Technology Guide

2

Software

TG2.1 Software Fundamentals and Types

Computer hardware, networks, and mobile devices, such as PDSs and iPhones, cannot function or perform a single operation without instructions. These instructions are known as **software**—or **computer programs, applications, or plug-ins.** Software enables users to customize a computer or device.

SOFTWARE FUNDAMENTALS

Software consists of sequences of instructions that a computer, network, or device executes to perform a task. The process of writing programs is called **programming** or **coding,** and individuals who perform this task are called programmers. Stored software programs are accessed and their instructions are executed in the computer's CPU.

Computer programs include **documentation,** which is a written description of the functions of the program. Documentation helps the user operate the computer system or helps other programmers understand what the program does and how it accomplishes its purposes. Documentation is vital to the business organization. Without it, if a key programmer or user leaves (or forgets), the knowledge of how to use the program or how it is designed may be lost.

All software is written in a **programming language,** such as Visual Basic, PHP, COBOL, C+ +, or Java. For a list of the most popular programming languages, visit *langpop.com/*.

TYPES OF SOFTWARE

There are two major types of software: **application software** and **systems software.** Both operating system software and applications have specific jobs to perform.

Application software is a set of computer instructions, written in a programming language. The instructions direct the computer hardware to perform specific data- or information-processing activities that provide functionality to the user. This functionality may be broad, such as word processing, or narrow, such as an organization's payroll program.

Systems software acts primarily as an intermediary between computer hardware and application programs, and users may also manipulate it directly. Systems software such as the Windows operating system (OS) provides important self-regulatory functions for computer systems, such as loading (**booting**) itself when the computer is first turned on.

Application programs primarily manipulate data or text to produce or provide information. Systems programs primarily manipulate computer hardware resources. The systems software available on a computer provides the capabilities and limitations within which the application software can operate. Figure TG2.1 shows that systems software is a necessary intermediary between hardware and application software; the application software cannot run without the systems software.

Both application software and systems software are written in programming languages, which are also presented in this guide.

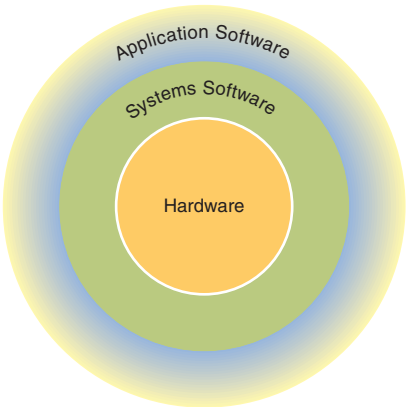

Figure TG2.1 Systems software serves as an intermediary between hardware and functional applications.

TG2.2 Application Software

Application software is programs that perform specific tasks. You have used many different types of applications. Several common types of applications are the following:

- **Spreadsheet applications** for creating documents to manage and organize numerical data
- **Word-processing applications** for creating documents that are formatted and organized for readability
- **Database applications** for developing databases that can organize and retrieve large amounts of information
- **Accounting applications** for managing personal checkbooks or the accounting functions of businesses.
- **Activity management applications** such as calendars and address books
- **Presentation applications** for making slide shows
- **Graphics applications** for creating pictures
- **Communications programs** such as e-mail, text messaging, and fax software for sending and receiving messages
- **Multimedia applications** for creating video and music
- **Utilities or utility programs** for performing a variety of tasks that maintain or enhance the computer's operating system

Table TG 2.1 shows the most popular productivity software suites for the three different types of software: proprietary suite, open source, and Web based.

Some of these general-purpose tools are actually development tools. That is, you use them to construct applications. For example, you can use Excel to build decision support applications such as resource allocation, scheduling, or inventory control. You can use these and similar packages for doing statistical analysis, for conducting financial analysis, and for supporting marketing research.

Off-the-shelf application software can be purchased, leased, or rented from a vendor that develops programs and sells them to many organizations. Off-the-shelf software may be a standard package or it may be customizable. Special-purpose programs or "packages" can be tailored for a specific purpose, such as inventory

TABLE TG2.1 Popular Productivity Software Suites: Proprietary Suite, Open Source, and Web Based

	Proprietary Microsoft Office	Open Source Open Office	Web-Based Google Docs
Price	Has a range of different prices, starting at $59.95 for the student-only version	*Openoffice.org* free; Oracle Open Office, starts at $49.95 per license	Free for personal use; Google Apps for Business, $50 per user account per year
Supported operating systems	Windows; there is separate version for Apple's OS X	Windows, OS X, Linux, Solaris, and several other operating systems	Will run on any operating system that has a supported Web browser, including IE, Firefox, Chrome, or Safari
Online and offline	Use primarily offline; 2010 version now offers online apps	Offline only	Google gears make it possible to use Google Docs offline
Word processor	Word	Writer	Documents
Spreadsheet	Excel	Calc	Spreadsheets
Presentation	PowerPoint	Impress	Presentations
Database	Access	Base	Does not have database software
Other software	OneNote, Outlook, Publisher, Groove, Communicator, InfoPath	Math, Draw	Drawing, Forms

control or payroll. The word *package* is a commonly used term for a computer program (or group of programs) that has been developed by a vendor and is available for purchase in a prepackaged form.

Many decision support and business applications are built with programming languages rather than with general-purpose application programs. This is especially true for complex, unstructured problems. Information systems applications can also be built with a mix of general-purpose programs and/or with a large number of development tools ranging from editors to random number generators. Of special interest are the software suites, for example, Microsoft Office, Open Office, or Google Docs. These are integrated sets of tools that can expedite application development. Also of special interest are CASE tools and integrated enterprise software, which are described later in the guide.

Other Application Software. There exist hundreds of other application software products. Of special interest to business managers are the following.

1. Middleware. Internet applications designed to let one company interact with other companies are complex because of the variety of hardware and software with which they must be able to work. This complexity increased when mobile wireless devices began to access company networks via the Internet. Middleware is software designed to link application modules developed in different computer languages and running on heterogeneous platforms, whether on a single machine or over a network. Middleware keeps track of the locations of the software modules that need to link to each other across a distributed system and manages the actual exchange of information.

2. Enterprise applications. Enterprise software consists of programs that manage the vital operations of an organization, such as supply chain management (movement of raw materials from suppliers through shipment of finished goods to customers), inventory replenishment, ordering, logistics coordination, human resources management, manufacturing, operations, accounting, and financial management. Some common modules of enterprise applications software are payroll, sales order processing, accounts payable/receivable, and tax accounting.

Enterprise software vendors are producing software that is less expensive, based on industry standards, compatible with other vendors' products, and easier to configure and install. The largest vendors—SAP, Oracle, PeopleSoft, and Computer Associates—offer software programs that make the jobs of business users and IT personnel easier.

3. Presence software. Presence technology can detect when you're online and what kind of device you're using. It has its roots in instant messaging (IM). When you log on to an IM service, your arrival is immediately announced to a list of other users you've selected to be alerted to your online presence.

TG2.3 Systems Software

Systems software controls and supports the computer hardware and its information-processing activities. Systems software also facilitates the programming, testing, and debugging of computer programs. It is more general than applications software and is usually independent of any specific type of application. Systems software programs support application software by directing the basic functions of the computer. For example, when the computer is turned on, the initialization program (a systems program) prepares and readies all devices for processing. Other common operating systems tasks are the following:

- Monitoring performance
- Correcting errors

- Providing and maintaining the user interface
- Starting ("booting") the computer
- Reading programs into memory
- Managing memory allocation to those programs
- Placing files and programs in secondary storage
- Creating and maintaining directories
- Formatting diskettes
- Controlling the computer monitor
- Sending jobs to the printer
- Maintaining security and limiting access
- Locating files
- Detecting viruses
- Compressing data

Systems software can be grouped into three major functional categories:

1. System control programs are programs that control the use of the hardware, software, and data resources of a computer system during its execution of a user's information-processing job. An operating system is the prime example of a system control program.

2. System support programs support the operations, management, and users of a computer system by providing a variety of services. System utility programs, performance monitors, and security monitors are examples of system support programs.

3. System development programs help users develop information-processing programs and procedures and prepare user applications. Major development programs are language compilers, interpreters, and translators.

SYSTEM CONTROL PROGRAMS

The most important system control programs are described below.

Operating Systems. The main component of systems software is a set of programs collectively known as the *operating system*. The OS supervises the overall operation of the computer, including monitoring the computer's status, handling executable program interruptions, and scheduling operations, which include controlling input and output processes.

Computers with only one CPU appear to perform multiple tasks simultaneously. In such cases, the OS controls which particular tasks have access to the various resources of the computer. At the same time, the OS controls the overall flow of information within the computer.

On a microcomputer, the OS controls the computer's communication with its display, printer, and storage devices. It also receives and directs inputs from the keyboard and other data input sources. The OS is designed to maximize the amount of useful work the hardware of the computer system accomplishes.

Programs running on the computer use various resources controlled by the OS. These resources include CPU time, primary storage or memory, and input/output devices. The operating system attempts to allocate the use of these resources in the most efficient manner possible.

The OS also provides an interface between the user and the hardware. By masking many of the hardware features, both the professional and the end-user programmers are presented with a system that is easier to use.

Portability, a desirable characteristic of OS, means that the same OS software can be run on different computers. An example of a portable operating system is UNIX and Unix-like operating systems. Versions of these operating systems can run on hardware produced by a number of different vendors. Examples include Linux, NetBSD, IBM's AIX, and Oracle's Solaris. However, there is no one standard version of UNIX that will run on all machines.

Operating System Functions. The operating system performs four major functions in the operation of a computer system: job management, resource management, server consolidation, and data management.

1. Job management is the preparing, scheduling, and monitoring of jobs for continuous processing by the computer system. A **job control language (JCL)** is a special computer language found in the mainframe-computing environment that allows a programmer to communicate with the operating system.

2. Resource management is controlling the use of computer system resources employed by the other systems software and application software programs being executed on the computer. These resources include primary storage, secondary storage, CPU processing time, and input/output devices.

3. Server consolidation is all about creating a simpler, more rational, and more manageable infrastructure. There are four possible consolidation strategies: logical consolidation, physical consolidation, workload consolidation, and application consolidation. Consolidation also leads to much more flexible, consistent, and efficient use of resources than distributed servers by allowing customers to strike the right balance within each server.

4. Data management is the controlling of the input and output of data as well as its location, storage, and retrieval. Data management programs control the allocation of secondary storage devices, the physical format, and cataloging of data storage.

Desktop and Notebook Computer OSs. Microsoft Windows is the leading series of desktop OSs. The MS-DOS (Microsoft Disk Operating System) was one of the original operating systems for the IBM PC and its clones.

Windows 95, released in 1995, was the first of a series of products in the Windows operating system that provided a streamlined GUI by using icons to provide instant access to common tasks. Subsequent products in the Microsoft Windows OS are the following:

- Windows 98 was not a major upgrade to Windows 95 but did offer minor refinements, bug fixes, and enhancements to Windows 95.
- Windows NT is an operating system for high-end desktops, workstations, and servers. Windows NT supports software written for DOS and Windows, and it provides extensive computing power for new applications with large memory and file requirements. It is also designed for easy and reliable connection with networks and other computing machinery; it is popular in networked systems.
- Windows 2000 is a renamed version of Windows NT 5.0. This OS had added security features and ran on multiple-processor computers.
- Windows XP was the first upgrade to Windows 2000 and had three versions: a 32-bit consumer version, a 32-bit business version, and a 64-bit business version.
- Windows Vista was the next major release of the Windows OS. Windows Vista was supposed to deliver major improvements in user productivity, important new capabilities for software developers, and significant advances in security, deployment, and reliability.
- Windows 7 is an improvement in performance, stability, and security over Windows Vista. Windows Touch optimizes Windows 7 to be used with a touch screen. This greatly improves the user experience when using Windows on a touch-screen device like a table computer. Windows 7 Professional and Ultimate both include Windows XP Mode, which runs a virtual installation of Windows XP that allows users to run software that is compatible only with Windows XP. New interface functionalities include Snap, which resizes the window depending on where it is moved along the edge of the screen.

UNIX is another operating system that provides sophisticated desktop features, including multiprocessing and multitasking. UNIX is valuable to business organizations

because it can be used on many different platforms, can support many different hardware devices (e.g., printers, plotters, etc.), and has numerous applications written to run on it. UNIX has many different versions. Most UNIX vendors are focusing their development efforts on servers rather than on desktops, and a few, like IBM, promote Linux for use on the desktop. However, the most popular version of UNIX on personal computers is Apple's OS X.

Linux was originally written by Linus Torvalds at the University of Helsinki in Finland in 1991. He then released the source code to the world (called *open-source* software). Since that time, many programmers around the world have worked on Linux and written software for it. The result is that, like UNIX, Linux now runs on multiple hardware platforms, can support many different hardware devices, and has numerous applications written to run on it.

Linux (sometimes called GNU/Linux) is a UNIX-like operating system that was designed to provide personal computer users with a free or very-low-cost operating system comparable to traditional, more expensive UNIX systems. Linux is a remarkably complete operating system, including a graphical user interface, an X Window system, multitasking, virtual memory management, TCP/IP, the Emacs editor, and other components usually found in a comprehensive UNIX system.

Even though Linux was designed for personal computers, it has seen most of its success on servers. In fact, the majority of both Web servers and supercomputers run Linux. Linux is used on personal computers, but it only has at most 3 percent of the personal computer operating system market. Some statistics claim Linux only has a little over 1 percent of the market. As a result, the majority of companies developing Linux, such as Red Hat, focus on the server market. However, Canonical does make an easy-to-use desktop Linux operating system called Ubuntu (*ubuntu.com*).

The Macintosh operating system, Mac OS X, for Apple Macintosh microcomputers, currently is a 64-bit UNIX OS that runs on Intel X86_64 CPUs. OS X legally can only run on an "Apple-branded computer" according to the terms of the OS X end-user software license agreement (EULA). OS X supports Internet integration, virtual memory management, and AppleTalk networking. Mac OS X features a user interface (named Aqua), advanced graphics, virtual memory management, and multitasking.

Netbook OSs. Currently most netbooks come with Windows 7 Starter, which is the most basic version of Windows 7. However, there are still a few that ship with Windows XP or with Linux-based OSs. Another option that may be available in the future is Google's Chromium OS, often called Chrome OS. It is an open-source Linux-based netbook operating system. While the source code has been released under the BSD license, it is still a work in progress. No major manufacturers have shipped netbooks running Chromium OS. The most notable feature of Chromium OS is that it only runs the Chrome Web browser, so it is only useable when connected to the Internet.

Smartphones have their own OSs, as described in Table TG 2.2.

Mainframe Operating Systems. Mainframe computers usually require specialized OSs that can handle a large load and that have advanced security features. The major server operating systems include Linux, Windows Server 2008, UNIX, HP NonStop, and IBM's z/OS. Although some of these are also desktop operating systems, all can serve as departmental server operating systems because of their strong scalability, reliability, backup, security, fault tolerance, multitasking, multiprocessing, TCP/IP networking (Internet integration), network management, and directory services.

Enterprise Server Operating Systems. Enterprise server operating systems (e.g., Red Hat's RHES, Novell's SLES, IBM's z/OS, VM, VSE, and IBM's I) generally run on mainframes and midrange systems. Enterprise operating systems offer superior manageability, security, and stability as well as support for online applications, secure

TABLE TG2.2 Smartphone Operating Systems

OS	Symbian	BlackBerry OS	Windows Phone 7	iOS	Android	WebOS
Manufacturer	Nokia with the Symbian foundation	RIM (Research In Motion)	Microsoft	Apple	Google with the open handset Alliance	Palm (now owned by HP)
Comments	Most popular phone OS in the world	Most popular smartphone OS in the United States	Windows Phone 7 is a complete overhaul of Microsoft's Smartphone Operating System	Changed the way people in United States looked at cell phones when it was first sold on 6/29/07	Open source under the Apache License 2.0, while still being strongly controlled by Google	After several bad decisions by Palm, HP bought Palm primarily for WebOS
Who can use the OS on their devices?	Symbian v3 is open source, so anyone can use it; however, prior versions were proprietary	Only RIM	Licensed to hardware manufacturers by Microsoft	Only Apple	Open source but strongly controlled by Google	Both open source and propriety components;. Palm had talked about licensing WebOS before it was bought by HP
Official application store	There are several different stores; A few examples are Nokia's Ovi store, Samsung Applications Store, and AT&T Media Mall	App World was created to compete with the iPhone's app store; apps can still be acquired from other sources like they were before App World	Mobile Marketplace	Has the largest and fastest-growing app store, with over 200,000 apps; all apps must be approved by Apple	Android Market is the second-largest and fastest-growing application store, with over 38,000 apps	Palm App Catalog
First phone	Ericsson R380	BlackBerry 6710 and 6720	Not yet announced	Original iPhone	G1	Palm Pre
Capable of full multitasking, running multiple apps at once	Yes	Yes	Unlike Windows Mobile, Windows Phone 7 doesn't have multitasking for third-party apps; Instead, apps are paused when they are switched	Beginning with 4.0, iOS will have full multitasking	Yes	Yes
Also used on	Symbian will likely be used on tablet computers	Was originally used on two-way pagers	Currently no devices running Windows Phone 7 are available to the public	iPod touch and modified version on iPad	Netbooks, tablets, e-book readers, portable media player	HP plans to port WebOS to tablets

electronic commerce, multiple concurrent users, large (terabyte) databases, and millions of transactions per day. Enterprise server operating systems also offer partitioning, a method of segmenting a server's resources to allow the processing of multiple applications on a single system.

Supercomputer Operating Systems. Supercomputer operating systems target the supercomputer hardware market. Most of these systems are based on Linux, such as CNK/SLES 9 and RedHat Enterprise Linux 4. However, supercomputer OSs are not limited to Linux. Some examples of non-Linux systems include the Cray Xs1's Unicos, HP-UX and HP's 2000 K/S/X, and IBM's AIX (both types of UNIX). Other manufacturers are Sun, NEC, Silicon Graphics, and Fujitsu. These OSs manage highly parallel multiprocessor and multiuser environments.

Graphical User Interface Operating Systems. The graphic user interface (GUI) OS is a system in which users have direct control of visible objects (such as icons and pointers) and actions that replace command syntax. The next step in the evolution of GUIs is social interfaces. A social interface is a user interface that guides the user through computer applications by using cartoon-like characters, graphics, animation, and voice commands.

Processing Tasks. Operating systems manage processing activities with some task management features that allocate computer resources to optimize each system's assets. The most notable features are described below.

Multiprogramming and Multiprocessing. Multiprogramming involves two or more application modules or programs placed into main memory at the same time. The first module runs on the CPU until an interrupt occurs, such as a request for input. The input request is initiated and handled while the execution of a second application module is started. The execution of the second module continues until another interruption occurs, when execution of a third module begins. When the processing of the interrupt has been completed, control is returned to the program that was interrupted, and the cycle repeats. Because switching among programs occurs very rapidly, all programs appear to be executing at the same time.

In a multiprocessing system, more than one processor is involved. The processors may share input/output devices, although each processor may also control some devices exclusively. In some cases, all processors may share primary memory. As a result, more than one CPU operation can be carried on at exactly the same time; that is, each processor may execute an application module or portion of an application module simultaneously with another. Multiprogramming is implemented entirely by software, whereas multiprocessing is primarily a hardware implementation, aided by sophisticated software.

Time-Sharing. Time-sharing is an extension of multiprogramming. In this mode, a number of users operate online with the same CPU, but each uses a different input/output terminal. An application module of one user is placed into a partition (a reserved section of primary storage). Execution is carried on for a given period of time, a time slice, or until an input/output request (an interrupt) is made. As in multiprogramming, modules of other users have also been placed into primary storage in other partitions. Execution passes on to another application module at the end of a time slice and rotates among all users.

Virtual Memory. Virtual memory allows the user to write a program as if primary memory were larger than it actually is. Users are provided with "virtually" all of the primary storage they need. With virtual memory, all of the pages of an application module need not be loaded into primary memory at the same time. As the program executes, control passes from one page to another. If the succeeding page is already in primary memory, execution continues. If the succeeding page is not in primary memory, a delay occurs until that page is loaded. In effect, primary memory is extended into a secondary storage device.

Virtual Machine OS. A virtual machine is a computer system that appears to the user as a real computer but, in fact, has been created by the operating system. A virtual machine OS makes a single real machine appear as multiple machines to its users, each with its own unique operating system. Each user may choose a different operating system for his or her virtual machine. As a result, multiple operating systems may exist in the real machine at the same time.

System Support Programs. System utilities are programs that have been written to accomplish common tasks such as sorting records, merging sets of data, checking the integrity of magnetic disks, creating directories and subdirectories, restoring accidentally erased files, locating files within the directory structure, managing memory usage, and redirecting output. These are basic tasks for most OSs and application programs. TestDrive, for example, allows you to download software; you try it, and TestDrive helps you either with a payment or with removal of the software. Some hard-disk cleanup software, such as Microsoft's Disk Defragmenter, also called defraggers or diagnostic and repair tools, can help tidy up the hard disk by packing the files together to make more continuous room for new files, locating seldom-used files, leftover temporary files, and other space wasters. Norton's Utilities performs routine housekeeping tasks on hard drives and on secondary storage devices.

System Performance Monitors. System performance monitors monitor computer system performance and produce reports containing detailed statistics concerning the use of system resources, such as processor time, memory space, input/output devices, and system and application programs.

System Security Monitors. System security monitors are programs that monitor the use of a computer system to protect it and its resources from unauthorized use, fraud, or destruction. Such programs provide the computer security needed to allow only authorized users access to the system. Security monitors also control use of the hardware, software, and data resources of a computer system. Finally, these programs monitor use of the computer and collect statistics on attempts at improper use.

System Development Programs. Translating user computer programs written in source code into object or machine code requires the use of compilers or interpreters, which are examples of system development programs. Another example is computer-aided software engineering (CASE) programs.

TG2.4 Programming Languages, Software Development, and CASE Tools

Programming languages provide the basic building blocks for all systems and application software. Programming languages are a set of symbols and rules used to write program code. Each language uses a different set of rules and the syntax that dictates how the symbols are arranged so they have meaning.

The characteristics of the languages depend on their purpose. For example, if the programs are intended to run batch processing, they will differ from those intended to run real-time processing. Languages for Internet programs differ from those intended to run mainframe applications.

THE EVOLUTION OF PROGRAMMING LANGUAGES

The different stages of programming languages are called generations. The term *generation* may be misleading. In hardware generation, older generations are becoming obsolete and are not used. All software generations are still in use. They are shown in Figure TG2.2 and are discussed next.

Machine Language. First generation (1G). Machine language is the lowest-level computer language, consisting of the internal representation of instructions and data. This machine code—the actual instructions understood and directly executable by the CPU—is composed of binary digits. A program using this lowest level of coding is called a machine language program and represents the first generation of programming languages. A computer's CPU is capable of executing only machine language

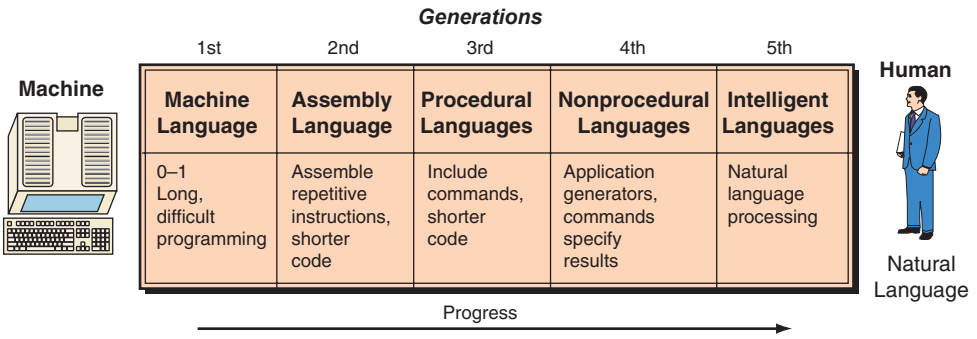

Figure TG2.2 The evolution of programming languages. With each generation, progress is made toward natural language.

programs, which are machine dependent. That is, the machine language for one type of central processor may not run on other types.

Machine language is extremely difficult for programmers to understand and use. As a result, increasingly more user-oriented languages have been developed. These languages make it much easier for people to program, but they are impossible for the computer to execute without first translating the program into machine language. The set of instructions written in a user-oriented language is called a source program. The set of instructions produced after translation into machine language is called the object program.

Assembly Language. Second generation (2G). An assembly language is a more user-oriented language that represents instructions and data locations by using mnemonics, or memory aids, that people can more easily use. Assembly languages are considered the second generation of computer languages. Compared to machine language, assembly language eases the job of the programmer considerably. However, one statement in an assembly language is still translated into one statement in machine language. Because machine language is hardware dependent and assembly language programs are translated mostly on a one-to-one statement basis, assembly languages are also hardware dependent.

A systems software program called an assembler accomplishes the translation of an assembly language program into machine language. An assembler accepts a source program as input and produces an object program as output. The object program is then processed into data, as shown in Figure TG2.3.

HIGH-LEVEL LANGUAGES

High-level languages are the next step in the evolution of user-oriented programming languages. High-level languages are much closer to natural language and

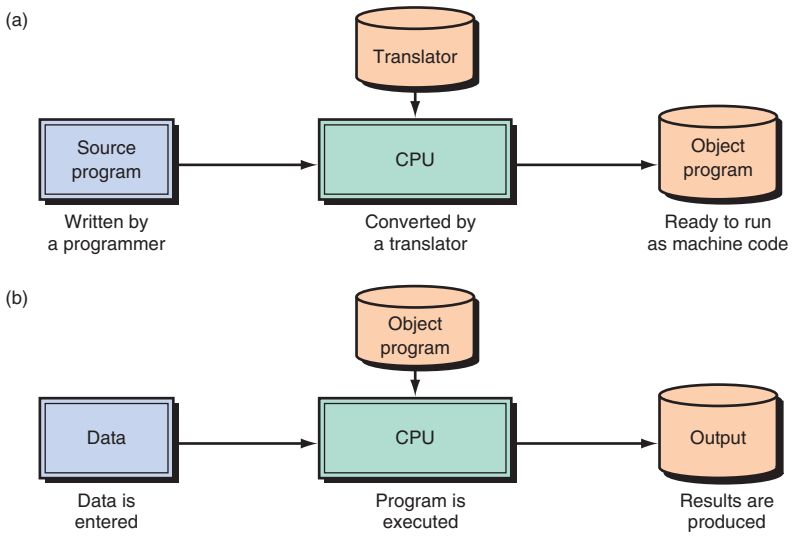

Figure TG2.3 The language translation process.

therefore easier to write, read, and alter. Moreover, one statement in a high-level language is translated into a number of machine language instructions, thereby making programming more productive.

Procedural Languages. Third generation (3G). Procedural languages are the next step in the evolution of user-oriented programming languages. They are also called third-generation languages, or 3GLs. Procedural languages are much closer to so-called natural language (the way we talk) and therefore are easier to write, read, and alter. Moreover, one statement in a procedural language is translated into a number of machine language instructions, thereby making programming more productive. In general, procedural languages are more like natural language than assembly languages are, and they use common words rather than abbreviated mnemonics. Because of this, procedural languages are considered the first level of higher-level languages.

Procedural languages require the programmer to specify—step by step—exactly how the computer will accomplish a task. A procedural language is oriented toward how a result is to be produced. Because computers understand only machine language (i.e., 0s and 1s), higher-level languages must be translated into machine language prior to execution. This translation is accomplished by systems software called language translators. A language translator converts the high-level program, called source code, into machine language code, called object code. There are two types of language translators—compilers and interpreters.

- *Compilers.* The translation of a high-level language program to object code is accomplished by a software program called a compiler. The translation process is called compilation.
- *Interpreters.* An interpreter is a compiler that translates and executes one source program statement at a time. Therefore, interpreters tend to be simpler than compilers. This simplicity allows for more extensive debugging and diagnostic aids to be available on interpreters.
- *Examples of procedural languages.* FORTRAN (Formula Translator) is an algebraic, formula-type procedural language. FORTRAN was developed to meet scientific processing requirements.

COBOL (Common Business-Oriented Language) was developed as a programming language for the business community. The original intent was to make COBOL instructions approximate the way they would be expressed in English. As a result, the programs would be "self-documenting." There are more COBOL programs currently in use than any other computer language.

Microsoft Visual BASIC is the extension of BASIC programming language. This language is famous for its graphical user interface and is ideal for creating prototypes. In 2002, Microsoft launched its .NET platform, so that all of its languages, including Visual BASIC, support this powerful platform.

The C programming language experienced the greatest growth of any language in the 1990s. C is considered more transportable than other languages, meaning that a C program written for one type of computer can generally be run on another type of computer with little or no modification. Also, the C language is easily modified.

Nonprocedural Languages. Fourth generation (4G). Another type of high-level language, called nonprocedural or fourth-generation language (4GL), allows the user to specify the desired results without having to specify the detailed procedures needed to achieve the results. A nonprocedural language is oriented toward what is required. 4GLs, also referred to as command languages, greatly simplify and accelerate the programming process as well as reduce the number of coding errors.

Natural Programming Languages. Fifth generation. Natural language programming (NLP) languages are the next evolutionary step and are sometimes known as

fifth-generation languages or intelligent languages. Translation programs to translate natural languages into a structured, machine-readable form are extremely complex and require a large amount of computer resources. Examples are INTELLECT and ELF. These are usually front-ends to 4GLs (such as FOCUS) that improve the user interface with the 4GLs. Several procedural artificial intelligence languages (such as LISP) are labeled by some as 5GLs.

Object Programming Languages. Object languages were designed to fit new technologies such as multimedia, hypermedia, document management, and the Internet. These languages are described next.

Object-Oriented Programming Languages. Object-oriented programming (OOP) models a system as a set of objects. Like structured programming, OOP tries to manage the behavioral complexity of a system, but it also tries to manage the information complexity of a system. The object-oriented (OO) approach involves programming, operating systems environment, object-oriented databases, and a new way of approaching business applications.

Concepts of the Object-Oriented Approach. The basic concepts of OO are objects, classes, message passing, encapsulation, inheritance, and polymorphism. Since these concepts sound very complex and technical at first, it may be helpful to relate them to aspects of graphical user interfaces in popular operating systems, such as Windows and Mac OS X for Apple's computers. These interfaces were developed through object-oriented programming, and they incorporate object-oriented features.

Object-oriented systems view software as a collection of interacting objects. An object models things in the real world. These things may be physical entities such as cars, students, or events. Or they may be abstractions, such as bank accounts, or aspects of an interface, such as a button or a box to enter text.

When we refer to an object, we can have two possible meanings: a class or an instance. A class is a template or general framework that defines the methods and attributes to be included in a particular type of object. An object is a specific instance of a class, able to perform services and hold data. For example, "student" may be a class in a student registration system. A particular student, John Kim, can be an instance of that class, and thus an object.

Objects have data associated with them. The data elements are referred to as attributes or as variables because their values can change. For example, the John Kim object could hold the data that he is a senior, majoring in management information systems, and registering for the fall quarter.

Objects exhibit behaviors, which are things that they do. The programmer implements these behaviors by writing sections of code that perform the methods of each object. Methods are the procedures or behaviors performed by an object that will change the attribute values of that object. Methods are sometimes referred to as the operations that manipulate the object. Common behaviors include changing the data in an object and communicating information on data values. By clicking on a "check box" in a Windows system, a user initiates the behavior that changes the attribute to "checked" and shows an X or checkmark in the box.

Objects interact with each other using messages. These messages represent requests to exhibit the desired behaviors. The object that initiates a message is the sender, and the object that receives a message is the receiver. When we interact with objects, we send messages to them and they may also send messages to us. Clicking on a button, selecting an item from a menu, and dragging and dropping an icon are ways of sending messages to objects. These messages may activate methods in the recipient objects, and in turn new messages may be generated.

Message passing is the only means to get information from an object because an object's attributes are not directly accessible. The inaccessibility of data in an object is called *encapsulation*, or information hiding. By hiding its variables, an object protects other objects from the complications of depending on its internal structure.

The other objects do not have to know each variable's name, the type of information it contains, or the physical storage format of the information. They only need to know how to ask the object for information.

With inheritance, a class of objects can be defined as a special case of a more general class, automatically including the method and variable definitions of the general class. Special classes of a class are subclasses, and the more general class is a superclass. For example, the student class is a subclass of human being, which is the superclass. The student class may be further divided into in-state students, out-of-state students, or scholarship students, which would be subclasses of the student class. This type of organization results in class hierarchies.

Inheritance is particularly valuable because analysts can search through predefined class hierarchies, called class libraries, to find classes that are similar to the classes they need in a new system. This process saves large amounts of time. For example, if the end user needs to deal with students as a class of objects, the analyst may be able to find a general class that is similar to the student class as viewed by the end user. Therefore, the analyst can reuse information from an existing class instead of starting from the beginning to define a student class. The relationship between classes and subclasses is shown in Figure TG2.4.

Polymorphism is the ability to send the same message to several different receivers (objects) and have the message trigger the desired action. For example, suppose that there are three classes of objects in a tuition-and-fee system: in-state students, out-of-state students, and scholarship students. We must calculate tuition and fees for all three types of student (classes) while noting that the tuition and fees will differ for the three classes. Polymorphism allows us to send the same "calculate tuition and fees" message to these three different classes and have the correct tuition and fees calculated for each one.

Programming with OO. Building programs and applications using OO programming languages is similar to constructing a building using prefabricated parts. The object containing the data and procedures is a programming building block. The same objects can be used repeatedly, a process called *reusability*. By reusing program code, programmers can write programs much more efficiently and with fewer errors. Object-oriented programming languages offer advantages such as reusable code, lower costs, reduced errors and testing, and faster implementation times. Popular object-oriented programming languages include Smalltalk, C++, Java, and C#.

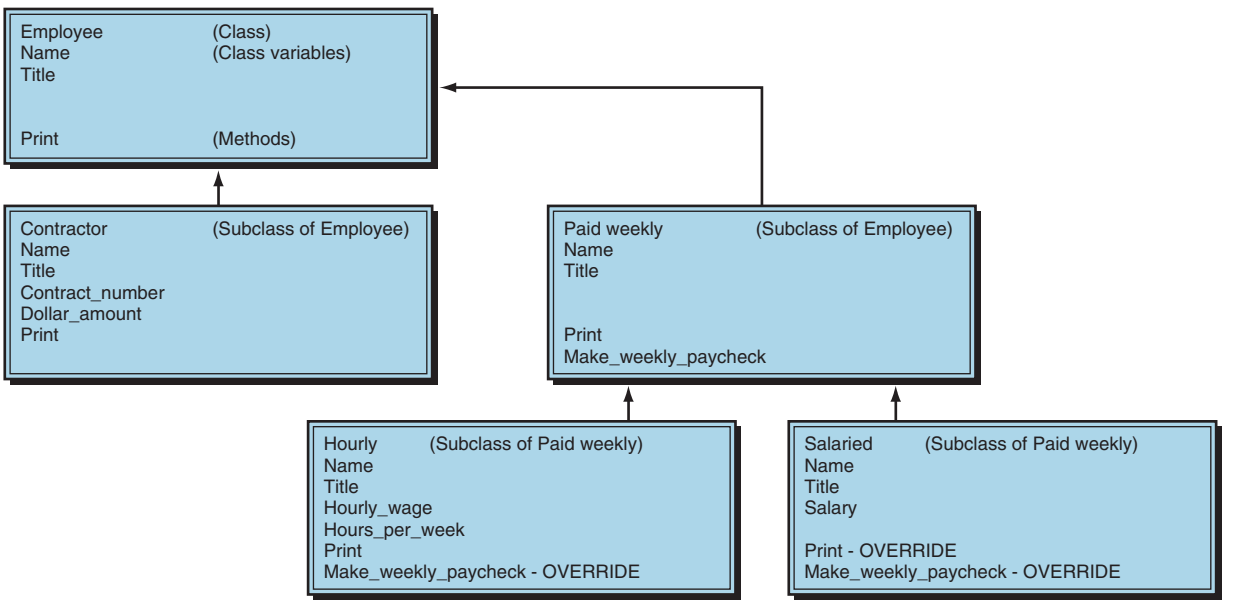

Figure TG2.4 Object classes, subclasses, inheritance, and overriding. (*Source:* © Courtesy of Apple Corporation. Used with permission.)

Smalltalk. Smalltalk is a pure object-oriented language developed at the Xerox Palo Alto Research Center. The syntax is fairly easy to learn, being much less complicated than C and C++.

C++. C++ is a direct extension of the C language, with 80 to 90 percent of C++ remaining pure C.

Unified Modeling Language (UML). Developing a model for complex software systems is as essential as having a blueprint for a large building. The UML is a language for specifying, visualizing, constructing, and documenting the artifacts (such as classes, objects, etc.) in object-oriented software systems. The UML makes the reuse of these artifacts easier because the language provides a common set of notations that can be used for all types of software projects.

Visual Programming Languages. Programming languages that are used within a graphical environment are often referred to as visual programming languages. Visual programming allows developers to create applications by manipulating graphical images directly, instead of specifying the visual features in code. These languages use a mouse, icons, symbols on the screen, or pull-down menus to make programming easier and more intuitive. Visual Basic and Visual C++ are examples of visual programming languages.

WEB PROGRAMMING LANGUAGES AND SOFTWARE

Several languages exist specifically for the Internet. Most notable is HTML.

Hypertext Markup Language. The standard language the Web uses for creating and recognizing hypermedia documents is the Hypertext Markup Language (HTML). HTML is loosely related to the Standard Generalized Markup Language (SGML), which is a method of representing document-formatting languages. Languages such as HTML that follow the SGML format allow document writers to separate information from document presentation. That is, documents containing the same information can be presented in a number of different ways. Users have the option of controlling visual elements such as fonts, font size, and paragraph spacing without changing the original information.

HTML is easy to use. Web documents are typically written in HTML and are usually named with the suffix .html or .htm. HTML documents are standard 7- or 8-bit ASCII files with formatting codes that contain information about layout (text styles, document titles, paragraphs, lists) and hyperlinks. The HTML standard supports basic hypertext document creation and layout, as well as interactive forms and defined "hot spots" in images.

Hypertext is an approach to data management in which data is stored in a network of nodes connected by links (called hyperlinks). Users access data through an interactive browsing system. The combination of nodes, links, and supporting indexes for any particular topic is a hypertext document. A hypertext document may contain text, images, and other types of information such as data files, audio, video, and executable computer programs.

The World Wide Web uses Uniform Resource Locators (URLs) to represent hypermedia links and links to network services within HTML documents. The first part of the URL (before the two slashes) specifies the method of access. The second part is typically the address of the computer where the data or service is located. A URL is always a single unbroken line with no spaces.

Dynamic HTML is the next step beyond HTML. Dynamic HTML provides advances that include the following:

- It provides a richer, more dynamic experience for the user on Web pages, making the pages more like dynamic applications and less like static content. It lets the user interact with the content of those pages without having to download additional content from the server. This means that Web pages using Dynamic HTML provide more exciting and useful information.

- Dynamic HTML gives developers precise control over formatting, fonts, and layout, and it provides an enhanced object model for making pages interactive.
- It serves as the foundation for crossware, a new class of platform-independent, on-demand applications built entirely using Dynamic HTML, Java, and JavaScript. Netscape Netcaster, a component of Netscape Communicator, is Netscape's first crossware application.

Enhancements and variations of HTML make possible new layout and design features on Web pages. For example, cascading style sheets (CSSs) are an enhancement to HTML that act as a template defining the appearance or style (such as size, color, and font) of an element of a Web page, such as a box.

XML. XML (eXtensible Markup Language) is optimized for document delivery across the Net. It is built on the foundation of SGML. XML is a language for defining, validating, and sharing document formats. It permits authors to create, manage, and access dynamic, personalized, and customized content on the Web—without introducing proprietary HTML extensions. XML is especially suitable for electronic commerce applications. XQuery is an XML query language developed and standardized by the World Wide Web Consortium (W3C). XQuery is a powerful and convenient language designed for processing XML data: that means not only files in XML format, but also other data, including databases whose structure is similar to XML. XQuery's purpose is to find, retrieve, and rearrange data viewed through the lens of XML.

Java. Java is an object-oriented programming language developed by Sun Microsystems. The language gives programmers the ability to develop applications that work across the Internet. Java is used to develop small applications, called applets, which can be included in an HTML page on the Internet. When the user uses a Java-compatible browser to view a page that contains a Java applet, the applet's code is transferred to the user's system and executed by the browser.

JavaScript. JavaScript is an object-oriented scripting language developed by Netscape Communications for client/server applications. It allows users to add some interactivity to their Web pages. Many people confuse JavaScript with Java. There is no relationship between these two programming languages. JavaScript is a very basic programming language and bears no relationship to the sophisticated and complex language of Java.

JavaBeans. JavaBeans is the platform-neutral component architecture for Java. It is used for developing or assembling network-aware solutions for heterogeneous hardware and operating system environments, within the enterprise or across the Internet. JavaBeans extends Java's write once, run anywhere capability to reusable component development. JavaBeans runs on any operating system and within any application environment.

ActiveX. ActiveX is a set of technologies from Microsoft that combines different programming languages into a single, integrated Web site. Before ActiveX, Web content was static, two-dimensional text and graphics. With ActiveX, Web sites come alive using multimedia effects, interactive objects, and sophisticated applications that create a user experience comparable to that of high-quality CD-ROM titles. ActiveX is not a programming language as such, but rather a set of rules for how applications should share information.

ASP. ASP (Active Server Pages) is a Microsoft CGI-like (common gateway interface) technology that allows you to create dynamically generated Web pages from the server side using a scripting language. Because ASP can talk to ActiveX controls and other OLE programs, users can take advantage of many report writers, graphic

controls, and all of the ActiveX controls that they may be used to. ASP can also be programmed in VBScript or JavaScript, enabling users to work in the language that they are most comfortable with.

Web-based software is software that is installed and runs on servers and then is accessed from a personal computer over a network. The personal computer often uses a Web browser to access the software that is housed on a server. Examples of this are Google Docs.

COMPUTER-AIDED SOFTWARE ENGINEERING TOOLS

Computer-aided software engineering (CASE) is a tool for programmers, systems analysts, business analysts, and systems developers to help automate software development and at the same time improve software quality.

CASE is a combination of software tools and structured software development methods. The tools automate the software development process, while the methodologies help identify those processes to be automated with the tools. CASE tools often use graphics or diagrams to help describe and document systems and to clarify the interfaces or interconnections among the components (see Figure TG2.5). They are generally integrated, allowing data to be passed from tool to tool.

Categories of CASE Tools. CASE tools support individual aspects or stages of the systems development process, groups or related aspects, or the whole process. Upper CASE (U-CASE) tools focus primarily on the design aspects of systems development, for example, tools that create data flow or entity-relationship diagrams. Lower CASE (L-CASE) tools help with programming and related activities, such as testing, in the later stages of the life cycle. Integrated CASE (I-CASE) tools incorporate both U-CASE and L-CASE functionality and provide support for many tasks throughout the SDLC.

CASE tools may be broken down into two subcategories: toolkits and workbenches. A toolkit is a collection of software tools that automate one type of software task or one phase of the software development process. A CASE workbench is a collection of software tools that are interrelated based on common assumptions about the development methodology being employed. A workbench also uses the data repository containing all of the technical and management information needed

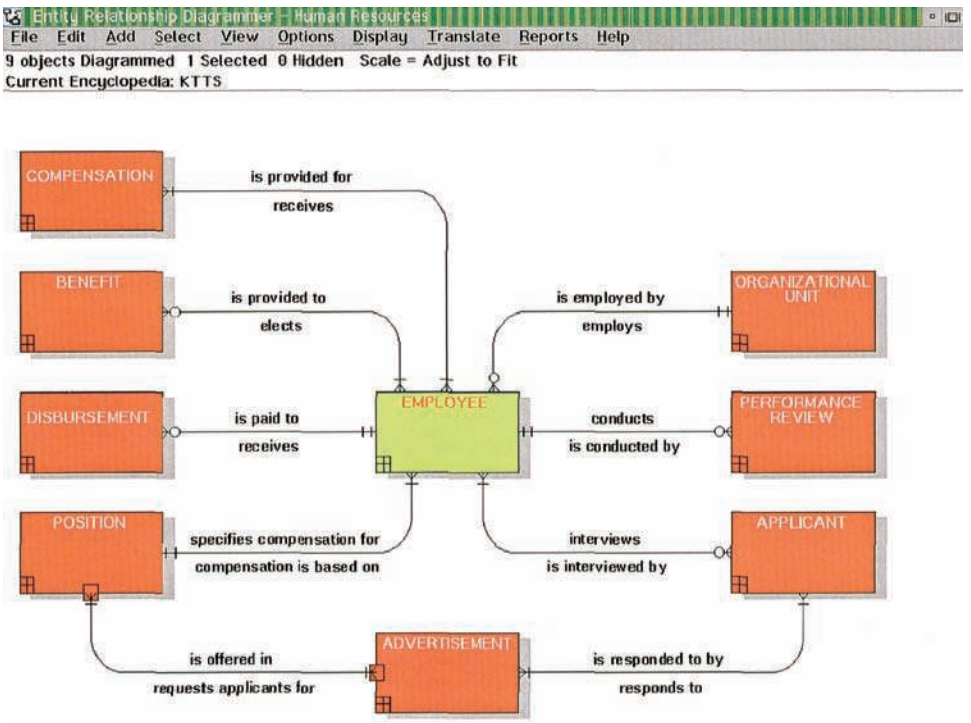

Figure TG2.5 CASE display.

to build the software system. Ideally, workbenches provide support throughout the entire software development process and help produce a documented and executable system.

Because most CASE tools are graphical in nature and have the ability to produce working prototypes quickly, non–technically trained users can participate more actively in the development process. They can see what the completed system will look like before it is actually constructed, resulting in fewer misunderstandings and design mistakes.

Using CASE can help make revising an application easier. When revisions are needed, one need only change specifications in the data repository rather than the source code itself. This also enables prototype systems to be developed more quickly and easily. Some CASE tools help generate source code directly, and the benefits can be significant.

CASE tools also have disadvantages. A lack of management support for CASE within organizations can be a problem. CASE is very expensive to install, train developers on, and use properly. Many firms do not know how to measure quality or productivity in software development and therefore find it difficult to justify the expense of implementing CASE. In addition, the receptivity of professional programmers can greatly influence the effectiveness of CASE. Many programmers who have mastered one approach to development are hesitant to shift to a new method.

TG2.5 Software Issues and Trends

The importance of software in computer systems has brought new issues and trends to the forefront for organizational managers. These issues and trends include software evaluation and selection, software licensing, software upgrades, software defects, malware, open systems, open-source software, shareware, software piracy, services-oriented architecture, and autonomic computing.

SOFTWARE EVALUATION AND SELECTION

There are dozens or even hundreds of software packages to choose from for almost any topic. The software evaluation and selection decision is a difficult one that is affected by many factors. The first part of the selection process involves understanding the organization's software needs and identifying the criteria that will be used in making the eventual decision. Once the software requirements are established, specific software should be evaluated. An evaluation team composed of representatives from every group that will have a role in building and using the software should be chosen for the evaluation process. The team will study the proposed alternatives and find the software that promises the best match between the organization's needs and the software capabilities.

SOFTWARE LICENSING

Proprietary Licensing. Vendors spend a great deal of time and money developing their software products. To protect this investment, they must protect their software from being copied and distributed by individuals and other software companies. A company can copyright its software, which means that the U.S. Copyright Office grants the company the exclusive legal right to reproduce, publish, and sell that software.

Open-Source Licensing. There are also Open Source Software licenses. Software licensed under an Open Source Software license can legally be freely distributed or sold by anyone. Examples of popular Open Source Software licenses include but are not limited to GNU General Public License (GPL), BSD, and Apache License 2.0.

Software licensed under an Open Source Software license must be free to be redistributed by anyone and the software must be distributed with the source code along with the right to modify the source code and to make derived works under the same terms of the same license. For more information please see the Open Source Initiatives "Open Source Definition" at *http://www.opensource.org/docs/osd*.

SOFTWARE UPGRADES

Another issue of interest to organizational management is software upgrades (also known as software maintenance). Software vendors revise their programs and sell new versions often. The revised software may offer valuable enhancements, or, on the other hand, it may offer little in terms of additional capabilities. Also, the revised software may contain bugs.

Deciding whether to purchase the newest software can be a problem for organizations and their IS managers. It is also difficult to decide whether to be one of the first companies to buy and take strategic advantage of new software before competitors do, but risk falling prey to previously undiscovered bugs.

SOFTWARE DEFECTS

Good software is usable, reliable, defect-free, cost-effective, and maintainable. However, all too often, computer program code is inefficient, poorly designed, and riddled with errors. Software defects have wrecked a European satellite launch, delayed the opening of Denver International Airport for a year, and destroyed a NASA Mars mission. In another example, on the same day that Microsoft first released Windows XP, the company posted 18 megabytes of patches on its Web site: bug fixes, compatibility updates, and enhancements.

With our dependence on computers and networks, the risks are getting worse. According to the Software Engineering Institute (SEI), professional programmers make on average 100 to 150 errors in every thousand lines of code they write. Using SEI's figures, Windows XP, with its 41 million lines of code, would have over 4 million bugs. The industry recognizes the problem, but the problem is so enormous that only initial steps are being taken. One step is better design and planning at the beginning of the development process.

OPEN SYSTEMS

The concept of open systems refers to a model of computing products that work together. Achieving this goal is possible through the use of the same operating system with compatible software on all of the different computers that would interact with one another in an organization. A complementary approach is to produce application software that will run across all computer platforms. If hardware, operating systems, and application software are designed as open systems, the user would be able to purchase the best software for the job without worrying about whether it will run on particular hardware. As an example, much Apple Macintosh application software will not run on Wintel (Windows-Intel) PCs. However, Windows can run on the newer Apple computers that use Intel CPUs. Neither of these will run on a mainframe.

Certain operating systems, such as UNIX, will run on almost any machine. Therefore, to achieve an open-systems goal, organizations frequently employ UNIX on their desktop and larger machines so that software designed for UNIX will operate on any machine. Recent advances toward the open-systems goal involve using the Java language, which can be run on many types of computers, in place of a traditional operating system.

OPEN-SOURCE SOFTWARE

Open systems should not be confused with open-source software. Open-source software is software made available in source code form at no cost to developers. There are many examples of open-source software, including but not limited to the GNU (GNU's Not UNIX) suite of software (*gnu.org/*) developed by the Free Software Foundation (*fsf.org/*), the Linux kernel (*kernel.org/*), Apache Web server (*apache.org/*), sendmail SMTP (Send Mail Transport Protocol) e-mail server (*sendmail.org/*), the Perl programming language (*perl.com*), the Mozilla Firefox Web browser (*mozilla.org/*), Oracle Open Office (*oracle.com*), Google's Android OS, and Symbian v3.

Open-source software is, in many cases, more reliable than proprietary software. Because the code is available to many developers, more bugs are discovered, are discovered early and quickly, and are fixed immediately. Support for open-source software is also available from companies that provide products derived from the

software, for example, Red Hat for Linux (*redhat.com/*). These firms provide education, training, and technical support for the software for a fee.

Linux was used to create the astounding effects for the movie trilogy *Lord of the Rings.* More than 200 workstations and 450 dual-processor servers ran on Red Hat Linux 7.3 to identify system resources and distribute rendering jobs, like shadows and reflections, across idle processors to speed up scene creation.

In terms of security and stability, open-source code is better because many people can search it so that hidden problems can be eradicated earlier than those in proprietary code. In addition to this, some entrepreneurs are afraid of being locked in by the proprietary code.

Open-source software is sometimes produced by vendors but is often produced by groups of volunteers. It is normally distributed at little or no cost by distributors who hope to make money by providing training, consulting work, add-on products, and custom software. Initially, it was perceived as unreliable and not a viable alternative to proprietary software produced by large firms with a strong reputation and with significant financial and people resources. Linux has altered this perception by using open-source software; companies can save significant money without compromising on quality, support, and future enhancements.

SHAREWARE AND FREEWARE

Shareware is software for which the user is expected to pay the author a modest amount for the privilege of using it. Freeware is software that is free. Both help to keep software costs down. Shareware and freeware are often not as powerful (do not have the full complement of features) as the professional versions, but some users get what they need at a good price. These are available now on the Internet in large quantities (*download.com/*). A deficiency of such software is the possible introduction of viruses or spyware. Some popular packages are WinZip, Adobe Reader, Mozilla, Zero Pop-up, and KaZaa.