CMP 1103: Information and Communication Technology

Lecture 05: Computer Software

Computer Software is the collection of computer programs and related data that provide the instructions telling a computer what to do. It is the stored, machine readable code that instructs a computer to carry out specific tasks**.**

Relationship to Computer Hardware
Computer software is intangible in contrast to computer hardware, which encompasses the physical interconnections and devices required to store and execute (or run) the software. Hardware starts functioning once software is loaded onto it.

In computers, software is loaded into RAM and executed in the central processing unit. At the lowest level, software consists of a machine language specific to an individual processor. A machine language consists of groups of binary values signifying processor instructions (object code), which change the state of the computer from its preceding state.
Software is thus an ordered sequence of instructions for changing the state of the computer hardware 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 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.

Software Classes/Types
Practical computer systems divide software into three major classes: system software, programming software and application software, although the distinction is arbitrary, and often blurred.

- **System Software** helps run the computer hardware and computer system. It includes operating systems, device drivers, diagnostic tools, servers, windowing systems, utilities and more.
  The purpose of systems software is to insulate the applications programmer as much as possible from the details of the particular computer complex being used, especially memory and other hardware features, and such accessory devices as communications, printers, readers, displays, keyboards, etc.
- **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 text editors, compilers, interpreters, linkers, debuggers, and so on. An Integrated Development Environment (IDE) merges those tools into a software bundle, and a programmer may not need to type multiple commands for compiling, interpreter, debugging, tracing, and etc., because the IDE usually has an advanced *graphical user interface,* or GUI.
- **Application Software** allows end users to accomplish one or more specific (non-computer related) tasks. Typical applications include industrial automation, business software, educational software, medical software, databases, and computer games. Businesses are probably the biggest users of application software, but almost every field of human activity now uses some form of application software. It is used to automate all sorts of functions.

The 3 Layers of Software
Users often see things differently than programmers. People who use modern general purpose computers (as opposed to embedded systems, analog computers, supercomputers, etc.) usually see three layers of software performing a variety of tasks: platform, application, and user software.

CMP 1103: Information and Communication Technology

1. Platform Software

Platform includes the basic input-output system (often described as *firmware* rather than *software*), 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, and users may not realize that it exists or that they have a choice to use different platform software.

2. 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 almost always 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.

3. User-Written Software

User software tailors systems to meet the users' specific needs. User software includes spreadsheet templates, word processor macros, scientific simulations, graphics and animation scripts. 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 purchased application packages, many users may not be aware of the distinction between the purchased packages, and what has been added by fellow co-workers.

How Software is Input into Computer

There are various ways of getting software into a computer. Some of the general ones include:

1. Built into the computer's circuits, the ROM chips.
2. Loaded into the computer from a secondary storage device, like a floppy disk or hard disk drive or CD-ROMs.
3. Typed in from the keyboard.
    - Usually need to use a programming language to create the software.
    - Rarely done by most computer users today.

Operation

Once software is 'loaded' into the computer's storage such as a *hard drive*, *memory*, or *RAM*, the computer is able to execute it. Computers operate by *executing* the computer program. 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.

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. 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 are 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 art— especially, pleasing music and literature.

Kinds of software by operation: computer program as executable, source code or script, configuration.

Program and library

A **Program** may not be sufficiently complete for execution by a computer. In particular, it may require additional software from a **software library** in order to be complete. A **Library** is a collection of subroutines or classes used to develop software. Libraries contain code and data that provide services to independent programs. *E.g. Microsoft Windows will check the registry to determine the proper place to find an ActiveX DLL (Dynamic Link Library), but for other DLLs it will check the directory where it loaded the program from; the current working directory; any directories set by calling the SetDllDirectory() function; the System32, System, and Windows directories; and finally the directories specified by the PATH environment variable.*

Such a library may include software components used by stand-alone programs, but which cannot work on their own. Thus, programs may include standard routines that are common to many programs, extracted from these libraries. Libraries may also include 'stand-alone' programs which are activated by some computer event and/or perform some function (e.g., of computer 'housekeeping') but do not return data to their calling program. Programs may be called by one to many other programs; programs may call zero to many other programs.

License

Software 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 OEM license when bundled with hardware. Software can also be in the form of freeware or shareware.

Patents

The definition for patents is debatable and the issue on the whole is controversial. A patent is a set of exclusionary rights granted by a state to a patent holder for a limited period of time, usually 20 years. Some believe that they hinder software development, while others argue that software patents provide an important incentive to spur software innovation.

CMP 1103: Information and Communication Technology

Operating system

An Operating System (OS) is a set of computer programs that manage the hardware and software resources of a computer. An operating system rationally processes electronic devices in response to approved commands. At the foundation of all system software, an operating system performs basic tasks such as controlling and allocating memory, prioritizing system requests, controlling input and output devices, facilitating networking, and managing files. It also may provide a graphical user interface (GUI) for higher level functions. It forms a platform for other system software and for application software.

OS Services

a) Process Management

Every program running on a computer, be it background services or applications, is a process.

As long as a von Neumann architecture is used to build computers, only one process per CPU can be run at a time. Older microcomputer OS such as MS-DOS did not attempt to bypass this limit, with the exception of interrupt processing, and only one process could be run under them (although DOS itself featured TSR as a very partial and not too easy to use solution). Mainframe operating systems have had multitasking capabilities since the early 1960s. Modern operating systems enable concurrent execution of many processes at once via multitasking even with one CPU. Process management is an operating system's way of dealing with running multiple processes. Since most computers contain one processor with one core, multitasking is done by simply switching processes quickly. Depending on the operating system, as more processes run, either each time slice will become smaller or there will be a longer delay before each process is given a chance to run. Process management involves computing and distributing CPU time as well as other resources. Most operating systems allow a process to be assigned a priority which affects its allocation of CPU time. Interactive operating systems also employ some level of feedback in which the task with which the user is working receives higher priority. Interrupt driven processes will normally run at a very high priority. In many systems there is a background process, such as the System Idle Process in Windows, which will run when no other process is waiting for the CPU.

b) Memory Management

Current computer architectures arrange the computer's memory in a hierarchical manner, starting from the fastest registers, CPU cache, random access memory and disk storage. An operating system's memory manager coordinates the use of these various types of memory by tracking which one is available, which is to be allocated or de-allocated and how to move data between them. This activity, usually referred to as virtual memory management, increases the amount of memory available for each process by making the disk storage seem like main memory. There is a speed penalty associated with using disks or other slower storage as memory – if running processes require significantly more RAM than is available, the system may start thrashing. This can happen either because one process requires a large amount of RAM or because two or more processes compete for a larger amount of memory than is available. This then leads to constant transfer of each process's data to slower storage.

Another important part of memory management is managing virtual addresses. If multiple processes are in memory at once, they must be prevented from interfering with each other's memory (unless there is an explicit request to utilize shared memory). This is achieved by having separate address spaces. Each process sees the whole virtual address space, typically from address 0 up to the maximum size of virtual memory, as uniquely assigned to it. The operating system maintains a page table that match virtual addresses to physical addresses. These memory allocations are tracked so that when a process terminates, all memory used by that process can be made available for other processes.

The operating system can also write inactive memory pages to secondary storage. This process is called "paging" or "swapping" – the terminology varies between operating systems.

It is also typical for operating systems to employ otherwise unused physical memory as a page cache; requests for data from a slower device can be retained in memory to improve performance. The operating system can also pre-load the in-memory cache with data that may be requested by the user in the near future; SuperFetch is an example of this.

c) Disk and File Systems

All operating systems include support for a variety of file systems.
Modern file systems are comprised of a hierarchy of directories. While the idea is conceptually similar across all general-purpose file systems, some differences in implementation exist. Two noticeable examples of this are the character used to separate directories, and case sensitivity. By default, Microsoft Windows demarcates its path components with a backslash (Japanese editions of Windows use ¥, and Korean editions use ₩) and its file names are not case sensitive whereas Unix/Linux-derived operating systems, as well as Amiga OS and Mac OS X, use the forward slash and their file names generally are case sensitive. Versions of Mac OS prior to OS X use a colon for a path separator. RISC OS uses a period.

File systems are journaled or non-journaled. A journaled file system is a safer alternative under the circumstances of a system crash. If a system comes to an abrupt stop in a system crash scenario, the non-journaled system will need to undergo an examination from the system check utilities, whereas the journaled file systems recovery is automatic.

Many Linux distributions support some or all of ext2, ext3, ReiserFS, Reiser5, GFS, GFS2, OCFS, OCFS2, NILFS. Linux also has full support for XFS and JFS, along with the FAT file systems, and NTFS.

Microsoft Windows includes support for FAT12, FAT16, FAT32, and NTFS. The NTFS file system is the most efficient and reliable of the four Windows systems, and as of Windows Vista, is the only file system which the operating system can be installed on. Windows Embedded CE 6.0 introduced ExFAT, a file system suitable for flash drives.

Mac OS X supports HFS+ as its primary file system, and it supports several other file systems as well.

Common to all these (and other) operating systems is support for file systems typically found on removable media. FAT12 is the file system most commonly found on floppy discs. ISO 9660 and Universal Disk Format are two common formats that target Compact Discs and DVDs, respectively. Mount Rainier is a newer extension to UDF supported by Linux 2.6 kernels and Windows Vista that facilitates rewriting to DVDs in the same fashion as what has been possible with floppy disks.

d) Networking

Most current operating systems are capable of using the TCP/IP networking protocols. This means that one system can appear on a network of the other and share resources such as files, printers, and scanners using either wired or wireless connections.

Many operating systems also support one or more vendor-specific legacy networking protocols as well, for example, SNA on IBM systems, DECnet on systems from Digital Equipment Corporation, and Microsoft-specific protocols on Windows. Specific protocols for specific tasks may also be supported such as NFS for file access.

e) Security

Many operating systems include some level of security. Security is based on the two ideas that:

- The operating system provides access to a number of resources, directly or indirectly, such as files on a local disk, privileged system calls, personal information about users, and the services offered by the programs running on the system;
- The operating system is capable of distinguishing between some requesters of these resources who are authorized (allowed) to access the resource, and others who are not authorized (forbidden). While some systems may simply distinguish between "privileged" and "non-privileged", systems commonly have a form of requester *identity*, such as a user name. Requesters, in turn, divide into two categories:

    - ✓ Internal security: an already running program. On some systems, a program once it is running has no limitations, but commonly the program has an identity which it keeps and is used to check all of its requests for resources.
    - ✓ External security: a new request from outside the computer, such as a login at a connected console or some kind of network connection. To establish identity there may be a process of *authentication*. Often a username must be quoted, and each username may have a password. Other methods of authentication, such as magnetic cards or biometric data might be used instead. In some cases, especially connections from the network, resources may be accessed with no authentication at all.

In addition, to the allow/disallow model of security, a system with a high level of security will also offer auditing options. These would allow tracking of requests for access to resources (such as, "who has been reading this file?").

Security of operating systems has long been a concern because of highly sensitive data held on computers, both of a commercial and military nature. The United States Government Department of Defense (DoD) created the *Trusted Computer System Evaluation Criteria* (TCSEC), which is a standard that sets basic requirements for assessing the effectiveness of security. This became of vital importance to operating system makers, because the TCSEC was used to evaluate, classify and select computer systems being considered for the processing, storage and retrieval of sensitive or classified information.

Internal security

Internal security can be thought of as protecting the computer's resources from the programs concurrently running on the system. Most operating systems set programs running natively on the computer's processor, so the problem arises of how to stop these programs doing the same task and having the same privileges as the operating system (which is after all just a program too). Processors used for general purpose operating systems generally have a hardware concept of privilege. Generally less privileged programs are automatically blocked from using certain hardware instructions, such as those to read or write from external devices like disks. Instead, they have to ask the privileged program (operating system kernel) to read or write. The operating system therefore gets the chance to check the program's identity and allow or refuse the request.

An alternative strategy, and the only sandbox strategy available in systems that do not meet the Popek and Goldberg virtualization requirements, is the operating system not running user programs as native code, but instead either emulates a processor or provides a host for a p-Code based system such as Java.

Internal security is especially relevant for multi-user systems; it allows each user of the system to have private files that the other users cannot tamper with or read. Internal security is also vital if auditing is to be of any use, since a program can potentially bypass the operating system, inclusive of bypassing auditing.

External security

Typically an operating system offers (hosts) various services to other network computers and users. These services are usually provided through ports or numbered access points beyond the operating systems network address. Typically services include offerings such as file sharing, print services, email, web sites, and file transfer protocols.

At the front line of security are hardware devices known as firewalls. At the operating system level, there are a number of software firewalls available. Most modern operating systems include a software firewall, which is enabled by default. A software firewall can be configured to allow or deny network traffic to or from a service or application running on the operating system. Therefore, one can install and be running an insecure service, such as Telnet or FTP, and not have to be threatened by a security breach because the firewall would deny all traffic trying to connect to the service on that port.

f) Graphical User Interfaces

Today, most modern operating systems contain Graphical User Interfaces (GUIs, pronounced *goo-eez*). A few older operating systems tightly integrated the GUI to the kernel—for example, the original implementations of Microsoft Windows and Mac OS. More modern operating systems are modular, separating the graphics subsystem from the kernel (as is now done in Linux, and Mac OS X, and to a limited extent Windows).

Many operating systems allow the user to install or create any user interface they desire. The X Window System in conjunction with GNOME or KDE is a commonly found setup on most Unix and Unix derivative (BSD, Linux, Minix) systems.

Graphical user interfaces tend to change and evolve over time. For example, Windows has modified its user interface almost every time a new major version of Windows is released, and the Mac OS GUI changed dramatically with the introduction of Mac OS X in 2001.

g) Device Drivers

A device driver is a specific type of computer software developed to allow interaction with hardware devices. Typically this constitutes an interface for communicating with the device, through the specific computer bus or communications subsystem that the hardware is connected to, providing commands to and/or receiving data from the device, and on the other end, the requisite interfaces to the operating system and software applications. It is a specialized hardware-dependent computer program which is also operating system specific that enables another program, typically an operating system or applications software package or computer program running under the operating system kernel, to interact transparently with a hardware device, and usually provides the requisite interrupt handling necessary for any necessary asynchronous time-dependent hardware interfacing needs.

The key design goal of device drivers is abstraction. Every model of hardware (even within the same class of device) is different. Newer models also are released by manufacturers that provide more reliable or better performance and these newer models are often controlled differently. Computers and their operating systems cannot be expected to know how to control every device, both now and in the future. To solve this problem, OSes essentially dictate how every type of device should be controlled. The function of the device driver is then to translate these OS mandated function calls into device specific calls. In theory a new device, which is controlled in a new manner, should function correctly if a suitable driver is available. This new driver will ensure that the device appears to operate as usual from the operating systems' point of view for any person.

Personal Computers

- IBM PC compatible - Microsoft Windows, Unix variants, and Linux variants.
- Apple Macintosh - Mac OS X (a Unix variant), Windows (x86 variants only), Linux and BSD

Mainframe Computers

The earliest operating systems were developed for mainframe computer architectures in the 1960s. The enormous investment in software for these systems caused most of the original computer manufacturers to continue to develop hardware and operating systems that are compatible with those early operating systems. Those early systems pioneered many of the features of modern operating systems. Mainframe operating systems that are still supported include:

- Burroughs MCP-- B5000, 1961 to Unisys Clearpath/MCP, present.
- IBM OS/360 -- IBM System/360, 1965 to IBM zSeries, present
- UNIVAC EXEC 8 -- UNIVAC 1108, 1965, to Unisys Clearpath IX, present.

Modern mainframes typically also run Linux or Unix variants. A "Datacenter" variant of Windows Server 2003 is also available for some mainframe systems.

Embedded Systems

Embedded systems use a variety of dedicated operating systems. In some cases, the "operating system" software is directly linked to the application to produce a monolithic special-purpose program. In the simplest embedded systems, there is no distinction between the OS and the application. Embedded systems that have certain time requirements are known as Real-time operating systems.

Microsoft Windows

A Vista desktop launched for the first time.

The *Microsoft Windows* family of operating systems originated as a graphical layer on top of the older MS-DOS environment for the IBM PC. Modern versions are based on the newer Windows NT core that first took shape in OS/2 and borrowed from VMS. Windows runs on 32-bit and 65-bit Intel and AMD processors, although earlier versions also ran on the DEC Alpha, MIPS, Fairchild (later Intergraph) Clipper and PowerPC architectures (some work was done to port it to the SPARC architecture).

As of 2006, Windows held a near-monopoly of around 95% of the worldwide desktop market share, although some predict this to dwindle due to the increased interest in open source operating systems.[1] It is also used on low-end and mid-range servers, supporting applications such as web servers and database servers. In recent years, Microsoft has spent significant marketing and R&D money to demonstrate that Windows is capable of running any enterprise application which has resulted in consistent price/performance records (see the TPC) and significant acceptance in the enterprise market at the cost of existing Unix based system market share.

The most widely used version of the Microsoft Windows family is Microsoft Windows XP, released on October 25, 2001. The latest release of Windows XP is Windows XP Service Pack 2, released on August 6, 2005.

In November 2006, after more than five years of development work, Microsoft released Windows Vista, a major new version of Microsoft Windows which contains a large number of new features and architectural changes. Chief amongst these are a new user interface and visual style called Windows Aero, a number of new security features such as User Account Control, and new multimedia applications such as Windows DVD Maker.

Others

Mainframe operating systems, such as IBM's z/OS, and embedded operating systems such as VxWorks, eCos, and Palm OS, are usually unrelated to Unix and Windows, except for Windows CE, Windows NT Embedded 5.0 and Windows XP Embedded which are descendants of Windows, and several *BSDs, and Linux distributions tailored for embedded systems. OpenVMS from Hewlett-Packard (formerly DEC), is still under active development.

Older operating systems which are still used in niche markets include OS/2 from IBM; Mac OS, the non-Unix precursor to Apple's Mac OS X; BeOS; XTS-300.

Popular prior to the Dot COM era, operating systems such as AmigaOS and RISC OS continue to be developed as minority platforms for enthusiast communities and specialist applications.

Research and development of new operating systems continues. GNU Hurd is designed to be backwards compatible with Unix, but with enhanced functionality and a microkernel architecture. Singularity is a project at Microsoft Research to develop an operating system with better memory protection based on the .Net managed code model.

Application Software

Application Software is a subclass of computer software that employs the capabilities of a computer directly to a task that the user wishes to perform. This should be contrasted with system software which is involved in integrating a computer's various capabilities, but typically does not directly apply them in the performance of tasks that benefit the user. In this context the term application refers to both the *application software* and its implementation.

A simple, if imperfect, analogy in the world of hardware would be the relationship of an electric light—an application—to an electric power generation plant—the system. The power plant merely generates electricity, itself not really of any use until harnessed to an application like the electric light which performs a service that the user desires.

The exact delineation between the operating system and application software is not precise, however, and is occasionally subject to controversy. For example, one of the key questions in the United States v. Microsoft antitrust trial was whether Microsoft's Internet Explorer web browser was part of its Windows operating system or a separable piece of application software. As another example, the GNU/Linux naming controversy is, in part, due to disagreement about the relationship between the Linux kernel and the Linux operating system.

Typical examples of **software applications** are word processors, spreadsheets, and media players.

Multiple applications bundled together as a package are sometimes referred to as an **application suite**. Microsoft Office and OpenOffice.org, which bundle together a word processor, a spreadsheet, and several other discrete applications, are typical examples. The separate applications in a suite usually have a user interface

that has some commonality making it easier for the user to learn and use each application. And often they may have some capability to interact with each other in ways beneficial to the user. For example, a spreadsheet might be able to be embedded in a word processor document even though it had been created in the separate spreadsheet application.

User-written software tailors systems to meet the user's specific needs. User-written software include spreadsheet templates, word processor macros, scientific simulations, graphics and animation scripts. Even email filters are a kind of user software. Users create this software themselves and often overlook how important it is.

In some types of embedded systems, the application software and the operating system software may be indistinguishable to the user, as in the case of software used to control a VCR, DVD player or Microwave Oven.

Application Software Classification

There are many subtypes of Application Software:

- Enterprise software addresses the needs of organization processes and data flow, often in a large distributed ecosystem. (Examples include Financial, Customer Relationship Management, and Supply Chain Management). Note that Departmental Software is a sub-type of Enterprise Software with a focus on smaller organizations or groups within a large organization. (Examples include Travel Expense Management, and IT Helpdesk)
- Enterprise infrastructure software provides common capabilities needed to create Enterprise Software systems. (Examples include Databases, Email servers, and Network and Security Management)
- Information worker software addresses the needs of individuals to **create** and manage information, often for individual projects within a department, in contrast to enterprise management. Examples include time management, resource management, documentation tools, analytical, and collaborative. Word processors, spreadsheets, email and blog clients, personal information system, and individual media editors may aid in multiple information worker tasks.
- Media and entertainment software addresses the needs of individuals and groups to **consume** digital entertainment and **published** digital content. (Examples include Media Players, Web Browsers, Help browsers, and Games)
- Educational software is related to Media and Entertainment Software, but has distinct requirements for delivering evaluations (tests) and tracking progress through material. It is also related to collaboration software in that many Educational Software systems include collaborative capabilities.
- Media development software addresses the needs of individuals who generate print and electronic media for others to consume, most often in a commercial or educational setting. This includes Graphic Art software, Desktop Publishing software, Multimedia Development software, HTML editors, Digital Animation editors, Digital Audio and Video composition, and many others.
- Product engineering software is used in developing hardware and software products. This includes computer aided design (CAD), computer aided engineering (CAE), computer language editing and compiling tools, Integrated Development Environments, and Application Programmer Interfaces.

Enterprise Software

- Accounting software
- Back office
- Business software
- Human Resource Management

Enterprise Infrastructure Software

- Business workflow software

- Database management system (DBMS) software
- Digital asset management (DAM) software
- Document Management software

Information Worker Software

- Time and Resource Management
    - Task and Scheduling
- Data Management
    - Contact Management
    - Spreadsheet
    - Personal Database
- Documentation
    - Word Processing
    - Desktop publishing software
    - Diagramming Software
    - Presentation software
- Analytical software
    - Baudline
    - DADiSP
    - JMP
    - Maple
    - MathCAD
    - Mathematica
    - MATLAB
    - Maxima
    - Minitab
    - Computer algebra systems
    - Statistical packages
    - Numerical computing
    - Neural network software
- Collaborative software
    - Blog
    - Wiki
    - E-mail
    - Wrike

Media and Entertainment Software

- Digitally Published Media
    - Web browser
    - Media players
    - Hybrid editor players

- Entertainment software
    - Arcade
    - Computer and Video Games
    - Console Emulations
    - Handheld Game Consoles Games
    - Video Game Console Games
    - Wireless or Mobile Phone Games

CMP 1103: Information and Communication Technology

Educational Software

- Classroom Management
- Survey Management
- Training Management
- Sales Readiness Software

Media Development Software

- Graphic art software
- Image organizer
- Media Editing
    - Image editing software
    - Video editing software
    - Sound editing software
    - Music sequencer
    - Hypermedia editing software
        - Web Development Software
    - Media Data Formats
        - Raster graphics
        - Vector graphics
        - 3D graphics
        - Animation
        - Video
        - Digital audio
        - MIDI
    - Media File Formats
        - Graphic file formats
        - Video file formats
        - Audio file formats

Product Engineering Software

- Hardware Engineering
    - Computer-Aided Engineering
    - Computer-Aided Design (CAD)
- Software Engineering
    - Computer Languages
    - Compiler Software
    - Integrated Development Environments

Software License

A **Software license** comprises the permissions, rights and restrictions imposed on software (whether a component or a free-standing program). Use of software without a license could constitute infringement of the owner's exclusive rights under copyright or, occasionally, patent law and allow the owner to sue the infringer.

Under a software license, the licensee is permitted to use the licensed software in compliance with the specific terms of the license. If there is a breach of the license, depending on the license it may result in termination of the license, and potentially the right of the owner to sue.

A software vendor may offer a software license unilaterally (without giving the licensee the opportunity to negotiate for more favorable terms) such as in a shrink wrap contract, or even as part of a software license agreement with another party. Virtually all mass produced proprietary software is sold under some form or fashion of software license agreement. One off, or custom software is often licensed under terms of which are specifically negotiated between the licensee and licensor.

In addition to granting rights and imposing restrictions on use of the software, software licenses typically contain provisions which allocate liability and responsibility between the parties. In enterprise and commercial software transactions these terms (such as limitations of liability, warranties and warranty disclaimers, and indemnity if the software infringes intellectual property rights of others) are often negotiated by attorneys specialized in software licensing. The legal field has seen the growth of this specialized practice area due to unique legal issues with software licenses, and the desire of software companies to protect assets which, if licensed improperly, could diminish their value.

Open-source software

Open-source software is an antonym for closed source software and refers to any computer software whose source code is available under a license (or arrangement such as the public domain) that permits users to study, change, and improve the software, and to redistribute it in modified or unmodified form. It is often developed in a public, collaborative manner. It is the most prominent example of open source development and often compared to user generated content.

Users should be treated as co-developers

> The users are treated like co-developers and so they should have access to the source code of the software. Furthermore users are encouraged to submit additions to the software, code fixes for the software, bug reports, documentation etc. Having more co-developers increases the rate at which the software evolves. Linus's law states that, "Given enough eyeballs all bugs are shallow." This means that if many users view the source code they will eventually find all bugs and suggest how to fix them. Note that some users have advanced programming skills, and furthermore, each user's machine provides an additional testing environment. This new testing environment offers that ability to find and fix a new bug.

Early Releases

> The first version of the software should be released as early as possible so as to increase one's chances of finding co-developers early.

Frequent Integration

> New code should be integrated as often as possible so as to avoid the overhead of fixing a large number of bugs at the end of the project life cycle. Some Open Source projects have nightly builds where integration is done automatically on a daily basis.

Several Versions

There should be at least two versions of the software. There should be a buggier version with more features and a more stable version with fewer features. The buggy version (also called the development version) is for users who want the immediate use of the latest features, and are willing to accept the risk of using code that is not yet thoroughly tested. The users can then act as co-developers, reporting bugs and providing bug fixes. The stable version offers the users fewer bugs and fewer features.

High Modularization

The general structure of the software should be modular allowing for parallel development.

Dynamic Decision Making Structure

There is a need for a decision making structure, whether formal or informal, that makes strategic decisions depending on changing user requirements and other factors.

Freeware

Freeware is copyrighted computer software which is made available for use free of charge, for an unlimited time, as opposed to shareware where the user is required to pay (e.g. after some trial period). Authors of freeware often want to "give something to the community", but also want credit for their software and to retain control of its future development. Sometimes when programmers decide to stop developing a freeware product, they will give the source code to another programmer or release the product's source code to the public as free software.

The only criterion for being classified as "freeware" is that the software must be made available for use for an unlimited time at no cost. The software license may impose one or more other restrictions on the type of use including personal use, individual use, non-profit use, non-commercial use, academic use, commercial use or any combination of these. For instance, the license may be "free for personal, non-commercial use." Everything created with the freeware programs can be distributed at no cost (for example graphic, documents, or sounds made by user).

There is some software that may be considered freeware, but that have limited distribution; that is, they may only be downloaded from a specific site, and they can not be redistributed. Hence, these software wouldn't be freely redistributable software. According to the basic definition, that software would be freeware; according to stricter definitions, they wouldn't be.

Freeware contrasts with free software, because of the different meanings of the word "free". Freeware is gratis and refers to *zero price*, versus free software that is described as "libre", which means *free to study, change, copy, redistribute, share and use the software in any purpose*. However, many programs are both freeware and free software. They are available for zero price, provide the source code and are distributed with free software permissions. This software would exclusively be called free software to avoid confusion with freeware that usually does not come with the source code and is therefore proprietary software.

Variations

There are many variations of freeware. Freeware is an umbrella term which can include loss leaders (in the form of crippleware), public domain software, free software, proprietary software, and shareware when there is no price to be paid to use the software.

Shareware Implementations

Free/open source software and shareware are similar in that they can be obtained and used without monetary cost. Usually shareware differs from free/open source software in that requests of voluntary shareware fees are made, often within the program itself, and in that source code for shareware programs is generally not available in a form that would allow others to extend the program. Notwithstanding that tradition, some

free/open source software authors ask for voluntary donations, although there is no requirement to do so. Free/open source software is usually compatible with the strict ASP shareware guidelines.

Sometimes, paying the fee and obtaining a password results in access to expanded features, documentation, or support. In some cases, unpaid use of the software is limited in time or in features — in which case the software is vernacularly called crippleware. Some shareware items require no payment; just an email address, so that the supplier can use this address for their own purposes.

Shareware is available on all major computer platforms including Microsoft Windows, Macintosh, and Linux. Titles cover a very wide range of categories including: business, software development, education, home, multimedia, design, drivers, games, and utilities.

There is a technical difference between shareware and demos. Up to the early 1990s, shareware could easily be upgraded to the full version by adding the other episodes or full portion of the game; this would leave the existing shareware files intact. Demos are different in that they are self-contained programs which are *not* upgradable to the full version. A good example is the *Descent* shareware versus the *Descent II* demo; players were able to retain their saved games on the former but not the latter.

Logistics of shareware

With shareware, a developer bypasses the normal distribution channel eliminating the normal retail middleman markups and directly markets to the end user. The end result is a reduced end-user price compared to the retail channel. Users of shareware are encouraged to copy and distribute unregistered versions of the software to friends, coworkers and other acquaintances. The hope is that users will find the program useful or entertaining and will pay to register to be able to access all the features.

Pertaining more towards shareware games, large online distribution channels known as "portals", such as Yahoo! Games and RealArcade, have emerged in recent years. These portals act as media of distribution for the shareware developers, providing an audience base for a percentage of the software's sale.

Shareware developers are usually individual computer programmers brave enough to take initiative and take risk — entrepreneurs. Therefore, online shareware author communities, like the newsgroup alt.comp.shareware.authors, are places for software seekers to post their novel software ideas for potential implementation.

Shareware Distribution

In the early 1990s, shareware distribution was a popular method of publishing games for smaller developers, including then-fledgling companies such as Apogee Software (now also operating under the brand 3D Realms), Epic Megagames (now Epic Games), and id Software. It gave consumers the chance to try a portion of the game, usually restricted to the game's complete first section or episode, before purchasing the rest of the adventure. Racks of games on single 5 1/5 inch and later 3.5 inch floppy disks were common in retail stores. However, bulletin board systems (BBS) and computer expositions such as Software Creations BBS were the primary distributors of all early low-cost software. Free software from a BBS was the motive force for consumers to purchase a computer equipped with a modem, so as to acquire software at no cost. At PC expositions, extant today, shareware was essentially free; the cost only covered the disk and minimal packaging.

As the increasing size of games in the mid-1990s made them impractical to fit on floppies, and retail publishers and developers began to earnestly mimic the practice, shareware games were replaced by shorter demos that were either distributed free on CDs with gaming magazines or as free downloads over the Internet, in some cases becoming exclusive content for specific websites.

CMP 1103: Information and Communication Technology

Kinds of Software - Summary

1. Public Domain Software
    - Has no copyright - no one owns the right to control who can make copies of the software.
    - Free to use or make copies of.
    - Can be copied, used in other programs, or changed by anyone.

2. Freeware
    - Has a copyright - someone owns the right to determine who can make copies of the software.
    - Free to use and make copies of.
    - Can only give away exact copies of the software.
    - Can not be changed or used in another program without the copyright holder's permission.

3. Shareware
    - Has a copyright.
    - Allowed to use the software before paying for it.
        - Can be a demo - which limits some major features like the Save command.
        - Can set an amount of time you can use the software.
        - Can trust that you will pay for it if you like the software.
    - Can only give away exact copies of the software.
    - Can not be changed or used in another program without the copyright holder's permission.

4. Open source
    - source code is available under a license(or arrangement such as the public domain)
    - permits users to study, change, and improve the software, and to redistribute it in modified or unmodified form.
    - often developed in a public, collaborative manner.

5. Commercial Software
    - Has the most esistive copyright.
    - Have to buy the software before you can use it.
    - Can usually make one copy of the software as a backup copy.
        - A backup copy is used in case something goes wrong with the original software.
        - Can not give away or sell the backup copy.
    - Can not copy, look at the program's code, change, or use the software in another program without the copyright holder's permission.
    - Commercial Software is the best software in the world.