization:** The various languages, time zones and other such features a software can function in.
- **Portability:** The ability of the software to be easily transferred from one location to another.
- **Reliability:** The software's ability to perform a required function under specific conditions for a defined period of time without any errors.
- **Scalability:** The measure of the software's ability to increase or decrease performance in response to changes in its processing demands.
- **Security:** The software's ability to protect against unauthorized access, invasion of privacy, theft, data loss, malicious software, etc.

To maintain software quality once it is deployed, developers must constantly adapt it to meet new customer requirements and handle problems customers identify. This includes **improving functionality, fixing bugs, and adjusting software code to prevent issues.**

When it comes to performing maintenance, there are four types of changes developers can make, including:

- **Corrective**: Users often identify and report bugs that developers must fix, including coding errors and other problems that keep the software from meeting its requirements.

- **Adaptive:** Developers must regularly make changes to their software to ensure it is compatible with changing hardware and software environments, such as when a new version of the OS comes out.
- **Perfective:** These are changes that improve system functionality, such as improving the user interface or adjusting software code to enhance performance.
- **Preventive:** These changes are done to keep software from failing and include tasks such as restructuring and optimizing code.

Modern Software Development

- DevOps is an organizational **approach that brings together software development and IT operations teams.**
- It promotes communication and collaboration between these two groups.
- The term also describes the use of iterative software development practices **that use automation and programmable infrastructure. Get the full picture in our ultimate guide to DevOps.**

Software Licensing and Patents

- A software license is a legally binding document **that restricts the use and distribution of software.**
- Typically, software licenses provide users **with the right to one or more copies of the software without violating copyright.**
- The license outlines the responsibilities of the parties that enter **into the agreement and may place restrictions on how the software can be used.**
- **Software licensing terms and conditions generally include fair use of the software**, the limitations of liability, warranties, disclaimers, and protections if the software or its use infringes on the intellectual property rights of others.
- Licenses typically are for proprietary software, which remains the property of the organization, group or individual that created it; **or for free software, where users can run, study, change and distribute the software.**
- Open source is a type of software where the **software is developed collaboratively, and the source code is freely available.**
- With open-source software licenses, **users can run, copy, share and change the software similar to free software.**

- Over the last two decades, software vendors have moved away from selling software licenses **on a one-time basis to a software-as-a-service subscription model.**
- Software vendors host the software in the cloud and **make it available to customers, who pay a subscription fee and access the software over the internet.**
- Although a copyright can prevent others from copying a developer's code, a copyright **cannot stop them from developing the same software independently without copying.**
- A patent, on the other hand, enables a developer to prevent another person from using the functional aspects of the **software a developer claims in a patent, even if that other person developed the software independently.**
- In general, the more technical software is, the more likely it can be patented. **For example, a software product could be granted a patent if it creates a new kind of database structure or enhances the overall performance and function of a computer.**

History of Software

The **term software was not used until the late 1950s**. During this time, although different types of programming software were being created, they were typically not commercially available.

Consequently, users -- mostly scientists and large enterprises -- often had to write their own software.

A Brief Timeline of the History of Software

- **June 21, 1948:** Tom Kilburn, a computer scientist, writes the world's first piece of software **for the Manchester Baby computer at the University of Manchester in England.**
- **The early 1950s:** General Motors creates the first OS, for the IBM 701 Electronic Data Processing Machine. **It is called General Motors Operating System, or GM OS.**
- **1958:** Statistician John Tukey coins the word software **in an article about computer programming.**
- **The late 1960s:** Floppy disks are introduced and **are used in the 1980s and 1990s to distribute software.**
- **Nov. 3, 1971:** AT&T releases the **first edition of the Unix OS.**
- **1977:** Apple releases the Apple II and consumer software takes off.

- **1979:** VisiCorp releases VisiCalc for the Apple II, **the first spreadsheet software for personal computers.**

- **1981:** Microsoft releases MS-DOS, the OS on which many of the early IBM computers ran. **IBM begins selling software, and commercial software becomes available to the average consumer**.

- **The 1980s:** Hard drives become standard **on PCs, and manufacturers start bundling software in computers.**

- **1983:** The free software movement is launched with Richard Stallman's **GNU (GNU is not Unix) Linux project to create a Unix-like OS with source code that can be freely copied, modified, and distributed.**

- **1984:** Mac OS is released to run Apple's Macintosh line.

- **The mid-1980s:** Key software applications, **including AutoDesk AutoCAD, Microsoft Word and Microsoft Excel, are released.**

- **1985:** Microsoft **Windows 1.0 is released.**

- **1989:** CD-ROMs become standard and hold much more data than floppy disks. **Large software programs can be distributed quickly, easily, and relatively inexpensively.**

- **1991:** The Linux kernel, the **basis for the open-source Linux OS, is released.**

- **1997:** DVDs are introduced and able to hold more data than CDs, making it possible **to put bundles of programs, such as the Microsoft Office Suite, onto one disk.**

- **1999:** Salesforce.com uses cloud computing **to pioneer software delivery over the internet.**

- **2000:** The term software **as a service (SaaS) comes into vogue.**

- **2007:** iPhone is launched, and mobile applications **begin to take hold.**

- **2010 to the present: DVDs are becoming obsolete as users buy and download software from the internet and the cloud.** Vendors move to subscription-based models and SaaS has become common.

Sources Referred

- **WIKI**
- **Information Practice Book**

Get the Complete Notes in our NABARD IT Course!

NABARD Grade A IT Officer Online Course

We have launched a comprehensive course for your all-round preparation for the NABARD Grade A IT Officer post.

Invest early in your NABARD Gr. A IT officer's exam preparations to gain an edge over your competitors.

Cover the basics of **Computer/IT, ESI & ARD, and Decision-Making** through **Crisp Study Notes** and test your understanding through **Objective + Descriptive Mock Tests**.

Take a look at the course highlights given below:

Enroll Here

Connect with us on:

- Telegram
- Discuss Forum
- YouTube