{{other uses}}
File:Operating system placement (software).svg|thumb|upright|A diagram showing how the User (computing)|user interacts with application software on a typical desktop computer.The application software layer interfaces with the operating system, which in turn communicates with the Personal computer hardware|hardware. The arrows indicate information flow.

'''Computer software''', or simply '''software''', is that part of a computer system that consists of data (computing)|data or computer instructions, in contrast to the Computer hardware|physical hardware from which the system is built. In computer science and software engineering, computer software is all information processed by computer systems, Computer program|programs and data. Computer software includes computer programs, Library (computing)|libraries and related non-executable Data (computing)|data, such as Software documentation|online documentation or digital media. Computer hardware and software require each other and neither can be realistically used on its own.

At the lowest level, executable code consists of Machine code|machine language instructions specific to an individual Microprocessor|processorâ€”typically a central processing unit (CPU). A machine language consists of groups of Binary numbers|binary values signifying processor instructions that change the state of the computer from its preceding state. For example, an instruction may change the value stored in a particular storage location in the computerâ€”an effect that is not directly observable to the user. An instruction may also (indirectly) cause something to appear on a display of the computer systemâ€”a state change which should be visible to the user. The processor carries out the instructions in the order they are provided, unless it is instructed to branch instruction|"jump" to a different instruction, or is interrupted (by now multi-core processors are dominant, where each core can run instructions in order; then, however, each application software runs only on one core by default, but some software has been made to run on many).

The majority of software is written in high-level programming languages that are easier and more efficient for programmers, meaning closer to a natural language.{{cite web|title=Compiler construction|url=http://www.cs.uu.nl/education/vak.php?vak=INFOMCCO}} High-level languages are translated into machine language using a compiler or an Interpreter (computing)|interpreter or a combination of the two. Software may also be written in a low-level assembly language, essentially, a vaguely mnemonic representation of a machine language using a natural language alphabet, which is translated into machine language using an Assembly language|assembler.

History
{{Main article|History of software}}
An outline (algorithm) for what would have been the first piece of software was written by Ada Lovelace in the 19th century, for the planned Analytical Engine. However, neither the Analytical Engine nor any software for it were ever created.

The first theory about softwareâ€”prior to creation of computers as we know them todayâ€”was proposed by Alan Turing in his 1935 essay ''Computable numbers with an application to the Entscheidungsproblem'' (decision problem).

This eventually led to the creation of the twin academic fields of computer science and software engineering, which both study software and its creation. Computer science is more theoretical (Turing's essay is an example of computer science), where as software engineering focuses on more practical concerns.

However, prior to 1946, software as we now understand itâ€”programs stored in the memory of stored-program digital computersâ€”did not yet exist. The first electronic computing devices were instead rewired in order to "reprogram" them.

Types of software
{{See also|List of software categories}}

On virtually all computer platforms, software can be grouped into a few broad categories.

=Purpose, or domain of use=
Based on the goal, computer software can be divided into:

 ''Application software'', which is software that uses the computer system to perform special functions or provide video game|entertainment functions beyond the basic operation of the computer itself. There are many different types of application software, because the range of tasks that can be performed with a modern computer is so largeâ€”see list of software.
 ''System software'', which is software that directly operates the computer hardware, to provide basic functionality needed by users and other software, and to provide a platform for running application software.{{cite web|title=System Software|url=http://home.olemiss.edu/~misbook/sfsysfm.htm|archive-url=https://web.archive.org/web/20010530092843/http://home.olemiss.edu:80/~misbook/sfsysfm.htm|dead-url=yes|archive-date=2001-05-30|publisher=The University of Mississippi}} System software includes:
 ''Operating systems'', which are essential collections of software that manage resources and provides common services for other software that runs "on top" of them. Supervisory programs, boot loaders, shell (computing)|shells and window systems are core parts of operating systems. In practice, an operating system comes bundled with additional software (including application software) so that a user can potentially do some work with a computer that only has an operating system.
 ''Device drivers'', which operate or control a particular type of device that is attached to a computer. Each device needs at least one corresponding device driver; because a computer typically has at minimum at least one input device and at least one output device, a computer typically needs more than one device driver.
 ''Software utility|Utilities'', which are computer programs designed to assist users in the maintenance and care of their computers.

''Malicious software'' or ''malware'', which is software that is developed to harm and disrupt computers. As such, malware is undesirable. Malware is closely associated with computer-related crimes, though some malicious programs may have been designed as practical jokes.

=Nature or domain of execution=
Desktop applications such as web browsers and Microsoft Office, as well as smartphone and Tablet computer|tablet applications (called "mobile app|apps"). (There is a push in some parts of the software industry to merge desktop applications with mobile apps, to some extent. Windows 8, and later Ubuntu Touch, tried to allow the same style of application user interface to be used on desktops, laptops and mobiles.)
JavaScript scripts are pieces of software traditionally embedded in web pages that are run directly inside the web browser when a web page is loaded without the need for a web browser plugin. Software written in other programming languages can also be run within the web browser if the software is either translated into JavaScript, or if a web browser plugin that supports that language is installed; the most common example of the latter is ActionScript scripts, which are supported by the Adobe Flash plugin.
Server software, including:
Web applications, which usually run on the web server and output dynamically generated web pages to web browsers, using e.g. PHP, Java (programming language)|Java, ASP.NET, or even Node.js|JavaScript that runs on the server. In modern times these commonly include some JavaScript to be run in the web browser as well, in which case they typically run partly on the server, partly in the web browser.
Plug-in (computing)|Plugins and extensions are software that extends or modifies the functionality of another piece of software, and require that software be used in order to function;
Embedded software resides as firmware within embedded systems, devices dedicated to a single use or a few uses such as cars and televisions (although some embedded devices such as wireless chipsets can ''themselves'' be part of an ordinary, non-embedded computer system such as a PC or smartphone).{{cite web|title=Embedded Softwareâ€”Technologies and Trends|url=http://www.computer.org/csdl/mags/so/2009/03/mso2009030014.html|publisher=IEEE Computer Society|date=Mayâ€“June 2009|accessdate=6 November 2013}} In the embedded system context there is sometimes no clear distinction between the system software and the application software. However, some embedded systems run embedded operating systems, and these systems do retain the distinction between system software and application software (although typically there will only be one, fixed, application which is always run).
Microcode is a special, relatively obscure type of embedded software which tells the processor ''itself'' how to execute machine code, so it is actually a lower level than machine code. It is typically proprietary to the processor manufacturer, and any necessary correctional microcode software updates are supplied by them to users (which is much cheaper than shipping replacement processor hardware). Thus an ordinary programmer would not expect to ever have to deal with it.

=Programming tools=
{{Main article|Programming tool}}

Programming tools are also software in the form of programs or applications that software developers (also known as ''programmers, coders, hackers'' or ''software engineers'') use to create, Debugging|debug, Software maintenance|maintain (i.e. improve or fix), or otherwise Technical support|support software. Software is written in one or more programming languages; there are many programming languages in existence, and each has at least one implementation, each of which consists of its own set of programming tools. These tools may be relatively self-contained programs such as compilers, debuggers, interpreter (computing)|interpreters, linker (computing)|linkers, and text editors, that can be combined together to accomplish a task; or they may form an integrated development environment (IDE), which combines much or all of the functionality of such self-contained tools. IDEs may do this by either invoking the relevant individual tools or by re-implementing their functionality in a new way. An IDE can make it easier to do specific tasks, such as searching in files in a particular project. Many programming language implementations provide the option of using both individual tools or an IDE.

Software topics

=Architecture=
{{See also|Software architecture}}
Users often see things differently from 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: The Platform (computing)|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 Personal computer|PC one 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.

=Execution=
{{Main article|Execution (computing)}}

Computer software has to be "loaded" into the computer storage|computer's storage (such as the hard drive or Computer memory|memory). Once the software has loaded, the computer is able to ''execute'' the software. This involves passing instruction (computer science)|instructions from the application software, through the system software, to the hardware which ultimately receives the instruction as machine language|machine code. Each instruction causes the computer to carry out an operationâ€”moving data (computing)|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.

<!-- This section is simply to long for this article and needs to be compressed into the intro above, or moved to the article itself.

Instructions may be performed sequentially, conditionally, or iteratively. Sequential instructions are those operations that are performed one after another. Conditional instructions are performed such that different sets of instructions execute depending on the value(s) of some data. In some languages this is known as an "if" statement. Iterative instructions are performed repetitively and may depend on some data value. This is sometimes called a "loop." Often, one instruction may "call" another set of instructions that are defined in some other program or module (programming)|module. When more than one computer processor is used, instructions may be executed simultaneously.

A simple example of the way software operates is what happens when a user selects an entry such as "Copy" from a menu. In this case, a conditional instruction is executed to copy text from data in a 'document' area residing in memory, perhaps to an intermediate storage area known as a 'clipboard' data area. If a different menu entry such as "Paste" is chosen, the software may execute the instructions to copy the text from the clipboard data area to a specific location in the same or another document in memory.

Depending on the application, even the example above could become complicated. The field of software engineering endeavors to manage the complexity of how software operates. This is especially true for software that operates in the context of a large or powerful computer system.

Currently, almost the only limitations on the use of computer software in applications is the ingenuity of the designer/programmer. Consequently, large areas of activities (such as playing grand master level chess) formerly assumed to be incapable of software simulation are now routinely programmed. The only area that has so far proved reasonably secure from software simulation is the realm of human artâ€” especially, pleasing music and literature.{{Citation needed|date=June 2007}}

Kinds of software by operation: computer program as executable, source code or script (computer programming)|script, computer configuration|configuration.-->

=Quality and reliability=
{{Main article|Software quality|Software testing|Software reliability}}
Software quality is very important, especially for commercial software|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 "Software bug|bugs" which are often discovered during alpha and beta testing. Software is often also a victim to what is known as software aging, the progressive performance degradation resulting from a combination of unseen bugs.

Many bugs are discovered and eliminated (debugged) through software testing. However, software testing rarelyâ€”if everâ€”eliminates every bug; some programmers say that "every program has at least one more bug" (Lubarsky's Law).<ref name="github">{{cite web| url=https://github.com/mark-watson/scripting-intelligence-book-examples/blob/master/part1/wikipedia_text/software.txt | title=scripting intelligence book examples }} In the Waterfall model|waterfall method of software development, separate testing teams are typically employed, but in newer approaches, collectively termed agile software development, developers often do all their own testing, and demonstrate the software to users/clients regularly to obtain feedback. 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. 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, and in the case of free software licenses, also grants other rights such as the right to make copies.

Proprietary software can be divided into two types:

 freeware, which includes the category of "free trial" software or "freemium" software (in the past, the term shareware was often used for free trial/freemium software). As the name suggests, freeware can be used for free, although in the case of free trials or freemium software, this is sometimes only true for a limited period of time or with limited functionality.
 software available for a fee, often inaccurately termed "commercial software", which can only be legally used on purchase of a license.

Open source software, on the other hand, comes with a free software license, granting the recipient the rights to modify and redistribute the software.

=Patents=
{{Main article|Software patent|Software patent debate}}
Software patents, like other types of patents, are theoretically supposed to give an inventor an exclusive, time-limited license for a ''detailed idea (e.g. an algorithm) on how to implement'' a piece of software, or a component of a piece of software. Ideas for useful things that software could ''do'', and user ''requirements'', are not supposed to be patentable, and concrete implementations (i.e. the actual software packages implementing the patent) are not supposed to be patentable eitherâ€”the latter are already covered by copyright, generally automatically. So software patents are supposed to cover the middle area, between requirements and concrete implementation. In some countries, a requirement for the claimed invention to have an effect on the physical world may also be part of the requirements for a software patent to be held validâ€”although since ''all'' useful software has effects on the physical world, this requirement may be open to debate.

Software patents are controversial in the software industry with many people holding different views about them. One of the sources of controversy is that the aforementioned split between initial ideas and patent does not seem to be honored in practice by patent lawyersâ€”for example the patent for Aspect-Oriented Programming (AOP), which purported to claim rights over ''any'' programming tool implementing the idea of AOP, howsoever implemented. Another source of controversy is the effect on innovation, with many distinguished experts and companies arguing that software is such a fast-moving field that software patents merely create vast additional litigation costs and risks, and actually retard innovation. In the case of debates about software patents outside the United States, the argument has been made that large American corporations and patent lawyers are likely to be the primary beneficiaries of allowing or continue to allow software patents.

Design and implementation
{{Main article|Software development|Computer programming|Software engineering}}
Design and implementation of software varies depending on the complexity of the software. For instance, the design and creation of Microsoft Word took much more time than designing and developing Microsoft Notepad because the latter has much more basic functionality.

Software is usually designed and created (aka coded/written/programmed) in integrated development environments (IDE) like Eclipse (software)|Eclipse, IntelliJ IDEA|IntelliJ and Microsoft Visual Studio that can simplify the process and compiler|compile the software (if applicable). As noted in a 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 (Java)|Swing. Libraries (APIs) can be categorized by their purpose. For instance, the Spring framework|Spring Framework is used for implementing enterprise applications, the 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. When a program is designed, it relies upon the API. For instance, if a user is designing a Microsoft Windows desktop application, he or she might use the .NET Framework|.NET Windows Forms library to design the desktop application and call its APIs like ''Form1.Close()'' and ''Form1.Show()''{{cite web |url=http://msdn.microsoft.com/en-us/library/default.aspx | title=MSDN Library|accessdate=2010-06-14}} to close or open the application, and write the additional operations him/herself that it needs to have. Without these APIs, the programmer needs to write these APIs him/herself. Companies like Oracle Corporation|Oracle and Microsoft provide their own APIs so that many applications are written using their Library (computing)|software libraries that usually have numerous APIs in them.

Data structures such as hash tables, array data type|arrays, and binary trees, and algorithms such as quicksort, can be useful for creating software.

Computer software has special economic characteristics that make its design, creation, and distribution different from most other economic goods.{{Specify|Which characteristics?|date=May 2012}}{{cite journal|author=v. Engelhardt, Sebastian |year=2008|url=https://ideas.repec.org/p/jrp/jrpwrp/2008-045.html |title=The Economic Properties of Software|journal= Jena Economic Research Papers| volume= 2| issue= 2008â€“045.}}{{cite web|url=http://dankaminsky.com/1999/03/02/69/ |title=Why Open Source Is The Optimum Economic Paradigm for Software|first= Dan |last=Kaminsky |year=1999}}

A person who creates software is called a programmer, software engineer or software developer, terms that all have a similar meaning. More informal terms for programmer also exist such as "coder" and "Hacker (expert)|'''hacker'''"{{Spaced ndash}}although use of the latter word may cause confusion, because it is more often used to mean Hacker (computer security)|someone who illegally breaks into computer systems.

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 co-founder of Microsoft was the richest person in the world in 2009, largely due to his ownership of a significant number of shares in Microsoft, the company responsible for Microsoft Windows and Microsoft Office software products.

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

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

```
See also
 Software release life cycle
 Independent software vendor
 List of software
 Software asset management
{{portal bar|Software|Free software|Information technology}}

References
{{reflist}}

External links
{{Sister project links | wikt=software | commons=Special:Search/Software |
b= | n= | s= | v=Computer Software | voy= |q=no}}
 {{dmoz|Computers/Software}}<!--ref name="github"/-->

{{Software digital distribution platforms|state=collapsed}}
{{Authority control}}

Category:Computing
Category:Computer science
Category:Software|
```