

Text A

Different Types of Software



扫码听课文

Software is a computer program that provides instructions and data to execute user's commands.

1. Application Software

As a user of technology, application software or “apps” are what you engage with the most. They are productive end-user programs that help you perform tasks. The following are some examples of application software that allow you to do specific work:

- MS Excel: It is a spreadsheet software that you can use for presenting and analyzing data.
- Photoshop: It is a photo editing application software by Adobe. You can use it to visually enhance, catalog and share your pictures.
- Skype: It is an online communication app that you can use for video chat, voice calling and instant messaging.

Software applications are also referred to as non-essential software. They are installed and operated on a computer based on the user's requirement. There are plenty of application software that you can use to perform different tasks. The number of such apps keeps increasing with technological advances and the evolving needs of the users. You can categorize these software types into different groups, as shown in the following Table 2-1.

Table 2-1 Application Software Type and Examples

Application Software Type	Examples
Word processing software: Tools that are used to create Word sheets and type documents etc.	Microsoft Word, WordPad, AppleWorks and Notepad
Spreadsheet software: Software used to compute quantitative data	Apple Numbers, Microsoft Excel and Quattro Pro
Database software: Used to store data and sort information	Oracle, MS Access and FileMaker Pro

Continued

Application Software Type	Examples
Application Suites: A collection of related programs sold as a package	OpenOffice, Microsoft Office
Multimedia software: Tools used for a mixture of audio, video, image and text content	Real Player, Media Player
Communication Software: Tools that connect systems and allow text, audio, and video-based communication	MS NetMeeting, IRC, ICQ
Internet Browsers: Used to access and view websites	Netscape Navigator, MS Internet Explorer, and Google Chrome
Email Programs: Software used for emailing	Microsoft Outlook, Gmail, Apple Mail

2. System Software

System software helps the user, hardware, and application software to interact and function together. This type of computer software allows an environment or platform for other software and applications to work in. This is why system software is essential in managing the whole computer system.

When you first power up your computer, it is the system software that is initially loaded into memory. Unlike application software, the system software is not used by end-users. It only runs in the background of your device at the most basic level while you use other application software. This is why system software is also called “low-level software”.

Operating systems are an example of system software. All of your computer-like devices run on an operating system, including your desktop, laptop, smartphone, and tablet, etc. Here is a list of examples of an operating system. Let’s take a look and you might spot some familiar names of system software:

- For desktop computers, laptops and tablets: Microsoft Windows, Mac (for Apple devices), Linux.
- Other than operating systems, some people also classify programming software and driver software as types of system software.

3. Programming Software

Programming software are programs that are used to write, develop, test, and debug other software, including apps and system software. For someone who works at a bespoke software development company, for example, this type of software would make their life easier and more efficient.

Programming software is used by software programmers as translator programs. They are facilitator software used to translate programming languages (i.e., Java, C++, Python, PHP, BASIC, etc) into machine language code. Translators can be compilers, interpreters and assemblers. You can understand compilers as programs that translate the whole source code into machine code

and execute it. Interpreters run the source code as the program is run line by line. And assemblers translate the basic computer instructions — assembly code — into machine code.

Different programming language editors, debuggers, compilers and integrated development environment (IDE) are an example of programming software.

4. Driver Software

Driver software is often classified as one of the types of system software. They operate and control devices and peripherals plugged into a computer. Drivers are important because they enable the devices to perform their designated tasks. They do this by translating commands of an operating system for the hardware or devices, assigning duties. Therefore, each device connected with your computer requires at least one device driver to function.

Since there are thousands of types of devices, drivers make the job of your system software easier by allowing it to communicate through a standardized language. Some examples of driver software that you may be familiar with are: printer driver, mouse driver.

Usually, the operating system comes built-in with drivers for mouse, keyboard, and printers by default. They often do not require third-party installations. But for some advanced devices, you may need to install the driver externally. Moreover, if you use multiple operating systems like Linux, Windows and Mac, then each of these supports different variants of drivers. For them, separate drivers need to be maintained for each.

5. Another Classification of Software

Let's discuss five additional subcategories of software and understand them using examples of trendy software.

5.1 Freeware

Freeware software is any software that is available to use for free. They can be downloaded and installed over the internet without any cost. Some well-known examples of freeware are: Google Chrome, Skype, Instagram, Snapchat, Adobe reader.

Although they all fall under the category of application or end-user software, they can further be categorized as freeware because they are free for you to use.

5.2 Shareware

Shareware, on the other hand, are software applications that are paid programs, but are made available for free for a limited period of time known as “trial period”. You can use the software without any charges for the trial period but you will be asked to purchase it for use after the trial ends. Shareware allows you to test drive the software before you actually invest in purchasing it. Some examples of shareware that you must be familiar with are: Adobe PhotoShop, Adobe Illustrator, Netflix App, MATLAB, MCAFEE Antivirus.

5.3 Open Source Software

This is a type of software that has an open source code that is available to use for all users. It can be modified and shared to anyone for any purpose. Common examples of open source software used by programmers are: LibreOffice, PHP, GNU Image Manipulation Program (GIMP).

5.4 Closed Source Software

These are the types of software that are non-free for the programmers. For this software, the source code is the intellectual property of software publishers. It is also called “proprietary software” since only the original authors can copy, modify and share the software. The following are some of the most common examples of closed source software: .NET, Java, Microsoft Office, Adobe Photoshop.

5.5 Utility Software

Utility software is considered a subgroup of system software. They manage the performance of your hardware and application software installed on your computer, to ensure they work optimally. Some features of utility software include: antivirus and security software, file compressor, disk cleaner, disk defragmentation software, data backup software.

6. Conclusion

In conclusion, there can be multiple ways to classify different types of computer software. The software can be categorized based on the function they perform such as application software, system software, programming software, and driver software. They can also be classified based on different features such as the nature of source code, accessibility, and cost of usage.

New Words

software	['sɒftweə]	<i>n.</i> 软件
command	[kə'mɑ:nd]	<i>n.</i> 命令, 指令
end-user	[end-'ju:zə]	<i>n.</i> 最终用户, 终端用户
spreadsheet	['spredʃi:t]	<i>n.</i> 电子表格
analyze	['ænəlaɪz]	<i>vt.</i> 分析
app	[æp]	<i>n.</i> 计算机应用程序 <i>abbr.</i> 应用(application)
non-essential	[nɒn-ɪ'senʃl]	<i>adj.</i> 不重要的, 非本质的
requirement	[rɪ'kwaɪəmənt]	<i>n.</i> 要求, 需求; 必要条件
quantitative	['kwɒntɪtətɪv]	<i>adj.</i> 定量的, 数量(上)的
database	['deɪtəbeɪs]	<i>n.</i> 数据库

suite	[swi:t]	<i>n.</i> (软件的) 套件
mixture	['mɪkstʃə]	<i>n.</i> 混合, 混杂
communication	[kə,mju:nɪ'keɪʃn]	<i>n.</i> 通信; 交流
browser	['braʊzə]	<i>n.</i> 浏览器, 浏览程序
email	['i:meɪl]	<i>n.</i> 电子邮件 <i>vt.</i> 给……发电子邮件
background	['bækgraʊnd]	<i>n.</i> 后台, 背景
desktop	['desktp]	<i>n.</i> 桌面
tablet	['tæblət]	<i>n.</i> 平板电脑
develop	[dɪ'veləp]	<i>v.</i> 开发
test	[test]	<i>v.</i> 测试
debug	[,di:'bʌg]	<i>vt.</i> 调试, 排除故障
bespoke	[br'spəuk]	<i>adj.</i> 定做的
facilitator	[fə'sɪlɪteɪtə]	<i>n.</i> 促进者, 帮助者
compiler	[kəm'paɪlə]	<i>n.</i> 编译器, 编译程序
interpreter	[ɪn'tɜ:pɪtə]	<i>n.</i> 解释器, 解释程序
assembler	[ə'semblə]	<i>n.</i> 汇编程序
editor	['edɪtə]	<i>n.</i> 编辑器, 编辑软件, 编辑程序
debugger	[,di:'bʌgə]	<i>n.</i> 调试器, 调试程序
driver	['draɪvə]	<i>n.</i> 驱动器, 驱动程序
enable	[ɪ'neɪbl]	<i>vt.</i> 使能够, 使可能
assign	[ə'saɪn]	<i>vt.</i> 分配
standardize	['stændədaɪz]	<i>vt.</i> 使标准化; 用标准校检
built-in	[bɪlt-ɪn]	<i>adj.</i> 嵌入的; 内置的
third-party	['θɜ:dpɑ:tɪ]	<i>adj.</i> 第三方的
installation	[,ɪnstə'leɪʃn]	<i>n.</i> 安装
multiple	['mʌltɪpl]	<i>adj.</i> 多重的; 多个的; 多功能的
variant	['veəriənt]	<i>n.</i> 变种, 变异体; 变形, 变量 <i>adj.</i> 不同的, 相异的
separate	['sepəreɪt]	<i>v.</i> (使) 分开, 分离; 分割; 划分; (使) 分离, 分散; 隔开
subcategory	['sʌb'kætɪgəri]	<i>n.</i> 亚类, 子类
freeware	['fri:weə]	<i>n.</i> 免费软件
download	[,daʊn'ləʊd]	<i>v.</i> 下载
shareware	['ʃəweə]	<i>n.</i> 共享软件

trial	['traɪəl]	adj. 试验的
antivirus	['æntɪvaɪrəs]	n. 抗病毒软件
available	[ə'veɪləbl]	adj. 可利用的; 可得到的; 有效的
publisher	['pʌblɪʃə]	n. 发布者, 发表者
proprietary	[prə'praɪətəri]	adj. 专有的, 专利的 n. 所有权, 所有物
optimally	['ɒptəməli]	adv. 最佳地
compressor	[kəm'presə]	n. 压缩程序
accessibility	[æk'sesə'bɪlɪti]	n. 可访问性

Phrases

engage with	接触; 处理
instant messaging	即时通信
a collection of	一组, 一些, 一批
system software	系统软件
power up	加电, 开机
low-level software	低级软件
machine language	机器语言
source code	源代码
machine code	机器代码
assembly code	汇编代码
be classified as	被归类为……, 被认为……
plug into	接入(计算机系统), 插入
open source software	开源软件
open source code	开源代码
closed source software	闭源软件
intellectual property	知识产权
original author	原作者
disk defragmentation software	磁盘碎片整理软件

Abbreviations

IDE (Integrated Development Environment)	集成开发环境
GNU (GNU's Not UNIX 的递归缩写)	革奴计划
GIMP (GNU Image Manipulation Program)	GNU 图像处理程序

Text A 参考译文

软件的不同类型

软件是一种计算机程序，提供执行用户命令的指令和数据。

1. 应用软件

作为技术用户，最常使用的是应用软件或“应用”。它们是高效的最终用户程序，可以帮助你执行任务。以下是一些应用程序软件示例，可让你执行特定工作：

- MS Excel：这是一个电子表格软件，可用于呈现和分析数据。
- Photoshop：它是 Adobe 的图片编辑应用程序软件。可以使用它来可视化地增强、分类和共享图片。
- Skype：这是一个在线通信应用程序，可用于视频聊天、语音呼叫和即时通信。

软件应用程序也称为非必需软件。它们可以根据用户要求在计算机上安装和操作。可以使用许多应用程序软件来执行不同的任务。随着技术的进步和用户需求的不断变化，此类应用的数量持续增加。可以将这些软件类型分为不同的组，如表 2-1 所示。

表 2-1 应用软件类型和示例

应用软件类型	示例
文字处理软件：用于创建文字表和键入文档等工具	Microsoft Word、WordPad、AppleWorks 和 Notepad
电子表格软件：用于计算定量数据的软件	Apple Numbers、Microsoft Excel 和 Quattro Pro
数据库软件：用于存储数据和排序信息	Oracle、MS Access 和 FileMaker Pro
应用程序套件：打包出售的相关程序的集合	OpenOffice、Microsoft Office
多媒体软件：用于混合音频、视频、图像和文本内容的工具	Real Player、Media Player
通信软件：用于连接系统并允许基于文本、音频和视频通信的工具	MS NetMeeting、IRC、ICQ
因特网浏览器：用于访问和查看网站	Netscape Navigator、MS Internet Explorer 和 Google Chrome
电子邮件程序：用于电子邮件发送的软件	Microsoft Outlook、Gmail、Apple Mail

2. 系统软件

系统软件可帮助用户、硬件和应用软件进行交互并协同工作。此类计算机软件是一个环境或平台，让其他软件 and 应用程序运行于其上。这就是系统软件对于管理整个计算机系统至关重要的原因。

首次打开计算机电源时，系统软件就被加载到内存中。与应用程序软件不同，最终用户不会使用系统软件。当使用其他应用程序软件时，它仅在最基本级别的设备后台运行。这就

是系统软件也被称为“低级软件”的原因。

操作系统是系统软件的一个示例。所有类似计算机的设备都在操作系统上运行，包括台式机、笔记本电脑、智能手机和平板计算机等。这是一个操作系统示例的列表。让我们看一下，可能会发现一些熟悉的系统软件名称：

- 对于台式机、笔记本电脑和平板计算机：Microsoft Windows、Mac（用于 Apple 设备）和 Linux。
- 除操作系统外，有些人还将编程软件和驱动程序软件归类为系统软件。

3. 编程软件

编程软件是用于编写、开发、测试和调试其他软件（包括应用程序和系统软件）的程序。例如，对于在定制软件开发公司工作的人来说，这种类型的软件将使他们的生活更轻松、更高效。

编程软件被软件程序员用作转换器程序。它们是用于将编程语言（即 Java、C++、Python、PHP、BASIC 等）转换为机器语言代码的辅助软件。转换器可以是编译器、解释器和汇编器。可以将编译器理解为将整个源代码转换为机器代码并执行的程序。当程序逐行运行时，解释器将运行源代码。汇编器将基本的计算机指令（汇编代码）转换为机器代码。

不同的编程语言编辑器、调试器、编译器和集成开发环境（IDE）是编程软件的示例。

4. 驱动程序软件

驱动程序软件通常被分入系统软件的类型。它们操作和控制插入计算机的设备和外围设备。驱动程序很重要，因为它们使设备能够执行其指定的任务。它们通过为硬件或设备转换操作系统的命令、分配任务来实现此目的。因此，与计算机连接的每个设备都需要至少一个设备驱动程序才能运行。

由于存在数千种设备，因此驱动程序通过允许其用标准化语言进行通信，使系统软件的工作更加轻松。你可能熟悉的一些驱动程序软件示例：打印机驱动程序、鼠标驱动程序。

通常，操作系统默认内置了鼠标、键盘和打印机的驱动程序。它们通常不需要第三方安装。但是对于某些高级设备，可能需要在外部安装驱动程序。此外，如果使用多个操作系统（例如 Linux、Windows 和 Mac），则每个操作系统都支持不同的驱动程序变体。对于它们，需要为每个设备编制单独的驱动程序。

5. 另一种软件分类

让我们讨论软件的另外 5 个子类别，并使用流行的软件示例来了解它们。

5.1 免费软件

免费软件指可以免费使用的任何软件。可以通过互联网免费下载和安装它们。免费软件

的一些著名示例是：Google 浏览器、Skype、Instagram、Snapchat、Adobe Reader。

尽管它们都属于应用程序或最终用户软件的类别，但它们可以进一步归为免费软件，因为它们免费供用户使用。

5.2 共享软件

另一方面，共享软件是付费应用程序，但是在有限的“试用期”内免费提供。你可以在试用期内免费使用该软件，但是在试用期结束后，系统会要求购买该软件以继续使用。共享软件使你可以在实际投资购买软件之前对其进行测试。共享软件常见的一些示例包括：Adobe PhotoShop、Adobe Illustrator、Netflix App、MATLAB、McAfee Antivirus。

5.3 开源软件

这是一种所有用户都可使用的开源代码软件。无论出于何种目的，任何人都可对其进行修改并分享给其他人。程序员使用的开源软件的常见示例包括：LibreOffice、PHP、GNU 图像处理程序（GIMP）。

5.4 闭源软件

这类软件程序员不可免费获得。对于此软件，源代码是软件发行者的知识产权。它也被称为“专有软件”，因为只有原始作者才能复制、修改和分享该软件。以下是闭源软件的一些最常见示例：.NET、Java、Microsoft Office、Adobe Photoshop。

5.5 实用软件

实用程序软件被视为系统软件的子类。它们管理安装在计算机上的硬件和应用程序软件，以确保它们以最佳状态工作。实用程序软件的某些功能包括：防病毒和安全软件、文件压缩器、磁盘清理器、磁盘碎片整理软件及数据备份软件。

6. 结论

总之，可以有多种方法对不同类型的计算机软件进行分类。可以根据软件执行的功能将其分类，例如应用程序软件、系统软件、编程软件和驱动程序软件。还可以根据不同的特征对它们进行分类，例如源代码的性质、可访问性和使用成本。

Text B

The Seven Phases of the System Development Life Cycle



扫码听课文

The system development life cycle enables users to transform a newly-developed project

into an operational one.

The system development life cycle, “SDLC” for short, is a multistep, iterative process, structured in a methodical way. This process is used to model or provide a framework for technical and non-technical activities to deliver a quality system which meets or exceeds a business’s expectations or manage decision-making progression.

Traditionally, the systems development life cycle consisted of five stages. That has now increased to seven phases. Increasing the number of steps helps systems analysts to define clearer actions to achieve specific goals.

Similar to a project life cycle (PLC), the SDLC uses a systems approach to describe a process. It is often used and followed when there is an IT or IS project under development.

The SDLC highlights different stages of the development process. The life cycle approach is used so users can see and understand what activities are involved within a given step. It is also used to let them know that at any time, steps can be repeated or a previous step can be reworked when needing to modify or improve the system.

1. Planning

This is the first phase in the systems development process. It identifies whether or not there is the need for a new system to achieve a business’s strategic objectives. This is a preliminary plan (or a feasibility study) for a company’s business initiative to acquire the resources to build on an infrastructure to modify or improve a service. The company might be trying to meet or exceed expectations for their employees, customers and stakeholders too. The purpose of this step is to find out the scope of the problem and determine solutions. Resources, costs, time, benefits and other items should be considered at this stage.

2. Systems Analysis and Requirements

The second phase is where businesses will work on the source of their problem or the need for a change. In the event of a problem, possible solutions are submitted and analyzed to identify the best fit for the ultimate goals of the project. This is where teams consider the functional requirements of the project or solution. It is also where system analysis takes place — analyzing the needs of the end users to ensure the new system can meet their expectations. Systems analysis is vital in determining what a business’s needs are, as well as how they can be met, who will be responsible for individual pieces of the project, and what sort of timeline should be expected.

There are several tools businesses can use that are specific to the second phase. They include:

- CASE (computer aided systems/software engineering)
- Requirements gathering
- Structured analysis

3. Systems Design

The third phase describes, in detail, the necessary specifications, features and operations that will satisfy the functional requirements of the proposed system which will be in place. This is the step for end users to discuss and determine their specific business information needs for the proposed system. It's during this phase that they will consider the essential components (hardware and/or software) structure (networking capabilities), processing and procedures for the system to accomplish its objectives.

4. Development

The fourth phase is when the real work begins, in particular, when a programmer, network engineer and/or database developer are brought on to do the major work on the project. This work includes using a flow chart to ensure that the process of the system is properly organized. The development phase marks the end of the initial section of the process. Additionally, this phase signifies the start of production. The development stage is also characterized by instillation and change. Focusing on training can be a huge benefit during this phase.

5. Integration and Testing

The fifth phase involves systems integration and system testing of programs — normally carried out by a quality assurance (QA) professional — to determine if the proposed design meets the initial set of business goals. Testing may be repeated, specifically to check for errors, bugs and interoperability. This testing will be performed until the end user finds it acceptable. Another part of this phase is verification and validation, both of which will help ensure the program's successful completion.

6. Implementation

The sixth phase is when the majority of the code for the program is written. Additionally, this phase involves the actual installation of the newly-developed system. This step puts the project into production by moving the data and components from the old system and placing them in the new system via a direct cutover. While this can be a risky and complicated move, the cut-over typically happens during off-peak hours, thus minimizing the risk. Both system analysts and end-users should now see the realization of the project that has implemented changes.

7. Operations and Maintenance

The seventh and final phase involves maintenance and regular required updates. This step is when end users can fine-tune the system, if they wish, to boost performance, add new capabilities or meet additional user requirements.

If a business determines a change is needed during any phase of the SDLC, the company

might have to proceed through all the above life cycle phases again. The life cycle approach of any project is a time-consuming process. Even though some steps are more difficult than others, none are to be overlooked. An oversight could prevent the entire system from functioning as planned.

New Words

development	[dɪ'veləpmənt]	<i>n.</i> 开发, 发展, 进化
multistep	['mʌltɪstep]	<i>adj.</i> 多步的, 多级的
iterative	['ɪtərətɪv]	<i>adj.</i> 重复的, 反复的, 迭代的 <i>n.</i> 反复体
methodical	[mə'thɒdɪkl]	<i>adj.</i> 有方法的; 有条不紊的
framework	['freɪmwɜ:k]	<i>n.</i> 构架; 框架; (体系的) 结构
activity	[æk'tɪvɪtɪ]	<i>n.</i> 活动
deliver	[dɪ'lɪvə]	<i>vt.</i> 发表; 交付 <i>vi.</i> 投递; 传送
exceed	[ɪk'si:d]	<i>vt.</i> 超过; 超越; 胜过 <i>vi.</i> 突出, 领先
progression	[prə'greʃn]	<i>n.</i> (事件的) 连续; 一系列; 发展, 进展
achieve	[ə'tʃi:v]	<i>vt.</i> 取得; 获得; 实现
repeated	[rɪ'pi:tɪd]	<i>adj.</i> 反复的, 再三的, 重复的
rework	[,ri:'wɜ:k]	<i>vt.</i> 重做, 再做; 修订
plan	[plæn]	<i>n.</i> 计划; 打算
strategic	[strə'ti:dʒɪk]	<i>adj.</i> 战略性的, 至关重要的
preliminary	[pri'lɪmɪnəri]	<i>adj.</i> 初步的, 初级的; 预备的; 开端的 <i>n.</i> 准备工作; 初步措施
feasibility	[,fi:zə'bɪlɪtɪ]	<i>n.</i> 可行性, 可能性, 现实性
initiative	[ɪ'nɪʃətɪv]	<i>n.</i> 主动性; 主动权 <i>adj.</i> 自发的; 创始的
customer	['kʌstəmə]	<i>n.</i> 顾客, 客户
solution	[sə'lʊ:ʃn]	<i>n.</i> 解决方案
requirement	[rɪ'kwaɪəmənt]	<i>n.</i> 要求, 需求
submit	[səb'mɪt]	<i>vt.</i> 提交, 呈送
identify	[aɪ'dentɪfaɪ]	<i>vt.</i> 识别; 确定
ultimate	['ʌltɪmət]	<i>adj.</i> 最后的; 极限的 <i>n.</i> 终极

timeline	['taɪmlaɪn]	<i>n.</i> 时间轴, 时间表
satisfy	['sætɪsfaɪ]	<i>vt.</i> 符合, 达到 (要求、规定、标准等) <i>vi.</i> 使满足或足够
essential	[ɪ'senʃl]	<i>adj.</i> 基本的; 必要的; 本质的 <i>n.</i> 必需品; 基本要素
accomplish	[ə'kʌmplɪʃ]	<i>vt.</i> 完成; 达到 (目的)
signify	['sɪgnɪfaɪ]	<i>vt.</i> 表示……的意思; 意味; 预示 <i>vi.</i> 具有重要性, 要紧
integration	[,ɪntɪ'greɪʃn]	<i>n.</i> 集成, 整合
professional	[prə'feʃənl]	<i>adj.</i> 专业的; 职业的 <i>n.</i> 专业人士
determine	[dɪ'tɜ:mɪn]	<i>vt.</i> 决定, 确定; 判定
bug	[bʌg]	<i>n.</i> 缺陷, 瑕疵
interoperability	[,ɪntərəpərə'bɪlətɪ]	<i>n.</i> 互用性, 协同工作的能力
verification	[,verɪfɪ'keɪʃn]	<i>n.</i> 证明; 证实
validation	[,vælɪ'deɪʃn]	<i>n.</i> 确认; 有效; 校验
implementation	[,ɪmplɪmen'teɪʃn]	<i>n.</i> 执行, 履行; 实施, 贯彻; 生效; 完成
involve	[ɪn'vɒlv]	<i>vt.</i> 包含; 涉及, 使参与
actual	['æktʃuəl]	<i>adj.</i> 真实的; 实际的
cutover	['kʌt,əʊvə]	<i>v.</i> 切换, 转换
off-peak	['ɔ:f'pi:k]	<i>adj.</i> 非高峰的
realization	[,ri:ələɪ'zeɪʃn]	<i>n.</i> 实现; 认识, 领会
operation	[,ɒpə'reɪʃn]	<i>n.</i> 运行, 操作, 经营
maintenance	['meɪntənəns]	<i>n.</i> 维护; 维修; 保养
regular	['regjʊlə]	<i>adj.</i> 有规律的, 规则的; 定期的
fine-tune	[faɪn-tju:n]	<i>vt.</i> 调整, 对进行微调
update	[,ʌp'det]	<i>vt.</i> 更新
boost	[bu:st]	<i>vt.</i> 促进, 提高 <i>n.</i> 提高, 增加
time-consuming	[taɪmkən'sju:mɪŋ]	<i>adj.</i> 费时的, 旷日持久的
overlook	[,əʊvə'lʊk]	<i>vt.</i> 忽视, 忽略
oversight	['əʊvəsart]	<i>n.</i> 疏忽; 失察

Phrases

approach to	接近
feasibility study	可行性研究
systems analysis	系统分析
requirements gathering	需求收集, 需求采集
structured analysis	结构化分析
in detail	详细地
network engineer	网络工程师
database developer	数据库开发人员
flow chart	流程图
focus on	致力于; 使聚焦于
systems integration	系统集成

Abbreviations

SDLC (System Development Life Cycle)	系统开发生命周期
PLC (Project Life Cycle)	项目生命周期
IS (Information System)	信息系统
CASE (Computer Aided Systems/Software Engineering)	计算机辅助系统/软件工程
QA (Quality Assurance)	质量保证

Text B 参考译文

系统开发生命周期的 7 个阶段

系统开发生命周期使用户可以将新开发的项目转换为可操作的项目。

系统开发生命周期（简称 SDLC）是一个多步骤的迭代过程，以系统的方式进行构造。此过程用于为技术和非技术活动建模或提供框架，以提供满足或超过企业期望或管理决策进度的质量体系。

传统的系统开发生命周期包括 5 个阶段。现在已增加到 7 个阶段。增加步骤数量有助于系统分析人员定义更清晰的操作从而实现特定目标。

与项目生命周期（PLC）相似，SDLC 使用系统方法来描述过程。当正在开发 IT 或 IS 项目时，通常会使用并遵循它。

SDLC 突出了开发过程的不同阶段。它使用了生命周期方法，因此用户可以查看和了解

给定步骤中涉及哪些活动。还可以让他们知道，在需要修改或改进系统时，可以随时重复步骤或可以重新执行上一步。

1. 规划

这是系统开发过程的第一阶段。它确定是否需要新系统来实现业务的战略目标。这是公司业务计划的一项初步计划（或可行性研究），该计划旨在获取在基础设施上构建修改或改善服务的资源。该公司也可能试图达到或超过其员工、客户和利益相关者的期望。此步骤的目的是找出问题的范围并确定解决方案。在此阶段应考虑资源、成本、时间、收益和其他项目。

2. 系统分析与需求

第二阶段是企业将解决问题的根源或变更需求的阶段。出现问题时，将提交并分析可能的解决方案，以确定最适合项目最终目标的解决方案。此时，团队考虑项目或解决方案的功能要求。也进行系统分析——分析最终用户的需求，以确保新系统能够满足他们的期望。系统分析对于确定业务需求、如何满足需求、由谁来负责项目的各个部分以及预期的时间表至关重要。

企业可以使用几种特定于第二阶段的工具。它们包括：

- CASE（计算机辅助系统/软件工程）。
- 需求收集。
- 结构化分析。

3. 系统设计

第三阶段详细描述了必要的规范、特征和操作，这些规范、特征和操作需满足未来要实施的拟定系统的功能要求。这个步骤中，最终用户讨论和确定他们对拟定系统的特定业务信息需求。在此阶段，他们将考虑系统实现其目标所需的基本组件（硬件和/或软件）的结构（网络功能）、处理过程和步骤。

4. 开发

第四阶段是真正的工作开始的时候，尤其是当程序员、网络工程师和/或数据库开发人员参与项目的主要工作时。这项工作包括使用流程图来确保正确组织系统过程。开发阶段标志着该系统开发初始部分的结束。此外，此阶段表示生产开始。开发阶段的特征还在于灌输和变革。在此阶段，专注于培训可以带来巨大的好处。

5. 集成与测试

第五阶段涉及系统集成和程序的系统测试，通常由质量保证（QA）专业人员执行，以确

定拟定的设计是否满足最初的业务目标。测试可以重复进行，尤其是检查错误、缺陷和互用性。测试可一直执行，直到最终用户接受为止。此阶段的另一部分是验证和确认，这两者将有助于确保程序的成功完成。

6. 实施

第六阶段是编写程序的大部分代码的时候。另外，此阶段涉及新开发系统的实际安装。此步骤将数据和组件从旧系统中移出并直接转换到新系统中，使项目投入生产。尽管这可能是一个冒险且很复杂的操作，但切换过程通常会在非高峰时段进行，从而将风险降到最低。系统分析人员和最终用户现在都应该看到已经实施更改的项目的实现。

7. 运营与维护

第七阶段也是最后阶段，涉及维护和定期更新。此步骤让最终用户可以根据需要微调系统以提高性能、添加新功能或满足其他用户要求。

如果企业确定在 SDLC 的任何阶段需要进行更改，则该企业可能不得不重复上述生命周期的各个阶段。任何项目的生命周期方法都是一个耗时的过程。即使某些步骤比其他步骤更困难，但也不应忽略任何步骤。疏忽可能会阻碍整个系统按计划运行。

Exercises

[Ex. 1] Answer the following questions according to Text A.

1. What are some examples of application software that allow you to do specific work?
2. What is Word processing software? What are the examples?
3. What does system software do? Why is it essential in managing the whole computer system?
4. What is programming software?
5. Why are drivers important?
6. What are some examples of driver software that you may be familiar with?
7. What is freeware software? What are some well-known examples of freeware?
8. What does shareware allow you to do? What are some examples of shareware that you must be familiar with?
9. What are common examples of open source software used by programmers?
10. What are some of the most common examples of closed source software?

[Ex.2] Fill in the following blanks according to Text B.

1. The system development life cycle, “SDLC” for short, is a _____, iterative

- process, structured in a _____ way. This process is used to model or provide a framework for technical and non-technical activities to deliver a _____ which meets or exceeds a business's expectations or manage _____.
2. Traditionally, the systems development life cycle consisted of _____ stages. That has now increased to _____ phases.
 3. The life cycle approach is used so users can see and understand what activities are involved _____. It is also used to let them know that _____, steps can be repeated or a previous step _____ when needing to _____.
 4. Planning is the first phase _____. It identifies whether or not there is the need for a new system to _____.
 5. There are several tools businesses can use that are specific to the second phase. They include:
_____, _____ and _____.
 6. The third phase describes, in detail, _____, _____ and _____ that will satisfy the functional requirements of the proposed system which will be in place.
 7. The development phase marks _____ of the process. Additionally, this phase signifies _____. The development stage is also characterized by _____.
 8. The fifth phase involves _____ and _____ — normally carried out by a quality assurance (QA) professional — to determine if the proposed design meets _____.
 9. The sixth phase is when the majority of the code for _____ is written. Additionally, this phase involves _____ of the newly-developed system.
 10. The life cycle approach of any project is _____. Even though some steps are _____ than others, none are to be overlooked. An oversight could prevent the entire system from _____.

[Ex. 3] Translate the following terms or phrases from English into Chinese and vice versa.

- | | |
|---------------------|----------|
| 1. assembly code | 1. _____ |
| 2. machine language | 2. _____ |
| 3. source code | 3. _____ |
| 4. flow chart | 4. _____ |
| 5. systems analysis | 5. _____ |
| 6. n. 抗病毒软件 | 6. _____ |

7. <i>n.</i> 汇编程序	7. _____
8. <i>n.</i> 调试器, 调试程序	8. _____
9. <i>n.</i> 解释器, 解释程序	9. _____
10. <i>n.</i> 缺陷, 瑕疵	10. _____

[Ex. 4] Translate the following sentences into Chinese.

Responsibilities and Tasks of Software Developers

1. Primary Responsibilities

Here's a non-exhaustive list of common tasks software developers are expected to complete.

- Creating and developing new software. Researching users' requirements, designing and writing new software, and testing newly designed software.
- Evaluating new and existing software systems. Designing testing plans for newly developed software, performing QA testing on software systems, finding faults in software systems, and correcting faults found in software systems.
- Improving existing software systems. Analyzing users' requirements and suggestions, creating solutions for existing issues, and implementing these solutions.
- Performing maintenance to existing systems by monitoring and correcting defects. Writing code (e.g. HTML, PHP, XML) for new software and updates. Running code to test efficiency, rewriting code to correct errors, and running tests again until code is error free.
- Writing operational manuals and systems specifications.
- Working in tandem with other staff members such as project managers, graphic designers, other developers, database administrators, and sales and marketing employees. Consulting with clients or project managers on the progress of developing software to check for possible improvements, suggestions, or requirements.
- Writing reports on project progress.

2. Daily Tasks

- Meeting with clients and project managers to design and develop new software.
- Establishing parameters and designing the architecture of new software.
- Designing, writing, reading, testing, and correcting code for new software.
- Running QA testing and searching for bugs in developing software.
- Reporting to clients and project managers on the development of new software.
- Testing and implementing software updates and improvements when necessary.
- Writing documentation for new and updated software.

[Ex. 5] Fill in the blanks with the words given below.

new interact data complex software
 programs designing project testing create

Software Developers



Software developers often work for computer firms and manufacturers. Their main role is to 1 the foundations for operative systems on which computer programmers work. They design, write, and test code for new systems and 2 to ensure efficiency. Software developers also run diagnostic 3 and quality assurance (QA) testing on existing projects before launching them to certify effectiveness.

A software developer is involved in all the process related to creating and 4 new systems; from initial planning, to establishing parameters, designing, writing, coding, encrypting, and 5 . This process is usually undertaken by a team of software developers, with each member carrying out a particular step of the process and a supervisor overseeing the entire 6 .

The work of a software developer may sometimes overlap with that of a database administrator. Many systems have to 7 in one way or another with data management systems, so it is the responsibility of the software developer to ensure that both systems are compatible. Some software developers can do this by themselves if they possess enough knowledge on 8 management systems and software.

Software developers often use several programming languages, their job is often very 9 and it involves advanced knowledge in computer science and mathematics. Their field is constantly evolving and 10 technologies and advancements are made every day, so they must be in a constant state of learning and self-improvement.

Online Resources

二维码	内 容
	计算机专业常用语法 (2): 状语从句
	在线阅读 (1): The Different Types of Software Testing (软件测试的不同类型)
	在线阅读 (2): Top 4 Software Development Methodologies (4 种常用的软件开发方法)