"Software" redirects here. For other uses, see Software (disambiguation).

Computer software, or just **software**, is a collection of computer programs and related data that provide the instructions for telling a computer what to do and how to do it. In other words, software is a conceptual entity which is a set of computer programs, procedures, and associated documentation concerned with the operation of a data processing system. We can also say software refers to one or more computer programs and data held in the storage of the computer for some purposes. In other words software is a set of **programs, procedures, algorithms** and its **documentation**. Program software performs the function of the program it implements, either by directly providing instructions to the computer hardware or by serving as input to another piece of software. The term was coined to contrast to the old term hardware (meaning physical devices). In contrast to hardware, software is intangible, meaning it "cannot be touched".[1] Software is also sometimes used in a more narrow sense, meaning application software only. Sometimes the term includes data that has not traditionally been associated with computers, such as film, tapes, and records.[2]

Examples of computer software include:

- Application software includes end-user applications of computers such as word processors or video games, and ERP software for groups of users.
- Middleware controls and co-ordinates distributed systems.
- Programming languages define the syntax and semantics of computer programs. For example, many mature banking applications were written in the COBOL language, originally invented in 1959. Newer applications are often written in more modern programming languages.
- System software includes operating systems, which govern computing resources. Today[*when?*] large[*quantify*] applications running on remote machines such as Websites are considered[*by whom?*] to be system software, because[*citation needed*] the end-user interface is generally through a graphical user interface, such as a web browser.
- Testware is software for testing hardware or a software package.
- Firmware is low-level software often stored on electrically programmable memory devices. Firmware is given its name because it is treated like hardware and run ("executed") by other software programs.
- Shrinkware is the older name given to consumer-purchased software, because it was often sold in retail stores in a shrink-wrapped box.
- Device drivers control parts of computers such as disk drives, printers, CD drives, or computer monitors.
- Programming tools help conduct computing tasks in any category listed above. For programmers, these could be tools for debugging or reverse engineering older legacy systems in order to check source code compatibility.

Contents

[hide]

History

For the history prior to 1946, see History of computing hardware.

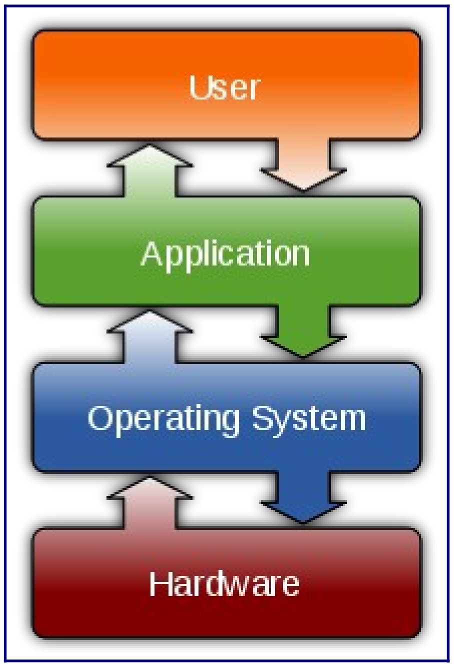
The first theory about software was proposed by Alan Turing in his 1935 essay *Computable numbers with an application to the Entscheidungsproblem (Decision problem)*.[3] The term "software" was first used in print by John W. Tukey in 1958.[4] Colloquially, the term is often used to mean application software. In computer science and software engineering, software is all information processed by computer system, programs and data.[4] The academic fields studying software are computer science and software engineering.

The history of computer software is most often traced back to the first software bug in 1946[*citation needed*]. As more and more programs enter the realm of firmware, and the hardware itself becomes smaller, cheaper and faster as predicted by Moore's law, elements of computing first considered to be software, join the ranks of hardware. Most hardware companies today have more software programmers on the payroll than hardware designers[*citation needed*], since software tools have automated many tasks of Printed circuit board engineers. Just like the Auto industry, the Software industry has grown from a few visionaries operating out of their garage with prototypes. Steve Jobs and Bill Gates were the Henry Ford and Louis Chevrolet of their times[*citation needed*], who capitalized on ideas already commonly known before they started in the business. In the case of Software development, this moment is generally agreed to be the publication in the 1980s of the specifications for the IBM Personal Computer published by IBM employee Philip Don Estridge. Today his move would be seen as a type of crowd-sourcing.

Until that time, software was *bundled* with the hardware by Original equipment manufacturers (OEMs) such as Data General, Digital Equipment and IBM[*citation needed*]. When a customer bought a minicomputer, at that time the smallest computer on the market, the computer did not come with Pre-installed software, but needed to be installed by engineers employed by the OEM. Computer hardware companies not only bundled their software, they also placed demands on the location of the hardware in a refrigerated space called a computer room. Most companies had their software on the books for 0 dollars, unable to claim it as an asset (this is similar to financing of popular music in those days). When Data General introduced the Data General Nova, a company called Digidyne wanted to use its RDOS operating system on its own hardware clone. Data General refused to license their software (which was hard to do, since it was on the books as a free asset), and claimed their "bundling rights". The Supreme Court set a precedent called Digidyne v. Data General in 1985. The Supreme Court let a 9th circuit decision stand, and Data General was eventually forced into licensing the Operating System software because it was ruled that restricting the license to only DG hardware was an illegal *tying arrangement*.[5] Soon after, IBM 'published' its DOS source for free, and Microsoft was born. Unable to sustain the loss from lawyer's fees, Data General ended up being taken over by EMC Corporation. The Supreme Court decision made it possible to value software, and also purchase Software patents. The move by IBM was almost a protest at the time. Few in the industry believed that anyone would profit from it other than IBM (through free publicity). Microsoft and Apple were able to thus cash in on 'soft' products. It is hard to imagine today that people once felt that software was worthless without a machine. There are many successful companies today that sell only software products, though there are still many common software licensing problems due to the complexity of designs and poor documentation, leading to patent trolls.

With open software specifications and the possibility of software licensing, new opportunities arose for software tools that then became the de facto standard, such as DOS for operating systems, but also various proprietary word processing and spreadsheet programs. In a similar growth pattern, proprietary development methods became standard Software development methodology.

Overview



A layer structure showing where operating system is located on generally used software systems on desktops

Software includes all the various forms and roles that digitally stored *data* may have and play in a computer (or similar system), regardless of whether the data is used as *code* for a CPU, or other interpreter, or whether it represents other kinds of information. Software thus encompasses a wide array of products that may be developed using different techniques such as ordinary programming languages, scripting languages, microcode, or an FPGA configuration.

The types of software include web pages developed in languages and frameworks like HTML, PHP, Perl, JSP, ASP.NET, XML, and desktop applications like OpenOffice.org, Microsoft Word developed in languages like C, C++, Objective-C, Java, C#, or Smalltalk. Application software usually runs on an underlying software operating systems such as Linux or Microsoft Windows. Software (or firmware) is also used in video games and for the configurable parts of the logic systems of automobiles, televisions, and other consumer electronics.

Computer software is so called to distinguish it from computer hardware, which encompasses the physical interconnections and devices required to store and execute (or run) the software. At the lowest level, executable code consists of machine language instructions specific to an individual processor. A machine language consists of groups of binary values signifying processor instructions that change the state of the computer from its preceding state. Programs are an ordered sequence of instructions for changing the state of the computer in a particular sequence. It is usually written in high-level programming languages that are easier and more efficient for humans to use (closer to natural language) than machine language. High-level languages are compiled or interpreted into machine language object code. Software may also be written in an assembly language, essentially, a mnemonic representation of a machine language using a natural language alphabet. Assembly language must be assembled into object code via an assembler.

Types of software

This section **does not cite any references or sources**. Please help improve this section by adding citations to reliable sources. Unsourced material may be challenged and removed. *(August 2010)*

Practical computer systems divide software systems into three major classes[*citation needed*]: system software, programming software and application software, although the distinction is arbitrary, and often blurred.

System software

System software provides the basic functions for computer usage and helps run the computer hardware and system. It includes a combination of the following:

- Device drivers
- Operating systems
- Servers
- Utilities
- Window systems

System software is responsible for managing a variety of independent hardware components, so that they can work together harmoniously. Its purpose is to unburden the application software programmer from the often complex details of the particular computer being used, including such accessories as communications devices, printers, device readers, displays and keyboards, and also to partition the computer's resources such as memory and processor time in a safe and stable manner.

Programming software

Programming software usually provides tools to assist a programmer in writing computer programs, and software using different programming languages in a more convenient way. The tools include:

- Compilers
- Debuggers
- Interpreters
- Linkers
- Text editors

An Integrated development environment (IDE) is a single application that attempts to manage all these functions..

Application software

Application software is developed to aid in any task that benefits from computation. It is a broad category, and encompasses software of many kinds, including the internet browser being used to display this page. This category includes:

- Business software
- Computer-aided design
- Databases
- Decision making software
- Educational software

- Image editing
- Industrial automation
- Mathematical software
- Medical software
- Molecular modeling software
- Quantum chemistry and solid state physics software
- Simulation software
- Spreadsheets
- Telecommunications (i.e., the Internet and everything that flows on it)
- Video editing software
- Video games
- Word processing

Software topics

Architecture

See also: Software architecture

Users often see things differently than programmers. People who use modern general purpose computers (as opposed to embedded systems, analog computers and supercomputers) usually see three layers of software performing a variety of tasks: platform, application, and user software.

- Platform software: Platform includes the firmware, device drivers, an operating system, and typically a graphical user interface which, in total, allow a user to interact with the computer and its peripherals (associated equipment). Platform software often comes bundled with the computer. On a PC you will usually have the ability to change the platform software.
- Application software: Application software or Applications are what most people think of when they think of software. Typical examples include office suites and video games. Application software is often purchased separately from computer hardware. Sometimes applications are bundled with the computer, but that does not change the fact that they run as independent applications. Applications are usually independent programs from the operating system, though they are often tailored for specific platforms. Most users think of compilers, databases, and other "system software" as applications.
- User-written software: End-user development tailors systems to meet users' specific needs. User software include spreadsheet templates and word processor templates. Even email filters are a kind of user software. Users create this software themselves and often overlook how important it is. Depending on how competently the user-written software has been integrated into default application packages, many users may not be aware of the distinction between the original packages, and what has been added by co-workers.

Documentation

Main article: Software documentation

Most software has software documentation so that the end user can understand the program, what it does, and how to use it. Without clear documentation, software can be hard to use—especially if it is very specialized and relatively complex like Photoshop or AutoCAD.

Developer documentation may also exist, either with the code as comments and/or as separate files, detailing how the programs works and can be modified.

Library

Main article: Software library

An executable is almost always not sufficiently complete for direct execution. Software libraries include collections of functions and functionality that may be embedded in other applications. Operating systems include many standard Software libraries, and applications are often distributed with their own libraries.

Standard

Main article: Software standard

Since software can be designed using many different programming languages and in many different operating systems and operating environments, software standard is needed so that different software can understand and exchange information between each other. For instance, an email sent from a Microsoft Outlook should be readable from Yahoo! Mail and vice versa.

Execution

Main article: Execution (computing)

Computer software has to be "loaded" into the computer's storage (such as the hard drive or memory). Once the software has loaded, the computer is able to *execute* the software. This involves passing instructions from the application software, through the system software, to the hardware which ultimately receives the instruction as machine code. Each instruction causes the computer to carry out an operation – moving data, carrying out a computation, or altering the control flow of instructions.

Data movement is typically from one place in memory to another. Sometimes it involves moving data between memory and registers which enable high-speed data access in the CPU. Moving data, especially large amounts of it, can be costly. So, this is sometimes avoided by using "pointers" to data instead. Computations include simple operations such as incrementing the value of a variable data element. More complex computations may involve many operations and data elements together.

Quality and reliability

Main articles: Software quality, Software testing, and Software reliability

Software quality is very important, especially for commercial and system software like Microsoft Office, Microsoft Windows and Linux. If software is faulty (buggy), it can delete a person's work, crash the computer and do other unexpected things. Faults and errors are called "bugs." Many bugs are discovered and eliminated (debugged) through software testing. However, software testing rarely – if ever – eliminates every bug; some programmers say that "every program has at least one more bug" (Lubarsky's Law). All major software companies, such as Microsoft, Novell and Sun Microsystems, have their own software testing departments with the specific goal of just testing. Software can be tested through unit testing, regression testing and other methods, which are done manually, or most commonly, automatically, since the amount of code to be tested can be quite large. For instance, NASA has extremely rigorous software testing procedures for many operating systems and communication functions. Many NASA based operations interact and identify each other through command programs called software. This enables many people who work at NASA to check and evaluate functional systems overall. Programs containing command software enable hardware engineering and system operations to function much easier together.

License

Main article: Software license

The software's license gives the user the right to use the software in the licensed environment. Some software comes with the license when purchased off the shelf, or an OEM license when bundled with hardware. Other software comes with a free software license, granting the recipient the rights to modify and redistribute the software. Software can also be in the form of freeware or shareware.

Patents

Main articles: Software patent and Software patent debate

Software can be patented in some but not all countries; however, software patents can be controversial in the software industry with many people holding different views about it. The controversy over software patents is about specific algorithms or techniques that the software contains, which may not be duplicated by others and considered intellectual property and copyright infringement depending on the severity.

Design and implementation

Main articles: Software development, Computer programming, and Software engineering

Design and implementation of software varies depending on the complexity of the software. For instance, design and creation of Microsoft Word software will take much more time than designing and developing Microsoft Notepad because of the difference in functionalities in each one.

Software is usually designed and created (coded/written/programmed) in integrated development environments (IDE) like Eclipse, Emacs and Microsoft Visual Studio that can simplify the process and compile the program. As noted in different section, software is usually created on top of existing software and the application programming interface (API) that the underlying software provides like GTK+, JavaBeans or Swing. Libraries (APIs) are categorized for different purposes. For instance, JavaBeans library is used for designing enterprise applications, Windows Forms library is used for designing graphical user interface (GUI) applications like Microsoft Word, and Windows Communication Foundation is used for designing web services. Underlying computer programming concepts like quicksort, hashtable, array, and binary tree can be useful to creating software. When a program is designed, it relies on the API. For instance, if a user is designing a Microsoft Windows desktop application, he/she might use the .NET Windows Forms library to design the desktop application and call its APIs like *Form1.Close()* and *Form1.Show()*[6] to close or open the application and write the additional operations him/herself that it need to have. Without these APIs, the programmer needs to write these APIs him/herself. Companies like Sun Microsystems, Novell, and Microsoft provide their own APIs so that many applications are written using their software libraries that usually have numerous APIs in them.

Computer software has special economic characteristics that make its design, creation, and distribution different from most other economic goods.[7][8] A person who creates software is called a programmer, software engineer, software developer, or code monkey, terms that all have a similar meaning.

Industry and organizations

Main article: Software industry

A great variety of software companies and programmers in the world comprise a software industry. Software can be quite a profitable industry: Bill Gates, the founder of Microsoft was the richest person in the world in 2009 largely by selling the Microsoft Windows and Microsoft Office software products. The same goes for Larry Ellison, largely through his Oracle database software. Through time the software industry has become increasingly specialized.

Non-profit software organizations include the Free Software Foundation, GNU Project and Mozilla Foundation. Software standard organizations like the W3C, IETF develop software standards so that most software can interoperate through standards such as XML, HTML, HTTP or FTP.

Other well-known large software companies include Novell, SAP, Symantec, Adobe Systems, and Corel, while small companies often provide innovation.

See also

- List of software
- hardware

References

1. ^ "Wordreference.com: WordNet 2.0". Princeton University, Princeton, NJ. Retrieved 2007-08-19.
2. ^ software..(n.d.). *Dictionary.com Unabridged (v 1.1)*. Retrieved 2007-04-13, from Dictionary.com website: http://dictionary.reference.com/browse/software
3. ^ Hally, Mike (2005:79). *Electronic brains/Stories from the dawn of the computer age*. British Broadcasting Corporation and Granta Books, London. ISBN 1-86207-663-4.
4. ^ ***a*** ***b*** John Tukey, 85, Statistician; Coined the Word 'Software', New York Times, Obituaries, July 28, 2000 [1]
5. ^ Tying Arrangements and the Computer Industry: Digidyne Corp. vs. Data General
6. ^ "MSDN Library". Retrieved 2010-06-14.
7. ^ v. Engelhardt, Sebastian (2008): "The Economic Properties of Software", Jena Economic Research Papers, Volume 2 (2008), Number 2008-045. (in Adobe pdf format)
8. ^ "Why Open Source Is The Optimum Economic Paradigm for Software" by Dan Kaminsky 1999