TECHNOLOGY GUIDE

2

Software

T2.1 SOFTWARE FUNDAMENTALS AND TYPES

Computer hardware cannot perform a single act without instructions. These instructions are known as **software** or computer programs. Software is at the heart of all computer applications. Computer hardware is, by design, general purpose. Software, on the other hand, enables the user to tailor a computer to provide specific business value.

Software Fundamentals

Software consists of **computer programs,** which are sequences of instructions for the computer. The process of writing (or *coding*) programs is called *programming,* and individuals who perform this task are called *programmers.*

Unlike the hardwired computers of the 1950s, modern software uses the **stored-program concept,** in which stored software programs are accessed and their instructions are executed (followed) in the computer's CPU. Once the program has finished executing, a new program is loaded into the main memory and the computer hardware addresses another task.

Computer programs included **documentation,** which is a written description of the functions of the program. Documentation helps the user operate the computer system and 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, the knowledge of how to use the program or how it is designed may be lost.

Types of Software

There are two major types of software: application software and systems software.

Application software is a set of computer instructions, written in a programming language. The instructions direct computer hardware to perform specific data or information processing activities that provide functionality to the user. This functionality may be broad, such as general word processing, or narrow, such as an organization's payroll program. An **application program** applies a computer to a need, such as increasing productivity of accountants or improved decisions regarding an inventory level. **Application programming** creates or modifies and improves application software.

Systems software acts primarily as an intermediary between computer hardware and application programs, and knowledgeable users may also directly manipulate it. Systems software provides important self-regulatory functions for computer systems, such as loading itself when the computer is first turned on, as in *Windows Professional;* managing hardware resources such as secondary storage for all applications; and providing commonly used sets of instructions for all applications to use. **Systems programming** either creates or modifies systems software.

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 T2.1 shows that systems software is a necessary intermediary between hardware and application software; the application software cannot run without the systems software.

Unlike computer hardware, which can be designed and manufactured on automated assembly lines, most software must be programmed by hand. Computer hardware power grows roughly by a factor of two every 18 months (see *Moore's law* in Chapter 13), but computer software power barely doubles in eight years. This lag presents a great challenge to software developers and to information systems users in general.

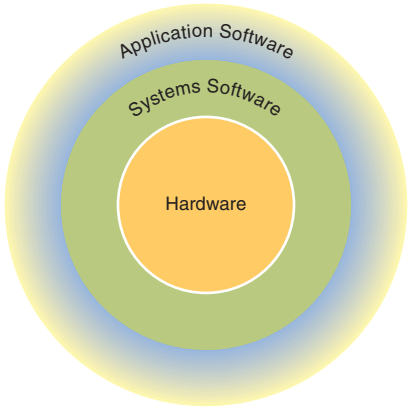

FIGURE T2.1 Systems software serves as intermediary between hardware and functional applications.

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

T2.2 APPLICATION SOFTWARE

Because there are so many different uses for computers, there are a correspondingly large number of different application programs. Application software includes proprietary application software and off-the-shelf application software. **Tailor-made application software** addresses a specific or unique business need for a company. This type of software may be developed in-house by the organization's information systems personnel or it may be commissioned from a software vendor. Such specific software programs developed for a particular company by a vendor are called **contract software.**

Alternatively, **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 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.

If a package is not available for a certain situation, it is necessary to build the application using programming languages or software development tools. There are also *general-purpose application programs* that are not linked to any specific business task, but instead support general types of information processing. The most widely used general-purpose application packages are spreadsheet, data management, word processing, desktop publishing, graphics, multimedia, and communications.

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.

Many decision support and business applications are built with programming languages rather than with general-purpose application programs. This is

Student Name	Exam 1	Exam 2	Exam 3	Total Points	Grade
Carr, Harold	73	95	90	258	B
Ford, Nelson	92	90	81	263	B
Lewis, Bruce	86	88	98	272	A
Snyder, Charles	63	71	76	210	C
Average	78.5	86.0	86.25	250.75	

FIGURE T2.2 Sample calculation of student grades in a spreadsheet.

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. 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.

General-Purpose Application Programs

SPREADSHEETS. **Spreadsheet software** transforms a computer screen into a ledger sheet, or grid, of coded rows and columns (see Figure T2.2). Users can enter numeric or textual data into each grid location, called a *cell*. In addition, a formula or macro can also be entered into a cell to obtain a calculated answer displayed in that cell's location. The term *macro* refers to a single instruction or formula that combines a number of other simpler instructions. A user-defined macro can enhance and extend the basic instructions and commands that are furnished with the spreadsheet. Spreadsheet packages include a large number of already-programmed statistical, financial, and other business formulas. They are known as *functions*.

Computer spreadsheet packages are used primarily for decision support such as in financial information processing (e.g., such as income statements or cash flow analysis). However, they also are relevant for many other types of data that can be organized into rows and columns. Spreadsheets are usually integrated with other software, such as graphics and data management, to form *software suites*. Therefore, they may be called *integrated packages*.

DATA MANAGEMENT. Data management software supports the storage, retrieval, and manipulation of data. There are two basic types of data management software: simple **filing programs** patterned after traditional, manual data filing techniques, and **database management systems (DBMSs)** that take advantage of a computer's extremely fast and accurate ability to store and retrieve data (see Technology Guide 3).

A *file* is a collection of related records organized alphabetically, chronologically, hierarchically in levels, or in some other manner. File-based management software is typically simple to use and often very fast, but it is difficult and time-consuming to modify because of the structured manner in which the files are created.

Database management software was addresses the problems of file-based management software. A **database** is a collection of files serving as the data resource for computer-based information systems. In a database, all data are integrated with established relationships. An example of database software is provided in Figure T2.3.

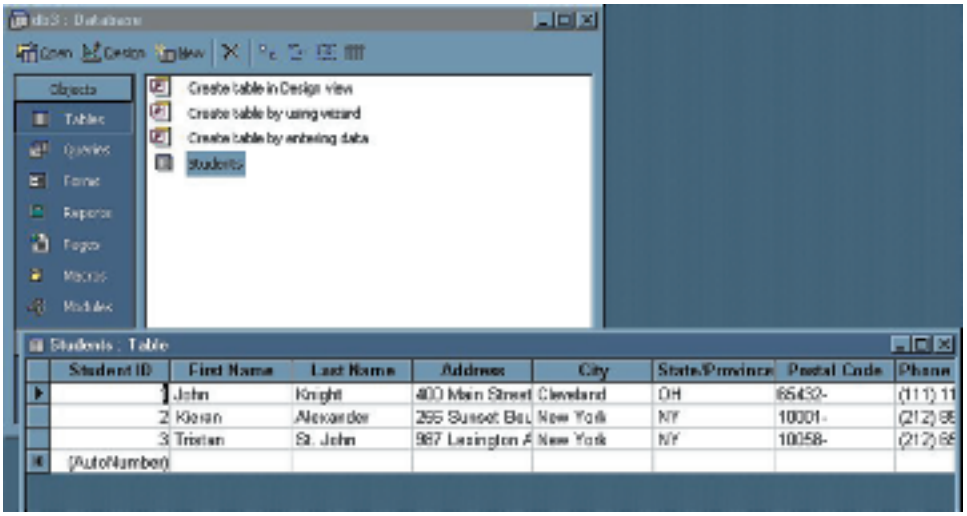

FIGURE T2.3 Database software.

Microsoft Office includes a database management system for personal use. An example for corporate use is the new Oracle Database 10g, which is packed with features designed to make the database administrator's (DBA)'s job easier, either by completely automating tasks or by transferring control of important functions to the server. This allows DBAs to manage large, complex environments with very little effort.

WORD PROCESSING. **Word processing** software allows the user to manipulate text rather than just numbers. Modern word processors contain many productive writing features. A typical word processing software package consists of an integrated set of programs, including an editor, a formatting program, a print program, a dictionary, a thesaurus, a grammar checker, a mailing list program, and integrated graphics, charting, and drawing programs. **WYSIWYG** (**W**hat **Y**ou **S**ee **I**s **W**hat **Y**ou **G**et) word processors have the added advantage of displaying the text material on the screen exactly—or almost exactly—as it will look on the final printed page (based on the type of printer connected to the computer).

DESKTOP PUBLISHING. In the past, newsletters, announcements, advertising copy, and other specialized documents had to be laid out by hand and then typeset. **Desktop publishing** software allows microcomputers to perform these tasks directly. Photographs, diagrams, and other images can be combined with text, including several different fonts, to produce a finished, camera-ready document. When printed on a high-resolution laser printer, the product is difficult to distinguish from one that was produced by a professional typesetter.

GRAPHICS. *Graphics software* allows the user to create, store, and display or print charts, graphs, maps, and drawings. Graphics software enables users to absorb more information more quickly, to spot relationships and trends in data more easily, and to make points more emphatically. There are three basic categories of graphics software packages: presentation graphics, analysis graphics, and engineering graphics.

Presentation Graphics. This software allows users to create pseudo-three-dimensional images, superimpose multiple images, highlight certain aspects of a drawing, and create freehand drawings. These packages typically contain drawing tools, presentation templates, various font styles, spell-checking routines, charting aids, and tools to aid in assembling multiple images into a complete presentation. Many packages have extensive built-in tutorials and libraries of *clip-art*—pictures that can be electronically "clipped out" and "pasted" into the finished image.

Analysis Graphics. These applications additionally provide the ability to present previously analyzed data—such as statistical data—in graphic formats like bar charts, line charts, pie charts, and scatter diagrams. The charts may also include elements of different textures, labels, and headings. Some packages will prepare three-dimensional displays.

Engineering Graphics. Various engineering software programs are available to shorten development time of applications and to increase productivity of draftspersons and engineers. Most notable are computer-aided design and computer-aided manufacturing (for details see Chapter 6).

MULTIMEDIA. There are two general types of multimedia software: presentation and interactive. *Presentation software* presents a sequential procession of information similar to a movie or television show. The order of events is fixed, although the presentation can be stopped and started. Speakers and trade show booths often use multimedia presentation software for marketing purposes. *Interactive software* allows a user to alter the sequence or flow of information, similar to looking at an encyclopedia or a photo album.

Educational, interactive multimedia products are popular at museums or at information kiosks and show great potential for public and private education both within and outside the classroom.

COMMUNICATIONS SOFTWARE. Computers are often interconnected in order to share or relate information. To exchange information, computers utilize *communications software*. This software allows computers located close together or far apart to exchange data over dedicated or public cables, telephone lines, satellite relay systems, or microwave circuits (see Technology Guide 4).

When communications software exists in both the sending and receiving computers, they are able to establish and relinquish electronic links, code and decode data transmissions, verify transmission errors (and correct them automatically), compress data streams for more efficient transmission, and manage the transmission of documents. Communications software establishes the switched routings needed to ensure successful "end-to-end" transmissions; it establishes electronic contact ("handshaking") between computers, and assures that data will be sent in the proper format and at the proper speed. It detects transmission speeds and codes, and routes information to the appropriate hardware. Communications software checks for and handles transmission interruptions or conflicting transmission priorities. Other communications software includes terminal emulators, remote control software, and fax programs. E-mail and desktop video conferencing rely on communications software.

Remote control software can let a remote user dial up and operate a computer as if that user is sitting in front of it. Representative software includes Symantec's PcAnywhere, Netopia's Timbuktu Pro, AT&T's WinVNC, and Microsoft NetMeeting.

SPEECH-RECOGNITION SOFTWARE. Two categories of **speech-recognition software** (also known as *voice recognition software*) are available today: discrete speech and continuous speech. *Discrete speech recognition* can interpret only one word at a time, so users must place distinct pauses between words. This type of voice recognition can be used to control PC software (by using words such as "execute" or "print"). But it is inadequate for dictating a memo, because users find it difficult to speak with measurable pauses between every word and still maintain trains of thought.

Software for *continuous speech recognition* can interpret a continuing stream of words. The software must understand the context of a word to determine its correct spelling, and be able to overcome accents and interpret words very quickly. These requirements mean that continuous speech-recognition software must have a computer with significantly more speed and memory than discrete speech software.

Many firms and people use speech-recognition software when use of a mouse and a keyboard is impractical. For example, such software can provide an excellent alternative for users with disabilities, repetitive strain injuries, or severe arthritis. Well-known products include IBM's ViaVoice and Dragon's Naturally Speech 7.0.

TEXT-TO-SPEECH. *Text-to-speech systems* convert computer text into voice. A text file is sent through special software that converts it into spoken words, which are output through speakers. Blind people use text-to-speech systems to listen to computer-based documents. People who cannot talk use text-to-speech systems to choose their words and have their computer speak for them.

Wireless phone companies are using text-to-speech systems to develop voice portals. Users call a phone number at the voice portal to hear a wide variety of real-time data, such as local weather, stock quotes, and traffic updates. This information is retrieved directly from Internet-based information systems and converted to speech (see *tellme.com*).

BRAIN WAVE INPUT. A cutting-edge technology is *brain wave input,* also called a *neural interface.* (For more information on neural interfaces, visit *www.sciam.com/1096issue/1096lusted.html.*) These systems use electrical signals from the brain as an input method. By using biofeedback techniques, users can learn how to control certain types of brain waves. A computer translates the brain wave activity into some action on the computer. For example, the U.S. Air Force has tried out this technology in flight simulations. It trained pilots to control certain brain waves that would make a plane turn right or left. In experiments, some pilots were able to control the plane, turning right or left at will; others could produce only one turn, running in circles.

SOFTWARE SUITES. Software **suites** are collections of application software packages in a bundle. Software suites can include word processors, spreadsheets, database management systems, graphics programs, communications tools, and others. Microsoft Office, Novell Perfect Office, and Lotus SmartSuite are widely used software suites for personal computers. Each of these suites includes a spreadsheet program, word processor, database program, and graphics package with the ability to move documents, data, and diagrams among them. In addition to end-user-type suites, such as described above, there are software kits for system developers, such as CASE tools, which are described later.

WORKGROUP SOFTWARE. Workgroup software, or **groupware,** helps groups and teams work together by sharing information and by controlling workflow within the group. The use of this type of software has grown because of a need for groups to work together more effectively, coupled with technological progress in networking and group-support products.

Many groupware products are designed to support specific group-related tasks such as project management, scheduling (called calendaring), workflow, and retrieving data from shared databases. For example, Lotus Notes is designed as a system for sharing text and images, and contains a data structure that is a combination of a table-oriented database and an outline. Using Lotus Notes, groups of users working together on projects are able to see each other's screens, share data, and exchange ideas and notes in an interactive mode. Such capabilities increase the productivity of work groups. A well-known workgroup product in academia is GroupSystems. It is suitable for groups working together in one room, to support brainstorming, categorization, voting, etc.

Other groupware products focus primarily on the flow of work in office settings. These products provide tools for structuring the process by which information for a particular task is managed, transferred, and routed. Other groupware systems are basically e-mail systems extended by classifying messages and using those classifications to control the way messages are handled. Of special interest are *group decision support systems (GDSSs)*, which are presented in Chapter 11.

OTHER APPLICATION SOFTWARE. There exist hundreds of other application software products. Of special interest to business managers are:

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 will increase as mobile wireless devices begin to access company sites 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.

ORGANIZATION-WIDE APPLICATIONS. **Enterprise software** consists of programs that manage the vital operations of an organization (enterprise), 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—Systeme Anwendung Produkte (SAP) AG, Oracle Corporation, PeopleSoft Inc., and Computer Associates—are developing software programs that make the jobs of business users and IT personnel easier. Because of the cost, complexity, and time needed to implement enterprisewide corporate applications, many companies are purchasing only the specific

application (or module) required, such as manufacturing, financial, or sales force automation.

There are two major approaches to *enterprise architecture* (EA): a top-down approach and a bottom-up approach. A *top-down approach* assumes a comprehensive scope and strictly follows a structured process. If organizations need to address significant inefficiencies and redundancies in their business process or application portfolios and can wait at least a year for measurable benefits, they should start with the top-down approach. Alternatively, organizations that need results to affect the bottom line quickly or those where rampant technology diversity has degraded service-delivery quality should start with the *bottom-up approach.* The best approach may be a hybrid, but the most important point is that the approach must fit the culture of the organization.

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. Microsoft's HailStorm services platform depends on this technology: When someone tries to get in touch with you, the system will detect your network location and level of accessibility and may even e-mail, page, or call you.

SCHEMATICS SOFTWARE. Microsoft Visio-2000 can create crystal-clear network and telecommunications schematics, space plans, and even detailed HVAC layouts, to quickly communicate just what goes where, when, and how. Besides this, it can help you draw many diagrams about systems analysis and design including DFD, ERD, UML and also help you complete forward-engineering as well as backward-engineering tasks.

EXAMPLES OF NEW APPLICATION SOFTWARE. New application software is being developed and marketed each year. Examples of such software products are the following.

- United Internet Technologies (UIT) has developed a solution that allows you to maximize the advantages of both CD-ROMs and the Internet in direct marketing. This solution is called digitally integrated video overlay (Divo) software which blends video from a CD-ROM into a Web site, providing fully integrated, full-screen, real-time video on the Internet without a high-speed connection. The Divo CD-ROM installs proprietary software onto the user's computer drive that allows Divo to take its cues from the Web site, enabling you to control the content a viewer sees according to the day of the week or time of day or even the month. The information on the CD is not updated from the Web site; everything is there on the CD when it is delivered into the user's hands. The coding from the Web site simply instructs the CD what to play and when.
- Microsoft's new software architecture, called the Dynamic Systems Initiative, supports the concept of autonomic computing. It attempts to provide a software environment for more automated and efficient and less complex data centers. Initially, they will have new tools in Windows Server 2003, which gives more control over CPU and memory utilization, for managing storage area networks. Next will be technology called Automated Deployment Service (ADS) that will support the intelligent provisioning of Windows and related software for faster setup on servers.

- A Swedish company, Cycore, has provided HMV.com interactive three-dimensional software called Cult3D. With it, shoppers can now electronically flip open the CD cover and zoom in to read the lyrics and liner notes. Cult3D is a multi-platform 3-D rendering engine that allows companies to easily build and display high-quality interactive 3-D animations of products on their Web sites. Users can spin items around, and zoom in on product details, viewing objects from any perspective with the click of a mouse. Additional examples of software applications are provided throughout the book.
- In the past, all automotive electronic systems platforms were missing the important elements of robustness and flexible software architecture. These characteristics would have enabled large automobile manufacturers to easily tailor their systems for each brand or vehicle program and offer their customers a high degree of personalization. Today, some solutions are available. For example, Ford's Vehicle Consumer Services Interface (VCSI) is an in-vehicle computing platform. It is based on Java and designed to manage effectively traditional vehicle systems and functions.
- The National Science Foundation's TeraGrid *www.teragrid.org* has a massive research computing infrastructure that combines five large computing and data management facilities and supports many academic institutions and research laboratories in their endeavors. The TeraGrid is one of the largest grid-based, high-performance computing (HPC) infrastructures ever created.

T2.3 SYSTEMS SOFTWARE

Systems software is the class of programs that 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 shown in Table T2.1.

Systems software can be grouped into three major functional categories:

- ***System control programs*** are programs that control the use of 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.

TABLE T2.1 Common Operating Systems Tasks

- 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

- ***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.
- ***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 operating system, such as Windows XP, 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.

Mainframes and minicomputers contain only one CPU, but they perform several tasks simultaneously (such as preparation and transfer of results). In such cases, the operating system controls which particular tasks have access to the various resources of the computer. At the same time, the operating system controls the overall flow of information within the computer.

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

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

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

- ***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.
- ***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.

- ***Data management*** is the controlling of the input and output of data as well as their location, storage, and retrieval. Data management programs control the allocation of secondary storage devices, the physical format and cataloging of data storage, and the movement of data between primary storage and secondary storage devices.

A variety of operating systems are in use today. The operating system used on most personal computers is some version of Microsoft's Windows and NT. Many minicomputers use a version of the Unix operating system. Mainframes primarily use the operating systems called *virtual memory system (VMS)* or *multiple virtual system (MVS)*.

Desktop and Notebook Computer Operating Systems. The Windows family is the leading series of desktop operating systems. The **MS-DOS (Microsoft Disk Operating System)** was one of the original operating systems for the IBM PC and its clones. This 16-bit operating system, with its text-based interface, has now been almost totally replaced by GUI operating systems such as Windows 2000 and Windows XP. **Windows 1.0** through **Windows 3.1** (successive versions) were not operating systems, but were operating environments that provided the GUI that operated with, and extended the capabilities of, MS-DOS.

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. Windows 95 is a 32-bit operating system that features multitasking, multithreading, networking, and Internet integration capabilities, including the ability to integrate fax, e-mail, and scheduling programs. Windows 95 also offers plug-and-play capabilities. **Plug-and-play** is a feature that can automate the installation of new hardware by enabling the operating system to recognize new hardware and install the necessary software (called device drivers) automatically.

Subsequent products in the Microsoft Windows operating system are:

- **Windows 98** was not a major upgrade to Windows 95, but did offer minor refinements, bug fixes, and enhancements to Windows 95.
- **Windows Millennium Edition (Windows ME)** is a major update to Windows 95, offering improvements for home computing in the areas of PC reliability, digital media, home networking, and the online experience. (However, it has received many negative comments about its memory management.)
- **Windows NT** is an operating system for high-end desktops, workstations, and servers. It provides the same GUI as Windows 95 and 98, and has more powerful multitasking, multiprocessing, and memory-management capabilities. 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, and is proving popular in networked systems in business organizations.
- **Windows 2000** is a renamed version of Windows NT 5.0. This operating system has added security features, will run on multiple-processor computers, and offers added Internet and intranet functionality.

- **Windows XP** is the first upgrade to Windows 2000 and has three versions: a 32-bit consumer version, a 32-bit business version, and a 64-bit business version. Windows XP is the first version of Windows to support Microsoft's .NET platform (discussed later in the chapter).
- Following Windows XP, Microsoft will release its first fully .NET-enabled Windows operating system, code-named **Blackcomb.** Blackcomb will feature natural interfaces, including speech recognition and handwriting support. **Windows 2003** is the first version of this improved operating system.

UNIX is another operating system that provides many sophisticated desktop features, including multiprocessing and multitasking. UNIX is valuable to business organizations because it can be used on many different sizes of computers (or 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 are promoting Linux for use on the desktop.

Linux is a powerful version of the UNIX operating system that is available to users completely free of charge. It offers multitasking, virtual memory management, and TCP/IP networking. 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 is becoming widely used by Internet service providers (ISPs), the companies that provide Internet connections. The clearinghouse for Linux information on the Internet may be found at *linuxhq.com*.

The **Macintosh operating system (Mac OS X Panther),** for Apple Macintosh microcomputers, is a 32-bit operating system that supports Internet integration, virtual memory management, and AppleTalk networking. Mac OS X features a new user interface (named Aqua), advanced graphics, virtual memory management, and multitasking.

IBM's **OS/2 Warp** is a 32-bit operating system that supports development of e-business applications, accommodates larger applications, allows applications to be run simultaneously, and supports networked multimedia and pen-computing applications.

Mobile Device Operating Systems. Operating systems for mobile devices are designed for a variety of devices, including handheld computers, set-top boxes, subnotebook PCs, mobile telephones, and factory-floor equipment. The mobile device operating system market includes embedded Linux, Microsoft's Windows CE and Pocket PC, Windows Embedded NT 4.0, and Palm OS from Palm. Some mobile device operating systems are described below:

- **Embedded Linux** is a compact form of Linux used in mobile devices. Both IBM and Motorola are developing embedded Linux for mobile devices.
- **Windows CE,** a 32-bit operating system, is Microsoft's information appliance operating system. Windows CE includes scaled-down versions (known as *pocket versions*) of Microsoft Word, Excel, PowerPoint, and Internet Explorer.
- **Pocket PC** is a version of Windows CE 3.0 specifically designed for personal digital assistants and handheld computers.

- **Windows Embedded NT 4.0,** a 32-bit operating system, is aimed at embedded devices that require more operating system capabilities and flexibility than Windows CE can offer.
- The **Palm operating system** was developed by Palm for its PalmPilot handheld PDAs. Palm OS includes a graphical user interface, and users must learn a stylized alphabet, called Graffiti, to make the device receive handwritten input.

Mainframe Operating Systems. Mainframe computers usually require specialized OSs that can handle a large load and that have advanced security features. They are used in entire organizations or large departments, and they are server-based. The major server operating systems include UNIX, Linux, Windows 2000 Server, Windows XP Server, Windows 2003 Server, and Novell NetWare. 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., IBM's OS/390, VM, VSE, and OS/400) generally run on mainframes and midrange systems. Enterprise operating systems offer superior manageability, security, stability, and support for online applications, secure 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.

OS/400 is IBM's operating system for the AS/400 server line, which was renamed *eServer iSeries 400*. IBM's z/Architecture (z/OS), a new 64-bit mainframe operating system, replaces all previous mainframe operating systems. The first system implementing the new architecture is the *eServer zSeries 900*.

Supercomputer Operating Systems. Supercomputer operating systems target the supercomputer hardware market. Examples of these systems include the Cray X1'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 operating systems manage highly parallel multiprocessor and multiuser environments.

Although operating systems are designed to help the user in utilizing the resources of the computer, instructions or commands necessary to accomplish this process are often user-*un*friendly. These commands are not intuitive, and a large amount of time must be spent to master them. Intelligent agents (Chapters 4 and 11) provide some help in this area.

GRAPHICAL USER INTERFACE OPERATING SYSTEMS. The **graphical user interface (GUI)** is a system in which users have direct control of visible objects (such as icons and pointers) and actions that replace complex command syntax. The next generation of GUI technology will incorporate features such as virtual reality, sound and speech, pen and gesture recognition, animation, multimedia, artificial intelligence, and highly portable computers with cellular/wireless communication capabilities. The most well-known GUIs are Microsoft Windows, as described in the previous material.

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 cartoonlike characters, graphics, animation, and voice commands. The cartoonlike characters can be cast as puppets, narrators, guides, inhabitants, avatars (computer-generated humanlike figures), or hosts.

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. 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 the primary storage they need. With virtual memory, all 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 Operating System. 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 operating system* 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.

A popular virtual machine operating system is IBM's VM/ESA. A control program supervises the real machine and keeps track of each virtual machine's operation. The conversational monitoring system (CMS) provides the user with a highly interactive environment coupled with easier access to translators, editors, and debugging tools. Of the newest tools, Java's Virtual Machine is of special interest.

SYSTEM SUPPORTS 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 to 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 clean-up software, like 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.

T2.4 PROGRAMMING LANGUAGES

Programming languages provide the basic building blocks for all systems and application software. Programming languages allow people to tell computers what to do and are the means by which systems are developed. Programming languages are basically 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.

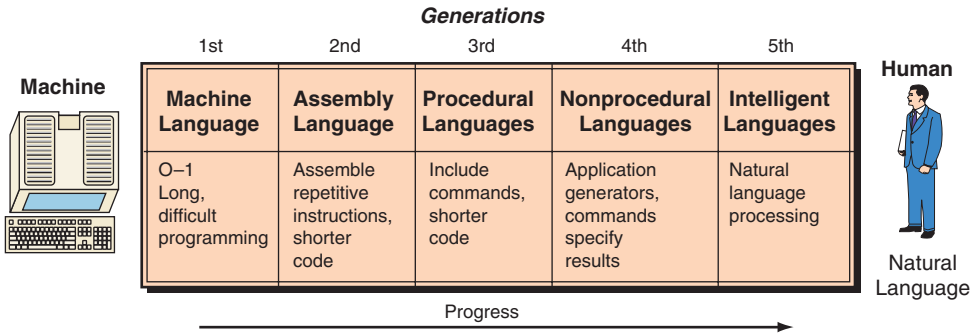

FIGURE T2.4 The evolution of programming languages. With each generation progress is made toward human-like natural language.

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. Languages also differ according to when they were developed; today's languages are more sophisticated than those written in the 1950s and the 1960s.

The Evolution of Programming Languages

The different stages of programming languages over time 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 T2.4 and are discussed next.

MACHINE LANGUAGE: FIRST GENERATION. **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 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 to understand and use by programmers. 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. An **assembly language** is a more user-oriented language that represents instructions and data locations by using mnemonics, or memory aids, which 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.

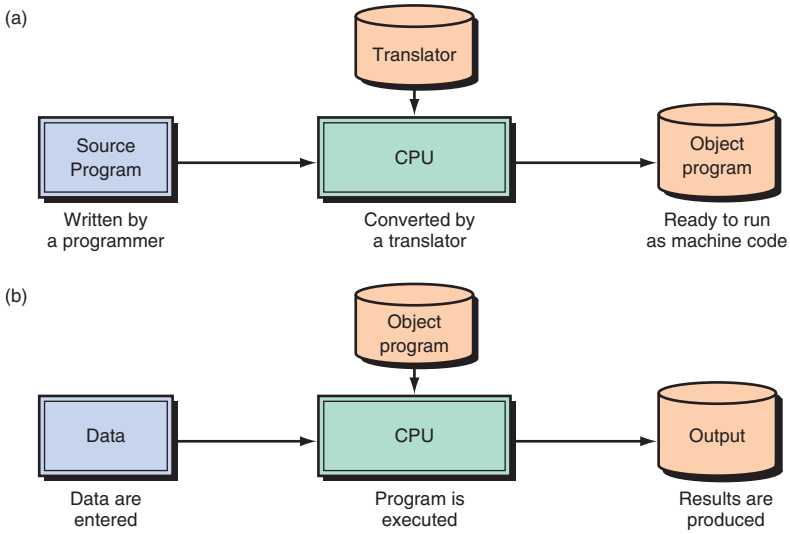

FIGURE T2.5 The language translation process.

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 (see Figure T2.5).

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 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. 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., 0's and 1's), 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 (**For**mula **Tran**slator) is an algebraic, formula-type procedural language. FORTRAN was developed to meet scientific processing requirements.

COBOL (**Co**mmon **B**usiness-**O**riented **L**anguage) 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. Other procedural languages are Pascal, BASIC, APL, RPG, PL/1, Ada, LISP, PROLOG, C#, C++, and Delphi. Some of these are used in object-oriented programming (to be described later).

Interpreted Languages. Java, designed by Sun, is now the most popular language for Web programming. It also uses an interpreter that translates into a machine language, called Bytecode. It is very similar to C, but it does not have the error-prone paint feature.

NONPROCEDURAL LANGUAGES: FOURTH GENERATION. 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. An advantage of nonprocedural languages is that they may be manipulated by nontechnical users to carry out specific functional tasks. 4GLs, also referred to as *command languages*, greatly simplify and accelerate the programming process as well as reduce the number of coding errors.

The term *fourth-generation language* is used to differentiate these languages from machine languages (first generation), assembly languages (second generation), and procedural languages (third generation). For example, *application (or program) generators* are considered to be 4GLs, as are *query* (e.g., MPG's RAMIS), *report generator* (e.g., IBM's RPG), and *data manipulation languages* (e.g., ADABASE's Natural) provided by most *database management systems (DBMSs)*. DBMSs allow users and programmers to interrogate and access computer databases using statements that resemble natural language. Many graphics languages (Powerpoint, Corel Draw, Photoshop, and Flash) are considered 4GLs. Other 4GLs are FOCUS, PowerHouse, Developer/2000, and Visual FoxPro.

NATURAL PROGRAMMING LANGUAGES: FIFTH-GENERATION LANGUAGES. *Natural language programming languages (NLPs)* are the next evolutionary step and are sometimes known as *fifth-generation languages* or intelligent languages.

TABLE T2.2 Language Generation Table

Language Generation	Features				
	Portable (Machine Independent)	Concise (One-to-Many)	Use of Mnemonics & Labels	Procedural	Structured
1st—Machine	no	no	no	yes	yes
2nd—Assembler	no	no	yes	yes	yes
3rd—Procedural	yes	yes	yes	yes	yes
4th—Nonprocedural	yes	yes	yes	no	yes
5th—Natural language	yes	yes	yes	no	no

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. Initial efforts in artificial intelligence in Japan were called the Fifth Generation Project. A comparison of the five generations is shown in Table T2.2.

SIXTH-GENERATION LANGUAGES. Although some people call advanced machine learning languages (see neural computing in Chapter 11) *sixth-generation languages,* there are no current commercial languages that are closer to human or natural languages than NLPs. Some research institutions are working on the concept of such languages, which could be commercialized in the future.

New Programming Languages

Several new languages have been developed in the last 10 to 15 years. These languages were designed to fit new technologies such as multimedia, hypermedia, document management, and the Internet. The major new languages are described next.

OBJECT-ORIENTED PROGRAMMING LANGUAGES. Object-oriented programming (OOP) models a system as a set of cooperating objects. Like structured programming, object-oriented programming tries to manage the behavioral complexity of a system, but it goes beyond structured programming, also trying to manage the information complexity of a system. The object-oriented approach involves programming, operating systems environment, object-oriented databases, and a new way of approaching business applications.

The object-oriented (OO) approach views a computer system as a collection of interacting objects. These objects have certain features, or attributes, and they can exhibit certain behaviors. Further, similar objects in a computer system can be grouped and classified as a specific class of things. The objects in a computer system can interact with each other, and people and objects can interact as well. People interact with objects by sending them messages telling them what to do. Objects also interact by sending each other messages.

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 9 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 check mark 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. The subclass can *override* or *add to* the definitions of the superclass attributes and methods. In other words, subclasses *inherit* all the characteristics of higher-level classes.

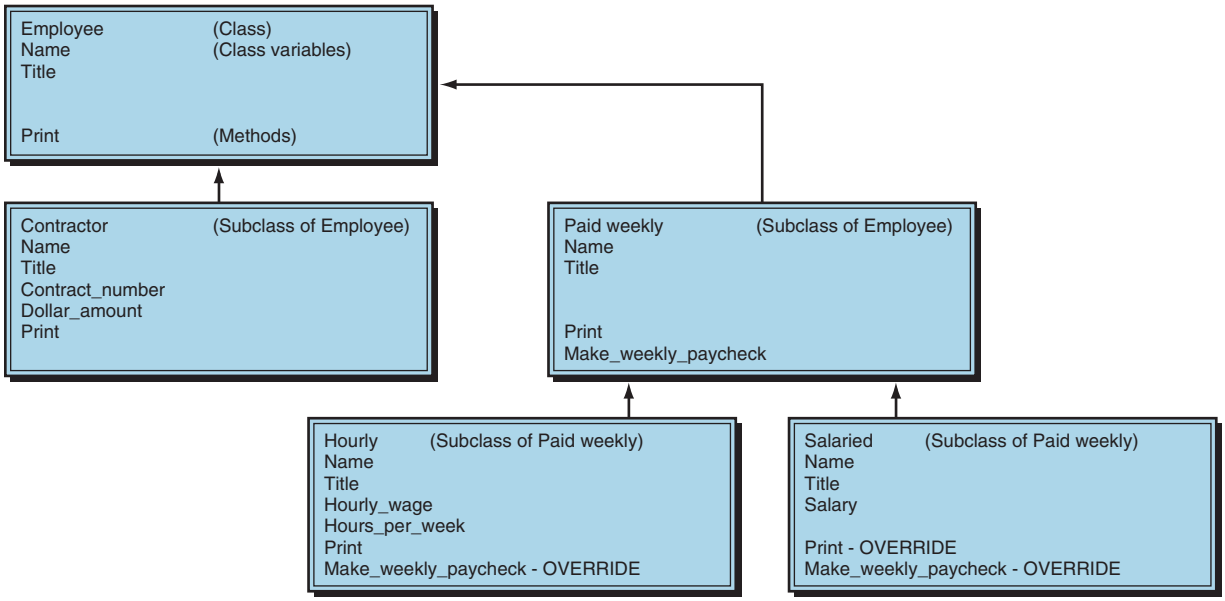

FIGURE T2.6 Object classes, subclasses, inheritance, and overriding. (*Source:* © Courtesy of Apple Corporation. Used with permission. All rights reserved.)

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 T2.6.

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 object-oriented 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++, and Java.

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.

The 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 very easy to use. Web documents are typically written in HTML and are usually named with the suffix ".html." 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 are 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 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. Figure T2.7 compares HTML and 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 the programming language known as 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.

English Text	HTML	XML
MNGT 3070 Introduction to MIS 3 semester hours Professor Smith	<TITLE>Course Number</TITLE> <BODY> <UL> <LI>Introduction to MIS <LI>3 semester hours <LI>Professor Smith </UL></BODY>	<Department and course="MNGT 3070"> <COURSE TITLE>Introduction to MIS<COURSE TITLE> <HOURS UNIT="Semester">3</NUMBER OF HOURS> <INSTRUCTOR>Professor Smith<INSTRUCTOR>

FIGURE T2.7 Comparison of HTML and XML.

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 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.

Other Software

Of the many other existing types of software, we present just a few.

DREAMWEAVER. DreamWeaver is an integrated development environment for developing Web pages. Its good companion is Flash, which can produce sophisticated animated graphics.

VIRTUAL REALITY MODELING LANGUAGE. The **virtual reality modeling language (VRML)** is a file format for describing three-dimensional interactive worlds and objects. It can be used with the Web to create three-dimensional representations of complex scenes such as illustrations, product definitions, and virtual reality presentations. VRML can represent static and animated objects and it can have hyperlinks to other media such as sound, video, and image.

WEB BROWSERS. The major software tool for accessing and working with the Web is the **Web browser.** It includes a point-and-click GUI that is controlled via a mouse or some keyboard keys. Browsers can display various media and they are used also to activate the hyperlinks. Microsoft's Explorer is the major browser.

E-MAIL. E-mail software, such as Qualcomm's Eudora and Microsoft's Outlook Express, allows users to send and receive e-mail messages over the Internet. By using these packages, users can organize and manage their e-mail messages. These packages typically include an address book that stores frequently used e-mail addresses. They also include blockers of unwanted mail and many other features. E-mail software is usually free.

T2.5 SOFTWARE DEVELOPMENT AND CASE TOOLS

Most programming today is done by taking a large process and breaking it down into smaller, more easily comprehended modules. This method is commonly described as *top-down programming*, *stepwise refinement*, or *structured programming*.

Structured programming models a system similar to a layered set of functional modules. These modules are built up in a pyramid-like fashion, with each layer a higher-level view of the system. Even with this approach, however,

many systems have developed severe complexity. Thousands of modules with crosslinks among them are often called "spaghetti code." The ability to break a programming job into smaller parts enables the deployment of special productivity tools, the best known of which is CASE.

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 T2.8). 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 automates one type of

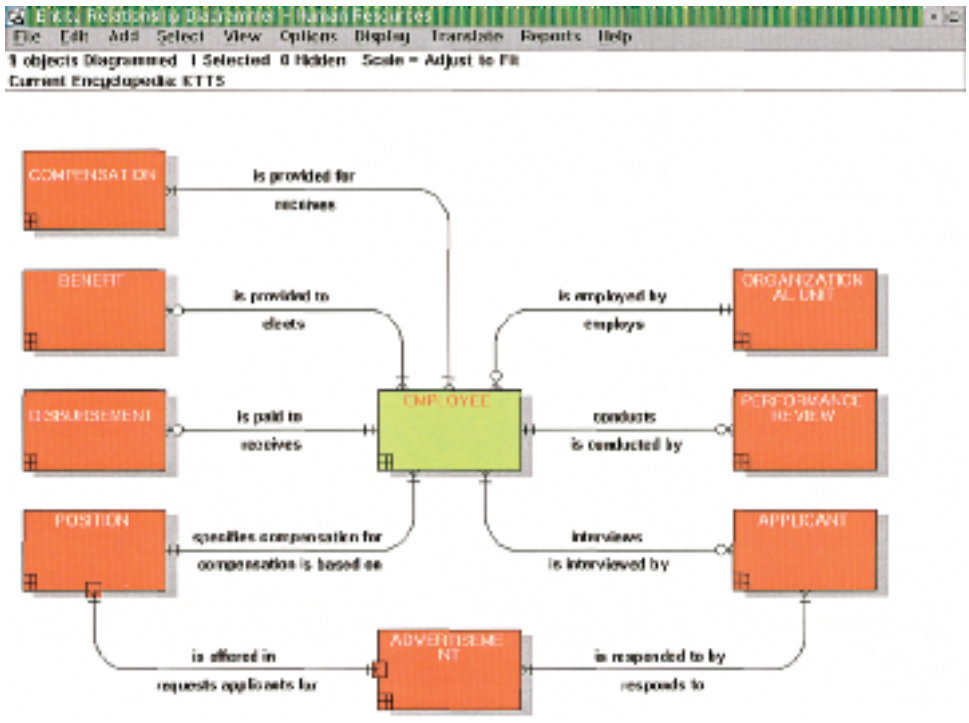

FIGURE T2.8 A CASE display.

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

CASE tools have several advantages:

- CASE improves productivity by helping the analyst understand the problem and how to solve it in an organized manner.
- CASE facilitates joint application and design (JAD) sessions, resulting in better interaction among users and information systems professionals.
- CASE makes it easier to create prototypes, so that users can see what they are going to get at an early stage in the development process.
- CASE makes it easier to make system design changes as circumstances change.

Tasks that are repeated may be automated with CASE tools, for example, drawing dataflow diagrams (a graphical technique for representing the system under development) or drawing system charts. Effectiveness results from forcing the developer to do the task in an organized, consistent manner as dictated by the CASE tool.

Because most CASE tools are graphical in nature and have the ability to produce working prototypes quickly, nontechnically 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.

Broadly speaking, there are two main approaches to systems development—namely, a *structured* approach and an *object-oriented* approach. Similarly, CASE tools have two broad types: One supports a structured approach (e.g., Visio-2000 Systems Architect); the second supports an object-oriented approach (e.g., IBM's Rational Rose, Borland's Together, and Visual Paradigm).

Also, the insistence on one structured method in a CASE program is good for standardization but can be stifling for creativity and flexibility. If an analyst is in an organization that does not use a structured methodology to accompany CASE, then the effectiveness of CASE will be greatly reduced. Creating software often entails imaginative solutions to procedural problems; being constrained to one methodology and the tools included in the CASE package can feel

TABLE T2.3 The Major Tools of CASE

Category	Comments
Analysis and design tools	● Create data flow, entity-relationship, etc. diagrams ● Generic, or specific to proprietary systems design methodologies
Code or application generators	● Some convert specifications directly into code ● Often have drag-and-drop capabilities for developing applications and interfaces
Prototyping tools	● Screen and menu generators ● Report generators/4GLs
Programming language support	● Templates for common code sequences in specific languages ● Subroutine libraries for common functions
Testing tools	● Produce data for testing programs ● Monitor program execution ● Check systems analysis diagrams for completeness and consistency
Problem-tracking tools	● Identify responsibility for fixing bugs and track progress in solving them
Change management/version control tools	● Repository of different versions of code, with "check out" and "check in" capabilities ● Allow access only to authorized personnel ● Maintain information on changes between versions of programs
Project management tools	● Critical path method (PERT charts) ● Gantt charts ● Time and expense tracking
Estimation tools	● Estimate personnel requirements and costs for systems development projects
Documentation generators	● Create flowcharts, other documentation for systems with poor or no documentation
Reverse engineering tools	● Help restructure code in legacy systems
Business process reengineering tools	● Analyze and improve existing processes ● Design new processes

constricting. Finally, CASE tools cannot overcome poor, incomplete, or inconsistent specifications. Popular CASE tools are Oracle's Designer/2000, Seer Technologies' Seer*HPS, and Texas Instruments' Composer. For a comprehensive list of CASE tools see Table T2.3.

T2.6 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 and pestware, open systems, open source software, shareware, componentware, 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. Table T2.4 summarizes these selection

TABLE T2.4 Software Selection Factors

Factor	Considerations
Size and location of user base	Does the proposed software support a few users in a single location? Or can it accommodate large numbers of geographically dispersed users?
Availability of system administration tools	Does the software offer tools that monitor system usage? Does it maintain a list of authorized users and provide the level of security needed?
Costs—initial and subsequent	Is the software affordable, taking into account all costs, including installation, training, and maintenance?
System capabilities	Does the software meet both current and anticipated future needs?
Existing computing environment	Is the software compatible with existing hardware, software, and communications networks?
In-house technical skills	Should the organization develop software applications in-house, purchase off the shelf, or contract software out of house?

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 selection becomes a major issue in systems development and is discussed further in Chapter 14.)

Software 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.

The Software and Information Industry Association (SIIA) enforces software copyright laws in corporations through a set of guidelines. These guidelines state that when IS managers cannot find proof of purchase for software, they should get rid of the software or purchase new licenses for its use. A **license** is permission granted under the law to engage in an activity otherwise unlawful. The SPA audits companies to see that the software used is properly licensed. Fines for improper software are heavy. IS managers are now taking inventory of their software assets to ensure that they have the appropriate number of software licenses.

Although many people do so, copying software is illegal. The Software Publishers Association has stated that software privacy amounts to approximately $15 billion annually. Types of software piracy include: "softlifting"; unrestricted

client access; hard-disk loading; OEM piracy/unbundling; commercial use of noncommercial software; counterfeiting; CD-ROM piracy; Internet piracy; sale of overruns by manufacturing plants; and renting.

Software developers, failing to recoup in sales the money invested to develop their products, are often forced to curtail spending on research and development. Also, smaller software companies may be driven out of business, because they cannot sustain the losses that larger companies can. The end result is that innovation is dampened and consumers suffer. Consumers also pay higher prices to offset the losses caused by software piracy.

Another association that was created to protect the interests of large software developers is the Business Software Alliance (BSA). Any infringer is liable to prosecution by the local government, and any person who gives the information to report such crimes would get a reward from the BASA of up to US$14,000 for each pirated software (*bsa.org*).

As the number of desktop computers continues to increase and businesses continue to decentralize, it becomes more and more difficult for IS managers to manage their software assets. As a result, new firms have sprouted up to specialize in tracking software licenses for a fee. Firms such as ASAP Software, Software Spectrum, and others will track and manage a company's software licenses, to ensure that company's compliance with U.S. copyright laws.

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. In the last 15 years alone, 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.

Malware and Pestware

On many computers one can find software that is running without the knowledge of the computers' owners. Such types of software is known as *malware* or *pestware*. It is installed by vendors who want to find information about you. A

well-known type of such software is *spyware* (see Chapter 15). These types of software use up valuable resources and can slow down your computer. The U.S. government is introducing a bill, known as the Spy Act, to combat the problem. Use of products such as Ad-aware, PestPatrol, or SpySweeper at least once a week will protect your computer by deleting spyware. (You can get such protective software for free.)

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 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 whether it will run on particular hardware. As an example, much Apple MacIntosh application software would not run on Wintel (Windows-Intel) PCs, and vice versa. Neither of these would run on a mainframe.

Certain operating systems, like 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 the GNU (GNU's Not UNIX) suite of software (*gnu.org*) developed by the Free Software Foundation (*fsf.org*); the Linux operating system; Apache Web server (*apache.org*); sendmail SMTP (Send Mail Transport Protocol) e-mail server (*sendmail.org*); the Perl programming language (*perl.com*); the Netscape Mozilla browser (*mozilla.org*); and Sun's StarOffice applications suite (*sun.com*).

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 has been used to create the astounding effects for the movie `Lord of the Rings'. More than 200 workstations and 450 dual-processor servers run 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.

If Linux is to become an enterprise-class operating system, it needs to be developed and tested with enterprise-class machines. The Linux developer community has always had the know-how but not the hardware resources. Open Source Development Lab (OSDL) solves this problem. It provides an independent Linux software development laboratory where developers can create and test applications that run on high-end servers.

Open source code is becoming a corporate building block. Some companies have already taken the steps to transition to use open source software like

Apache Web Server, FastCGI scripting language, FreeBSD or Linux operating system, Zope application server, OpenNMS, Velocity, MySQL, InterBase, PostgreSQL database, Enhydra, Tomcat, and Samba file integration system. Other examples are: Apple's Davinports at 2000; IBM's Derby Cambas's development platform; Mono's development platform; php; Open Office; Firefox; BEA's Beehive; and several other applications (*http://osdir.com/Downloads.phtml*). One reason for this is the new programmers find it very difficult to follow what the previous programmers have done if they do not use open source software. Another reason is outage rate of open source is lower than the proprietary code. Besides, open source code receives enthusiastic cooperation from some of the largest software vendors like IBM and Oracle. In terms of security and stability, open source code is better because many people can search its problem that hidden problems can be eradicated earlier than those of the proprietary code. In addition to this, some entrepreneurs are afraid of being locked in by the proprietary code.

Open source software is produced by vendors but is often produced by groups of volunteers. It is normally distributed for 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 broken this perception rule that has proven this by using open source software; companies can save significant money without compromise on quality, support and future enhancements.

A recent study has concluded that Linux Web servers are not just cheaper to install but also cheaper to run and support. In comparing the total costs of ownership (TCO) of Linux web servers and Microsoft-based web servers, the largest cost component is typically people's time. Linux has lower cost and shorter life cycles for their servers than Microsoft's. Although Linux has not only reached maturity as a web platform, it offers the potential of significant savings. In the area of e-business, open source web software is mature and has even become the de facto solution for many companies. In terms of training costs, open source alternatives win over Microsoft's.

There are positives and negatives of the success of open source software. Positives include quality and reliability, the rapid release schedules of projects and the reduced costs of development and ownership. The negatives are that it is an over-hyped strategy employed by the weak to compete with the strong. In terms of security, open source can enable developers to find the bugs or vulnerabilities in their programs. On the negative side, open source may allow hackers to know about the weaknesses or loopholes of the software more easily than closed-source software.

There is also disagreement from the research firms: IDG found that Linux was growing from strength to strength in Asia but Gartner Group found that Linux shipments to Asia remain very tiny and the little growth rate cannot threaten Microsoft's dominance.

Openness has taken a great stride forward. W3C has recently issued a new draft of its patented policy recommending that patented technologies be allowed only in Web standards when royalty-free. On the other hand, Microsoft announced that they would document and allow free use of its Windows 2000 Kerberos extensions. Sun has also taken similar step by undergoing a major

revision on the agreement on how third parties must implement Java standards.

Shareware and Freeware

Shareware is software where 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, KaZaa, and Ad-ware.

Usually, free software has never been better or more abundant while makers of free applications are alluring users toward paid versions of the software, which has many added features. However, features of some free software are found sufficient to cater to simple office work. Note that free software may be less secure and easy to be attacked by viruses and spyware. For further discussion, see *pcworld.com* (March 2002, p. 87).

Componentware

Componentware is a term used to describe a component-based software development approach. **Software components** are the "building blocks" of applications. They provide the operations that can be used by the application (or other applications) again and again. Any given application may contain hundreds of components, each providing specific business logic or user-interface functionality. Consider a database application as an example: The data-entry screen may contain several user-interface components for providing buttons, menus, list boxes, and so forth. There may also be business logic components to perform validation or calculations on the data, as well as components to write the data to the database. Finally, there can be components to create reports from the data, either for viewing in an on-screen chart or for printing. Component-based applications enable software developers to "snap together" applications by mixing and matching prefabricated plug-and-play software components.

Software Piracy

As discussed earlier, the issue of software piracy is critical to the advancement of software and the ability to innovate and improve software. According to the Business Software Alliance, the damage to the industry from illegal copying of software is about $20 billion a year. For a discussion, see Chapter 16.

Services-Oriented Architecture (SOA)

Services-oriented architecture (SOA) is a framework for constructing and interlinking a company's back-end systems in order to make the computing systems more flexible and cost-effective. SOA communications, enabled by Web Services, are different from existing middleware. Under SOA and Web Services, applications automatically link to one another as needed, which is the concept of "loose coupling." (See Technology Guide 6 for further discussion.)

Autonomic Computing

As systems become more interconnected and diverse, systems architects are less able to anticipate and design interactions among components. **Autonomic computing** refers to computing systems that can manage themselves given high-level objectives from administrators. It gets its name from the autonomic nervous system that governs our heart rate and body temperature, thus freeing

our conscious brain from the burden of dealing with these and many other low-level functions. An autonomic computing system consists of myriad interacting, self-governing components that in turn comprise large numbers of interacting, autonomous, self-governing components at the next level down.

REFERENCES

Babcock, C., "New Tools Unleash Power of Linux Clusters," *Interactive Week*, November 6, 2000.

Babcock, C., "Open Source Code Is Becoming a Corporate Building Block," *Interactive Week*, May 14, 2001.

Barker, J., *Beginning Java Objects: From Concepts to Code*. Birmingham: Wrox Press, 2000.

Berr, J., "AOL, Microsoft, Yahoo Band to Block Spam," *Honolulu Advertiser*, April 29, 2003.

"Best-Practice Case Studies—UPS for Parcel Shipment and Tracking," *mobileinfo.com/Case_Study/ups.htm*, 2001.

Booker, E., "Better Java," *Internet Week*, February 16, 1998.

Burns, J., "So, You Want ASP, Huh?" *Datamation*, September 24, 1998.

Burns, J., "Java vs. JavaScript: So… What Is the Difference Between Java and JavaScript Anyway?" *Datamation*, January 3, 2000.

Cortese, A., et al., "The Software Revolutions," *Business Week*, December 4, 1995.

Courter, G., and A. Marquis, "Mastering Microsoft Office 2000: Premium Edition," *Sybex*, July 1999.

Dahl, E., "Fee vs/. Free Software," *PCWorld*, March 2002.

Dallas, D.A., "Information Systems Executive Journal," *Linux: A Cost-Effective Alternative to Windows*, Spring 2002.

Deckmyn, D., and J. Vijayan, "Linux Applications Make Leap to Unix," *Computerworld*, August 21, 2000.

Dick, K., *XML: A Manager's Guide*. Reading, MA: Addison-Wesley, 2000.

Doke, E. R., and B. C. Hardgrave, *An Introduction to Object COBOL*. New York: Wiley, 1998.

Dorfman, M., and R. H. Thayer, *Software Engineering*. Los Alamits, CA: IEEE Computer Society Press, 1996.

Fayad, M., and M. P. Cline, "Aspects of Software Adaptability," *Communications of the ACM*, No. 10, October 1996.

Feibus, A., "A Garden of Visual Delights," *Information Week*, July 1, 1996.

Foley, J., "Information Week," *Internet Week*, Mar 13, 2003.

Gaudin, S., "ActiveX," *ComputerWorld*, August 10, 1998.

Grimes, B., "Linux Goes to the Movies," *PC Magazine*, May 27, 2003.

Grossman, L., "The Browser That Roared," *Time*, May 13, 2002.

Katz, H., ed., *Technology Forecast: 1998* (also 1999, 2000). Menlo Park, CA: Price Waterhouse World Technology Center, 1998, 1999, 2000.

Kephart, J., and D. M. Chess, "The Vision of Autonomic Computing," *IEEE*, January 2003.

Korson, T., and V. Vaishnave, "Managing Emerging Software Technologies: A Technology Transfer Framework," *Communications of the ACM*, Vol. 35, No. 9, September 1992.

LaMonica, M., "Services-Oriented Architecture Gains Support," *CNET News.com*, April 1, 2004.

Leganza, G., "Top-Down versus Bottom-Up: Approaches to Enterprise Architecture," *www.forrester.com*, March 29, 2004.

Lewin, J., "Linux Cheaper than Windows for Web Serving," *ITWorld.com*.

McDonald, A. B.,"The Next Best Thing to Being There," *PCWorld*, April 2002.

"Occasional Maintenance," *PCWorld*, May 2002.

"Open Source Software: Investigating the Software Engineering, Psychosocial, And Economic Issues," *Information Systems Journal*, 2001, pp. 273–276.

PCWorld.com, "Features Comparisons—Free and Paid Software," March 2002: p. 87.

Pont, M. J., *Software Engineering with C++ and CASE Tools*. Reading, MA: Addison-Wesley, 1996.

Reed, D. A., "Grids, the TeraGrid, and Beyond," *IEEE*, 2003.

Rupley, S., "Apple's Next Moves," *PC Magazine*, June 30, 2002.

Shelly, G. B., et al., *Microsoft Windows 2000: Complete Concepts and Techniques*. Cambridge, MA: Course Technology Inc., 2000.

Simonds, C., "Software for the Next-Generation Automobile," *IEEE*, November/December 2003.

Smith, S., "The Whiteboard Goes Digital," *LAPTOP*, May 2003.

Spanbauer, S., "Linux Bulks Up," *Business2.com*, November 28, 2000.

"3-D Technology Gives HMV.com Shoppers `Sneak Peek' into New Music Releases," *Stores* October 2001.

Test Center, "Sophisticated Simplicity–Oracle Database 10g Stresses Easier Administration," *Infoworld*, March 22, 2004.

Titchenell, D., *Getting Started with HTML*. Fort Worth,TX: Dryden Press, 1999.

von Hippel, E., "Learning from Open-Source Software," *MIT SLOAN Management Review*, Summer 2001.

Wallace, N., "ActiveX Template Library Development With Visual C++ 6.0," *Wordware Publishing*, May 1999.

"Welcome Steps Toward Openness," *eWeek*, March 18, 2002.

Welsh, M., et al., "Running Linux," O'Reilly & Associates, August 1999.

Wills, C., "Firms Scheme to Phase Out Bios, with Battle to Boot," *Technology Post*, February 25, 2003.