CHAPTER 2

AN INTRODUCTION TO SYSTEM CONCEPTS AND SYSTEMS ARCHITECTURE

"Now, this is just a simulation of what the blocks will look like once they're assembled."

C. Covert Darbyshire/The Cartoon Bank

2.0 INTRODUCTION

In this book we discuss systems: computer systems, operating systems, file systems, I/O (sub)systems, network systems, and more. Each of these same systems is also an element with a major role in the information technology systems that form the backbone of modern organizations. Indeed, these elements—computer hardware, software, data, and communication—together represent the infrastructure of every IT system. If we are to understand the various types of systems that are the focus of this book, it is important that we first understand the concept of "system" itself, and, then, equally important, the basic architectures of the IT systems that use these elements. Only then is it possible to see clearly the role of the various system elements in the larger IT picture as we visit each in turn.

Use of the word "system" is obviously not unique to IT. In our daily lives, too, we often use the word "system" to describe things in everyday language. Our homes have electrical systems, plumbing systems, heating and air conditioning systems, and maybe for some, even, home theatre systems. There are ignition, braking, fuel, exhaust, and electrical systems in our cars. Our cities have water systems, sewer systems, and transportation systems, to name a few. Philosophers and social scientists talk about social systems and linguistic systems. The economy deals with banking systems, financial systems and trading systems, and, for that matter, economic systems. The word "system" even appears in the names of thousands of companies.

So it seems as though everyone knows what a system is, but what is a system? We use the word "system" intuitively, without thinking about the meaning of the word, so we obviously have an intuitive understanding of what a system is. IT professionals, however, spend their careers analyzing, designing, developing, implementing, upgrading, maintaining, administering, and using systems everyday. It is therefore important that we have a deeper, more formal understanding of system concepts.

In this chapter, we consider the concept of a system from an IT perspective. We investigate the characteristics and composition of systems, explain the meaning of system architecture, and show the fundamental role of systems, particularly various types of IT systems, in business. We offer examples of different types of IT systems, and show how IT systems work together to accomplish tasks and solve problems. We show how systems can themselves be composed of subsystems, where the subsystems also fit the definition of systems.

After you have studied this chapter, you should have a clear understanding of what a system is, what kinds of systems are used in IT, the purpose and goals for each of these systems, and how these systems fit together and interact with each other and with their environment. You'll understand the concept of system architecture. This discussion will set the stage for the remainder of the book, which considers individually and collectively the specific computer-based systems and subsystems that constitute the primary tools and components of business information technology.

2.1 THE GENERAL CONCEPT OF SYSTEMS

The most important characteristic that is shared by all of the systems mentioned above, and, indeed, by all systems, is that each is built up from a set of components that are linked together to form what we think of as a single unit. The house plumbing system, for example, consists of sinks, faucets, toilets, a hot water heater, bathtubs or showers, valves, and more, all connected together by pipes. An IT system consists of groups of computer hardware, various I/O devices, and application and system software, connected together by networks.

Often, the system is intended to serve a purpose or to produce results. The purpose of the house plumbing system is to allow the residents of the home access to water to wash, bathe, and drink. The purpose of an IT system is to allow organizations to process, access, and share information. The results of a successful IT system are documents, information, improved business processes and productivity, profits, strategic plans, and the like. This is, in fact, the "output" of the IPO model described in Chapter 1. In general, though, there is no requirement that a system serve a specific, definable purpose. The fact that the set of components may be considered as a single unit is sufficient to satisfy the concept of a system. The solar system is an example of a system where the purpose is unspecified.

There is also no requirement that the components of a system be physical. The links between components can also be physical or conceptual. In fact, the system itself may be conceptual, rather than physical. The number system is an example of a conceptual system. Computer operating systems are also conceptual, rather than physical. Business systems are also conceptual, although some of the components that they contain may be physical. The words *tangible* and *intangible* are sometimes used in place of physical and conceptual, respectively. Intangible or conceptual components and systems include ideas, methods, principles and policies, processes, software, and other abstractions. If, for example, the components in a system represent steps (intangible) in a multistep process, the links may represent the need to complete one step before the next is begun (also intangible).

Figure 2.1 illustrates a number of different systems to show you some of the possibilities. Figure 2.1(a) is a model of a home plumbing system. This is a physical system. The components are plumbing fixtures, linked by pipes. Figure 2.1(b) is a simplified representation of the solar system. The sun and planets are physical; the links in this system are conceptual, specifically, the distance of each planet from the sun, interplanetary and solar gravity, orbital relationships, the distances between planets at a particular point in time, and other attributes. Figure 2.1(c) is a diagram of a home networking system. The links in this case are a mixture of physical wires and (intangible) wireless connections. Sometimes the nature of the links is important only in terms of providing the proper interface connections to the components. Figure 2.1(d) is a simplified diagram of part of the inventory control portion of a sales system. The relationships between the components in this case are temporal (i.e., related to time). For example, inventory from a previous sale must be deducted from stock before we process the next order; otherwise we can't promise delivery of goods on the new order because we don't know if we still have sufficient goods in stock to fill the order.

With these pictures and ideas about systems in mind, we will define a **system** as follows:

> A system is a collection of components linked together and organized in such a way as to be recognizable as a single unit.

FIGURE 2.1(a)

Plumbing System Diagram

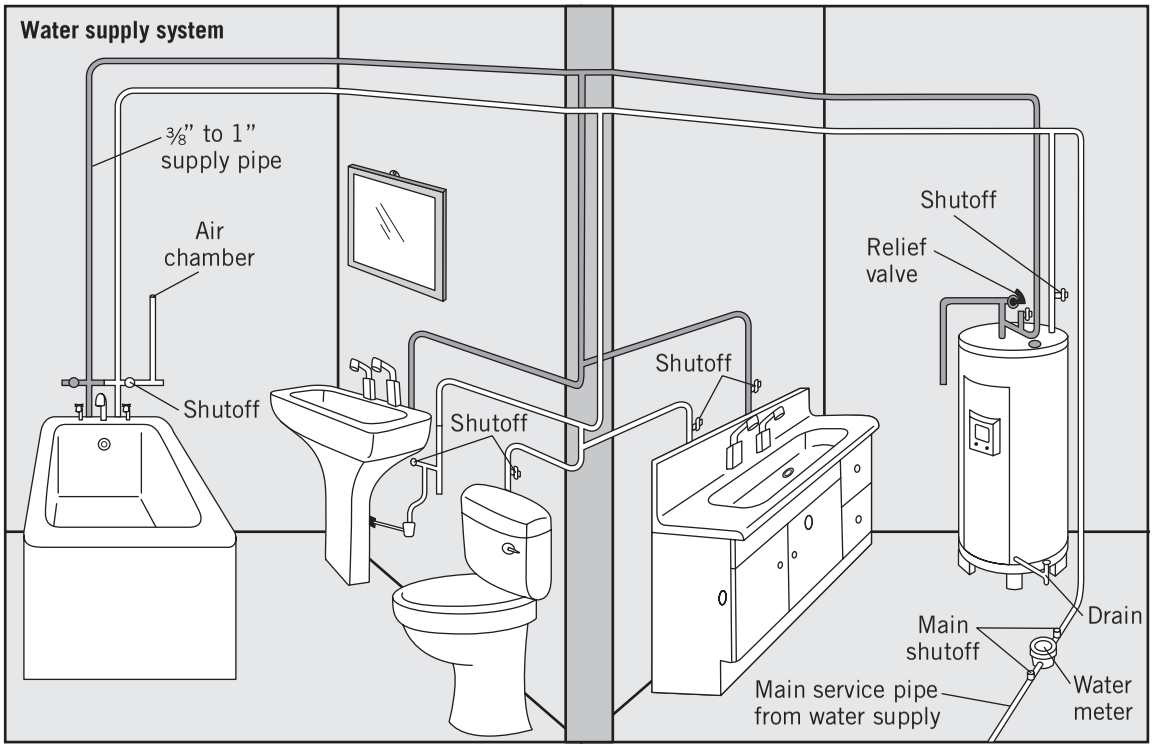


FIGURE 2.1(b)

The Solar System

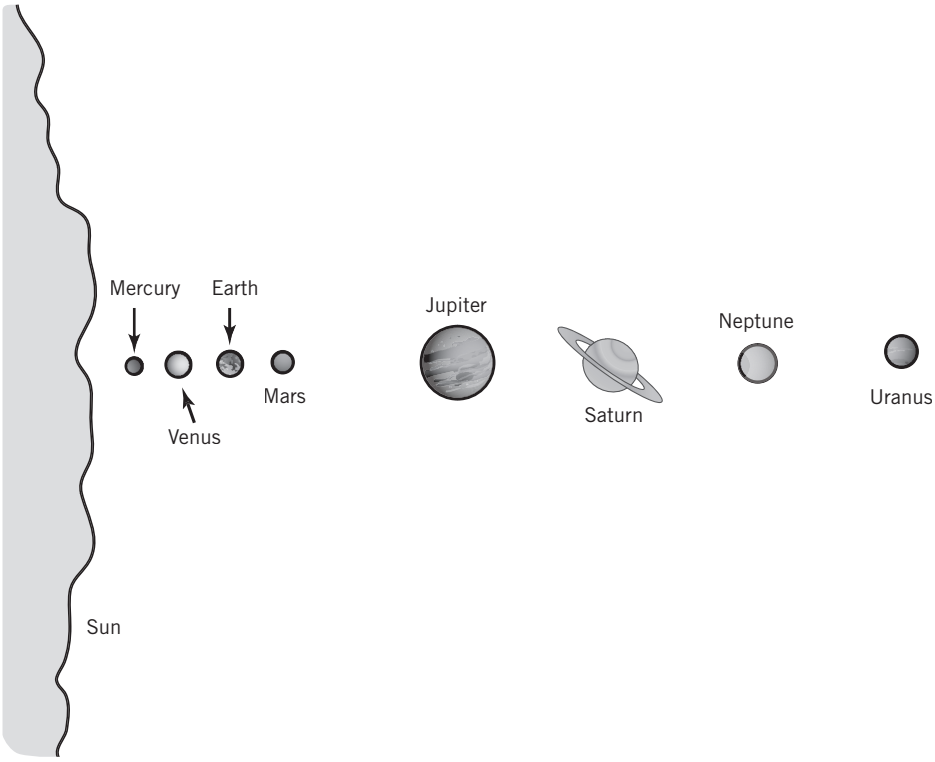


FIGURE 2.1(c)

A Typical Home Network System

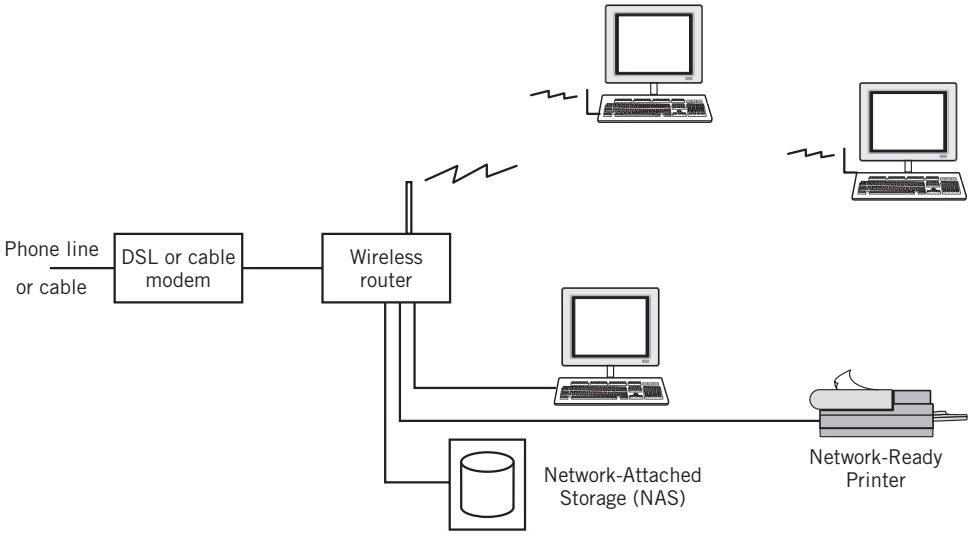
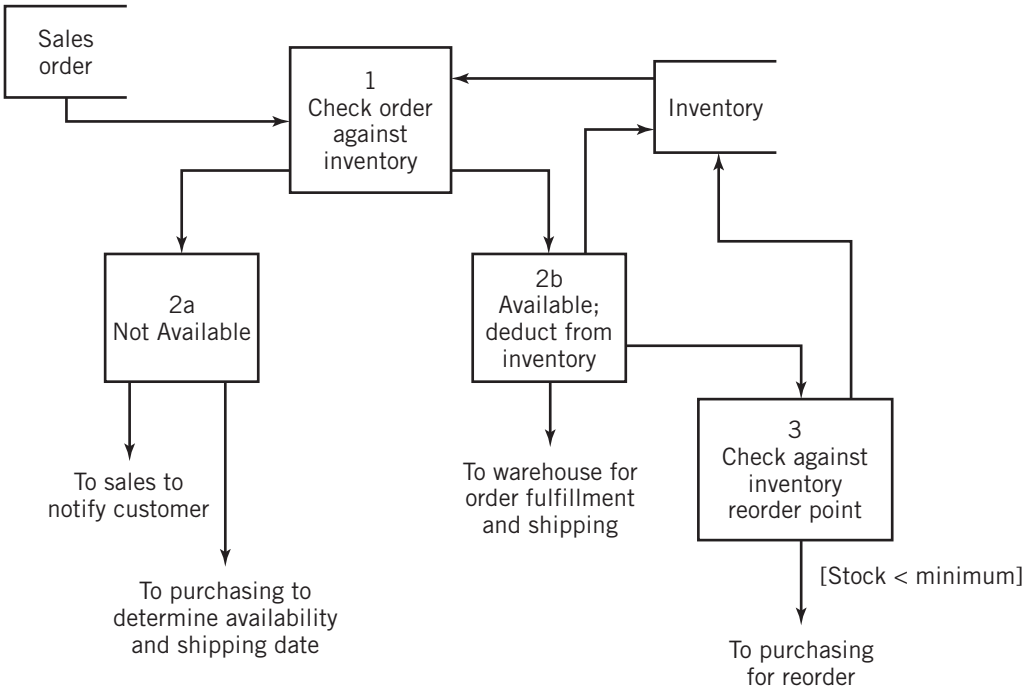


FIGURE 2.1(d)

Flow Diagram for Part of an Inventory Control System



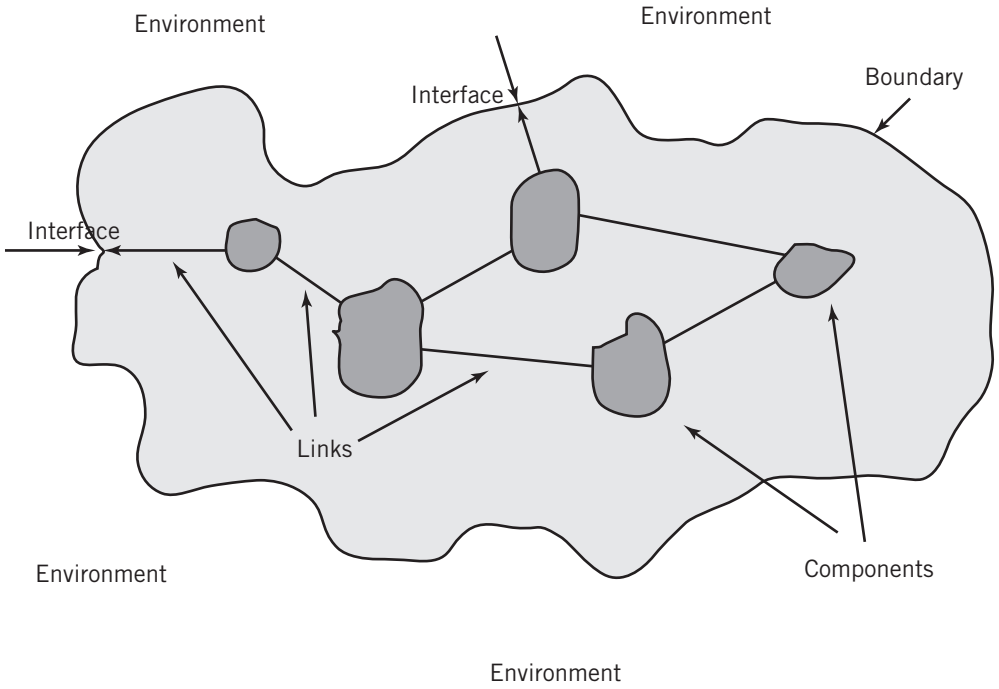
A general representation of a system is shown in Figure 2.2.

The linked components that constitute a system also define a boundary for the system. Anything outside the boundary repesents the **environment** that the system operates or presents itself within. The environment may interact with and affect the system in various ways. The reverse is also true. The **interface** between the system and its environment is an important characteristic of the system. If the interface is well-defined, it is often possible to replace the existing system with a different system, as long as the interface between the system and the environment remains constant. This idea can have important implications when designing IT systems. For example, in a particular IT installation, a single large computer may be functionally the same as a network of small computers. When we define inputs and outputs for a system, the environment is the source of the input and also the receiver of the output.

As an example of the relationship between a system and its environment, consider the rather simplistic view of an e-business system illustrated in Figure 2.3. The organization represented by this illustration purchases goods from suppliers and makes them available for sale. (The value-adding component in the figure consists of various operations that make it worthwhile to buy from this organization, rather than directly from the supplier. For example, Amazon.com makes it possible to buy a wide variety of books from one source, rather than having to place separate orders from a number of different suppliers.) The environment for this system consists of customers who purchase from the system, suppliers

FIGURE 2.2

General Representation of a System

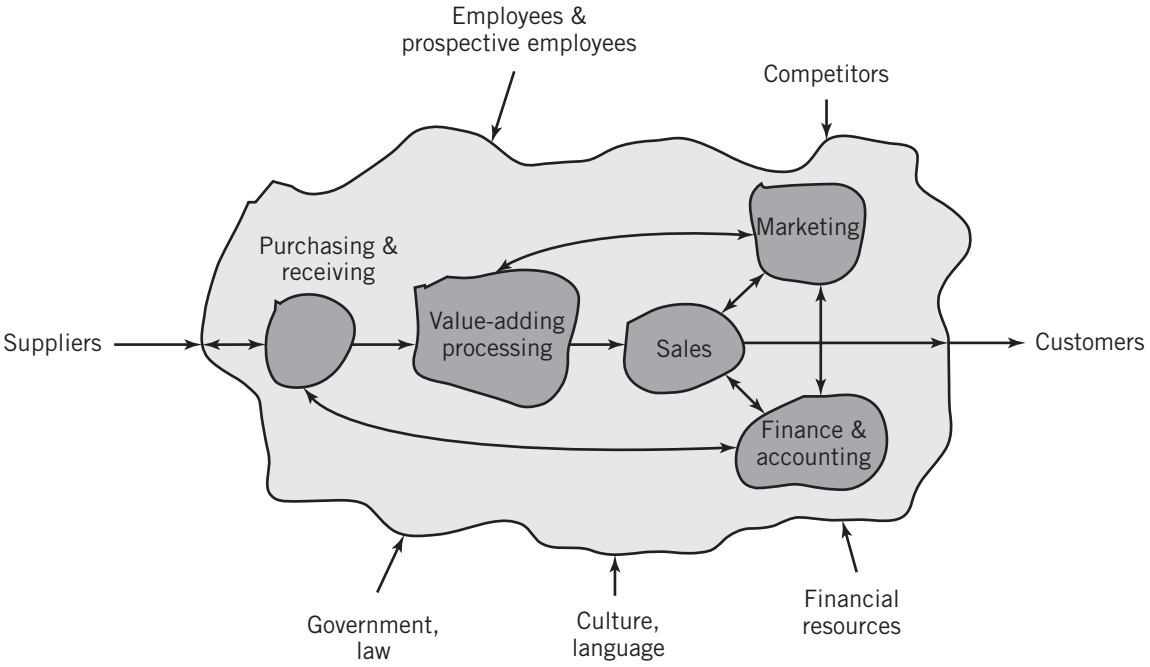


to the system, governments who control the legal aspects of the business and collect taxes, employees and prospective employees, external support personnel (such as repair people), financial resources, and others. The primary interfaces for this system are system input from suppliers and system output to purchasers; however, there are additional, more subtle interfaces to be considered, including legal, cultural, and financial interactions with the system. For example, sensitive cultural and language issues that offend potential customers on a website migh have an important impact on an organization's sales.

When analyzing a system, the components of the system may be treated as irreducible or they may themselves be representable as systems. When considered in the context of a particular system, these components would be viewed more accurately as **subsystems**. A business IT system, for example, might have marketing, manufacturing, purchasing, inventory, finance, and accounting subsystems, among others. Even these components might be expanded. The marketing subsystem might be further broken down into sales, development, and advertising components, as one possibility. The level of detail to be considered depends on the context in which the system is being considered, discussed, evaluated, or used. The division of a system or subsystem into its components and linkages is called **decomposition**. Decomposition is inherently hierarchical. The ability to decompose a system hierarchically into subsequent sets of components and subsytems is an important property of systems.

FIGURE 2.3

A Simple E-Business System



The fundamental properties, and the patterns of relationships, connections, constraints, and linkages among the components and between the system and its environment are known collectively as the **architecture** of the system. Some people choose to differentiate the *architecture* of a system from the *organization* of a system. The assumption is that the architecture is fundamental to the meaning and value of the system whereas the organization is one of possibly many combinations of components and linkages that meets the requirements of the architecture. The difference is subtle and often unimportant.

It is common to represent systems and their components by models or drawings on paper or objects within a computer program. These representations are **abstractions**. They *represent* the real system but are *not actually* the real system. (For example, the solar system does not fit conveniently inside a computer!) It should be obvious to you that all of the illustrations of systems in Figures 2.1, 2.2, and 2.3 are abstractions.

The primary reason for humans to group components into systems and to represent them as abstractions is to simplify understanding and analysis, particularly if the individual components are numerous and complex. We can study the relationships between the different components without the distraction created by the details of individual components. We can decompose, isolate and study individual components when required. We can study the interactions between the environment and the system as a whole. Effectively, our analyses are simplified by eliminating factors that are not relevant in the context of our interests. In a large network of computers, for example, we may be concerned primarily

with the flow of data between computers. The details of the individual computers are unimportant. In general, dealing with models at the system level allows us to isolate and focus on the specific elements of interest more easily, by treating other elements collectively.

To escape our fixation on information technology systems for an instant, consider, just for fun, the solar system that we've used previously as an example. If we are studying the Milky Way galaxy, it is convenient and sufficient to treat the solar system as a single irreducible component in the galaxy. We might be interested in the location and movement of our Sun in the galaxy, for example, but the structure of the planets is irrelevant to our study in this case. On the other hand, if we are interested in studying the effects of the tides on a sea shore where we are planning to vacation, we will have to expand the ''Earth'' component and look at the specific effects of the moon and other nearby objects as part of our analysis.

Consider, too, the role played by decomposition and the ability to isolate and study individual components. A complex system may be divided into relatively independent components and analyzed by different individuals, each a specialist in their own area. Thus a plumber can create a home water system component without concern for the details of the electrician's efforts. They can work together on the linkages that concern both of them, for example, the wiring for the boiler in a hot water heating system. The system architect coordinates the different efforts. The role of an IT system architect is similar: to work with finance experts on the finance component, marketing experts on the marketing component, and so forth.

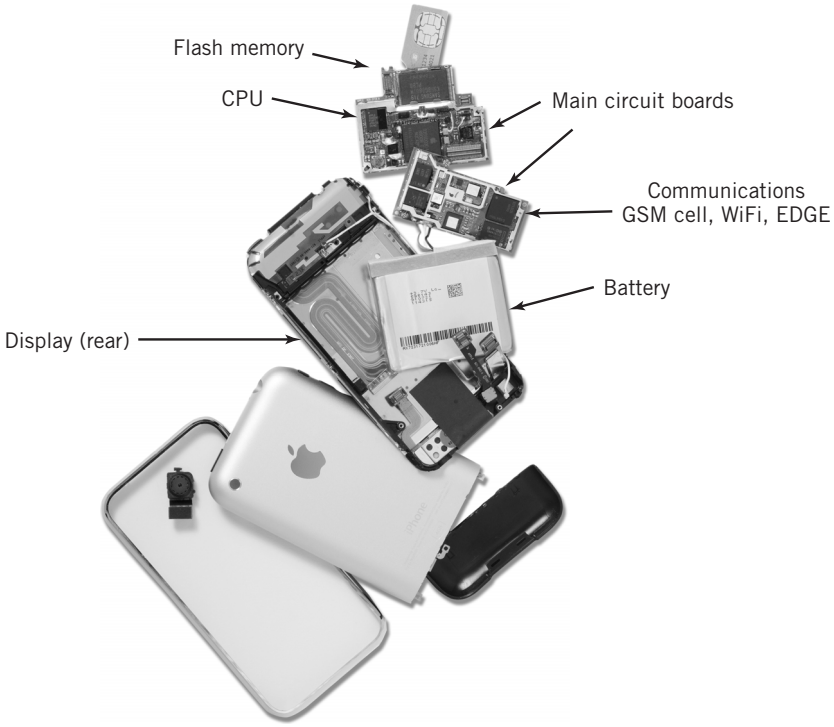
When the goal of a project is to implement a system of some type, it is sometimes convenient to view the components of a system as modules that can be implemented independently, then linked together to produce the final result. This technique can simplify analysis, design, assembly, upgrading, and even repair. It also supports collaboration during the design process, since individual components can be designed by different individuals using specifications for the interfaces.

For example, a cell phone might consist of a computer control module, a memory module, a display module, an audio input/output module, a radio transmitter/receiver module, a keypad/text input module, and a wireless networking module. Each component might have been developed by a different team. These modules, designed, constructed, and manufactured as individual assemblies, properly interfaced, wired together, and mounted into a case, constitute the design of a typical cell phone. They also represent the components that might appear in the system diagram for a cell phone. The same approach might be taken with a computer system, with a central processor module, a graphics display module, an audio module, a network module, a hard drive controller module, and so on. Figure 2.4, for example, shows the basic system hardware components that make up an iPhone.

It is also important to realize that there may be many different representations for a system, to reflect the various uses of the system model. Returning to our IT roots for an example, the representation of the business system shown in Figure 2.5(a) is a traditional hierarchical oranization chart. The components are departments that perform various functions within the business. In contrast, a partial model of the same business shown in Figure 2.5(b) represents the application architecture of an IT system within this business. Take another look at Figure 1.4 for still another representation of a business. As another simple example, you could represent a house by the physical appearance of its exterior, by the function and layout of its rooms, or by the various subsystems, electrical, plumbing, heating, and so on that the house requires. Presumably, each of these representations would be useful to a different participant. In fact, we would expect an architect to provide all of these for use by the owner, the builder, and the various contractors.

FIGURE 2.4

iPhone Components



Courtesy Christopher Harting.

FIGURE 2.5(a)

Business Organization Chart

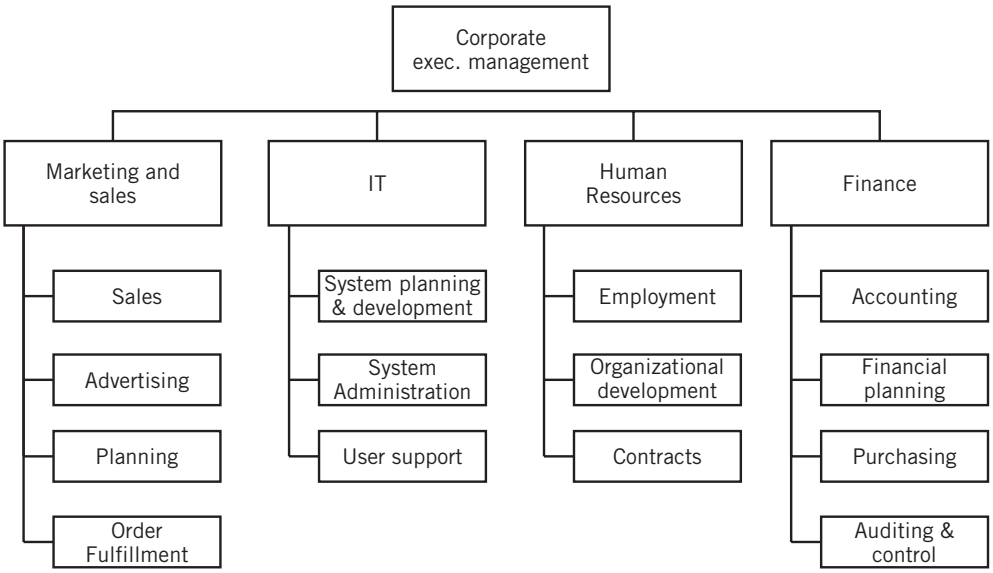
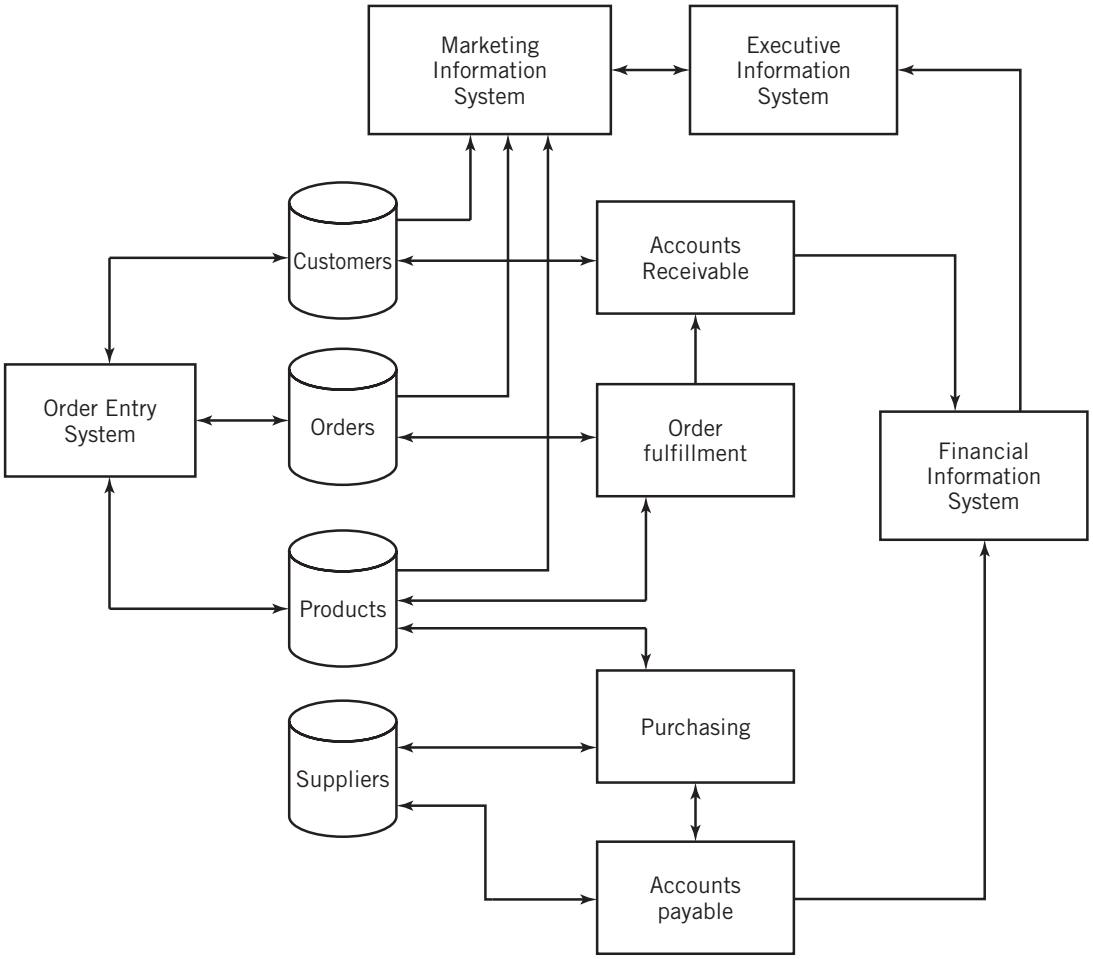


FIGURE 2.5(b)

Partial View of a Business Application Architecture



2.2 IT SYSTEM ARCHITECTURES

The use of system concepts is particularly applicable when discussing the various types of IT systems. In general, the goal of IT systems is to assist organizations to meet the strategic needs of the enterprise. Not surprisingly, IT systems are frequently complex, and the ability to separate them naturally into subsystems or components of manageable size simplifies understanding of the system as a whole. The analysis, design, and implementation of IT systems must take place at different levels of detail and frequently require collaboration among many analysts and designers. This corresponds well with the ability to decompose systems into components, hierarchically, which allows us to concentrate at the appropriate

levels of detail during each step along the way. This approach is known as a **top-down approach**. The top-down approach allows us to focus on the specific areas of interest without the distraction of details that are irrelevant for the level that we're studying. In this way, a system architect can analyze and study the IT system as a whole, encapsulating the computer systems, software systems, network architecture, and Web architecture that represent components, and focusing instead on the large picture: the purpose of each component and the requirements for the interfaces and linkages that connect and integrate them. With the IT system architecture firmly established, we can consider the individual business functions, computer systems, and networks that will link them together. For IT system analysis, this is often sufficient, at least superficially, assuming that the system architects actually understand the conditions and constraints imposed by details at the lower levels.

Although there are other, equally valid, approaches to IT system analysis and design, and many other important considerations as well, this approach suits the purposes of this book well because it allows us to establish general requirements for IT systems and then to show how the specific capabilities and characteristics of computer hardware, operating systems, networks, and data fulfill those requirements.

With these ideas in mind, let us return to the simple word processing example from Chapter 1 and reconsider it from a system architecture perspective. Recall that in this example you are sitting at your computer typing text into a word processor. We noted that the computer accepted input from your mouse and keyboard, processed it according to rules established by the application software, and produced output, which appeared on a display. From the system perspective, we can, for now, treat the whole computer, keyboard, display, printer, storage, software, and all as a single component. You're the relevant part of the environment for this discussion. Forgetting the issue of control for now, the system has an input and an output. Both of these interface with you, the environment. The data for this interface is alphanumeric text in human-readable form. Other input data to the document might include graphics drawn with the mouse, photographic images from a digital camera, bar codes, or music from an iPod or other audio source. We described this scenario earlier, in Chapter 1, as input-process-output.

A system this simple is unlikely to meet all the needs of even the smallest enterprise or, even, the least computer-literate individual. But it does serve as a starting point to recognizing the value of a system approach to the understanding of information technology.

Distributed Processing Systems

Realistically, modern IT system architectures generally rely on multiple computers connected by networks of communication channels to achieve their goals. In all but the smallest organizations, input data is collected from locations scattered throughout the organization, stored, processed, and distributed to other locations within the organization. Since modern computer hardware and networking equipment is plentiful and inexpensive, it is practical to distribute computing capability to everyone who needs it. Furthermore, the availability of the Internet and alternative structures, such as satellite communications, make global data communication practical. Web access, organization intranets, e-mail capability, analysis tools, such as Microsoft Excel, and document preparation tools are widely available and are considered essential business tools throughout most organizations. Collaboration

between different organizations, particularly in the area of automated business-to-business purchasing and sales, is commonplace.

Therefore, when envisioning effective IT systems, designers typically must create system architectures that are capable of supporting large numbers of user workstations who will have ready access to the organizational information that they need. The system must be able to reliably store and protect large amounts of organizational data. For many organizations, customers outside of the organization may also need access to the system to get information and to make purchases.

Consider a few typical simple scenarios:

- A global fast food chain collects data each day from each of its restaurants worldwide to establish sales figures and determine sales trends. This allows the company to determine which locations are most productive and which locations need assistance, which items sell best and which need to be modified or replaced, and so on.
- A large travel organization conducts much of its business online, using travel agents located all over the world. It maintains Web servers that have immediate access to large databases of client information and travel information, as well as continual and instant access to airline and hotel reservation systems to determine current airfares, seat availability, and hotel room availability. All of this information must be immediately accessible to every agent and must be current at every instant. Even brief system failures are very costly.
- A large Web-based retail sales organization sells large quantities of a wide variety of merchandise. (Think Amazon or Wal-Mart.) Orders initially come in to a central facility, where they are billed. Inventory is stored in warehouses in various countries and local regional areas to expedite delivery and reduce delivery costs. The system must be able to distribute orders to the various regional facilities efficiently; it must also maintain appropriate levels of goods at each warehouse to match sales and must be able to locate goods and arrange shipping in response to orders as they come in.

  Inventory replenishment is handled by an automated purchasing IT system component that is integrated with the IT systems of the suppliers. Purchase order data is passed from the retailer to a supplier, which triggers order placement, billing and shipment components in the supplier's systems. Web technology is commonly used to satisfy the need for data and communication compatability between the systems.
- Even conventional business order processing is inherently distributed within an organization. A purchase order, for example, might be entered into the system by a salesperson in the sales department; the order is checked by order fulfillment for inventory, then distributed to the accounting department for a credit check and billing, and sent to the warehousing area for packaging and shipping. Back orders and inventory replenishment are sent to the purchasing department. For planning and marketing purposes, data will be collected into a central location and processed into sales figures, inventory planning and purchase requirements data, and the like. In a large organization, these functions might be widely scattered over a city, country, or even the world.

The emphasis in each of these scenarios is the flow and processing of data within an organization or between organizations or between an organization and its environment. The system architecture representation of such operations is called **application architecture**. Application architecture is primarily concerned with the activities and processing of application programs and the communications between them. Since the application architecture addresses the fundamental business needs of the organization, the application architecture is typically the primary consideration in IT system design. Therefore, the system requirements and constraints set by the application architecture have major impact on the hardware architecture and network architecture requirements for the system. Within the application architecture realm the selection and layout of computer systems and communication networks is of concern primarily to the extent that it adequately supports the application software and its functionality. However, additional factors such as scalability, convenience, information availability, data security, system administration, power and space requirements, and cost may also play important roles in computer and network architectural designs.

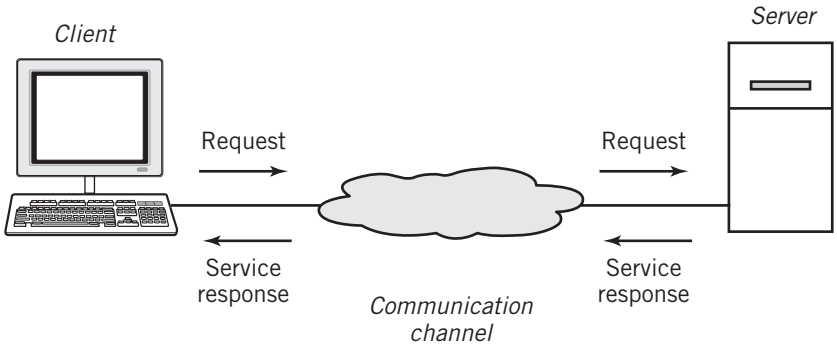
CLIENT-SERVER COMPUTING There are a variety of possible application architectures that can satisfy the requirements of modern organizations. Most, however, are based on different applications of a simple technological concept, the **client-server** model.

In a client-server configuration, a program on a client computer accepts services and resources from a complementary program on a server computer. The services and resources can include application programs, processing services, database services, Web services, file services, print services, directory services, e-mail, remote access services, even computer system initial startup service. In most cases, the client-server relationship is between complementary application programs. In certain cases, particularly for file services and printer sharing, the services are provided by programs in the operating system. Basic communication and network services are also provided by operating system programs.

Basic client-server architecture is illustrated in Figure 2.6. Notice that the link between client and server is essentially irrelevant within the application architecture view of the system. The ''cloud'' in the figure is intended to indicate only that there is a link of some kind between the client and the server. The link can be a network connection, an intranet

FIGURE 2.6

Basic Client-Server Architecture



or Internet connection, or some sort of direct connection. In fact, a single computer can act as both client and server, if desired. (A situation where this is the case is described in Chapter 16.)

The client-server model describes the relationship and behavior of programs in one or two computers under particular prescribed circumstances. It is important to understand that the client-server model does not require any special computer hardware. Furthermore, networking software within the operating system of each computer routinely provides basic communication capabilities. The only "special" software required is the software within the complementary application programs that provides the communications between the programs. The requests and responses take the form of data messages between the client and server that are understood by both application programs. As an example, slightly simplified, the HTTP request message sent to a Web server by a Web browser requesting a Web page consists of the word GET followed by a URL. If the request is successful, the message returned by the server contains the HTML text for the page.

From the description and the figure you can see that the Web browser–Web server application described as an example in Chapter 1 fits the description of a client-server application. We will return to this example momentarily.

A typical use of the client-server concept within an organization is shown in Figure 2.7. In this case, a number of clients are sharing a number of servers, showing both the **shared server** nature of client-server computing, as well as to show that there may be multiple servers offering different services on the same network. Notice, also, that the server computer labeled *S2* in the figure is running two different server applications. Since computers are capable of running multiple tasks concurrently, this is a possible scenario. The only limitations to running multiple applications on a single server are the potential slowdowns that may result from the load on the server computer and the traffic on the network to that server. Overall, there is a multiple-multiple relationship between clients and servers: a server can serve multiple clients, and a client can request services from multiple servers.

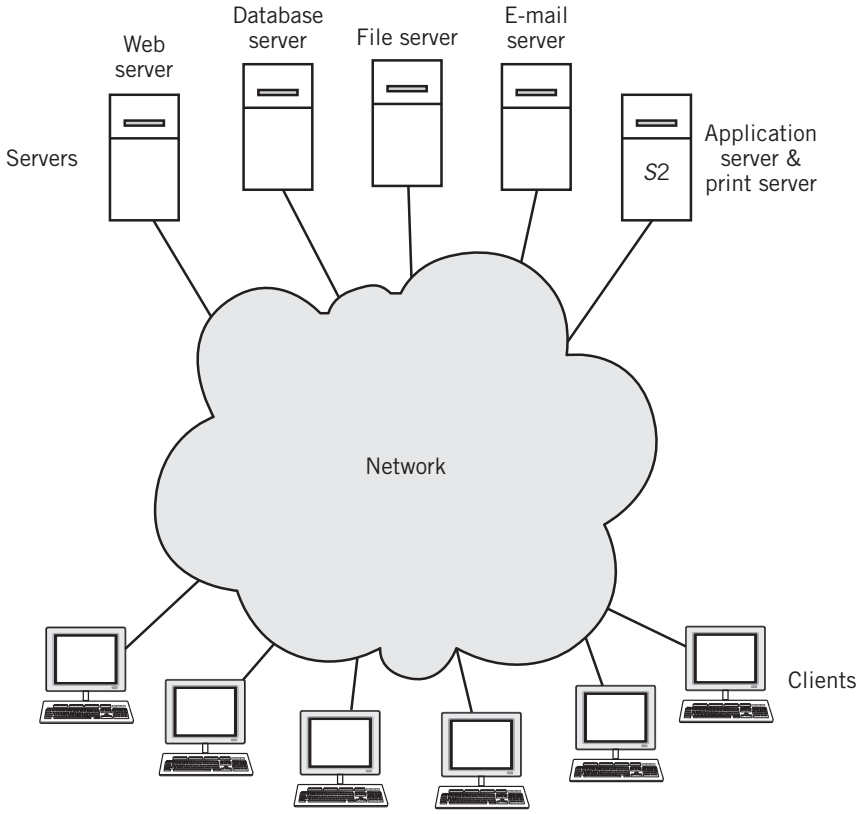
The use of client-server processing as a basis for IT system architecture has a number of advantages:

- Providing services on a single computer or on a small number of computers in a central location makes the resources and services easy to locate and available to everyone who needs them, but also allows the IT administrators to protect the resources and control and manage their use. The consistency of files and data can be managed and assured.

  For example, client-server technology can ensure that every user requesting a particular program from a server will receive the same version of the program. As another example, suppose a program has a license that limits the number of simultaneous users. The program server can easily limit distribution of the program appropriately.
- The amount of data to be stored, processed, and managed may be extremely large. It is more efficient to equip a small number of computers with the power needed than to require powerful computers at every station.
- Typically, humans request information from knowledgeable sources as they need it. Thus, the client-server approach is naturally consistent with the way humans acquire and use information.

FIGURE 2.7

Clients and Servers on a Network



The most familiar example of the use of client-server technology is the Web browser–Web server model used in intranets and on the Internet. In its simplest form, this model is an example of **two-tier architecture**. Two-tier architecture simply means that there are two computers involved in the service. The key features of this architecture are a client computer running the Web browser application, a server computer running the Web server application, a communication link between them, and a set of standard protocols, in this case, HTTP, for the communication between the Web applications, HTML for the data presentation requirements, and, usually, the TCP/IP protocol suite for the networking communcations.

In the simplest case, a Web browser requests a Web page that is stored as a pre-created HTML file on the server. More commonly, the user is seeking specific information, and a custom Web page must be created ''on the fly'', using an application program that looks up the required data in a database, processes the data as necessary, and formats it to build the desired page dynamically.

Although it is possible to maintain the database and perform the additional database processing and page creation on the same computer as the Web server, the Web server in a

large Internet-based business may have to respond to thousands of requests simultaneously. Because response time is considered an important measure by most Web users, it is often more practical to separate the database and page processing into a third computer system. The result, shown in Figure 2.8, is called a **three-tier architecture**. Note that, in this case, the Web server machine is a client to the database application and database server on the third computer. *CGI*, the *Common Gateway Interface*, is a protocol for making communication between the Web server and the database application possible. (In the figure, we have placed the page creation application software on the database machine, but it could be located on the Web server instead if doing so would balance the loads on the two machines better.) In some situations, it is even desirable to extend this idea further. Within reason, separating different applications and processing can result in better overall control, can simplify system upgrades, and can minimize scalability issues. The most general case is known as an ***n*-tier architecture**.

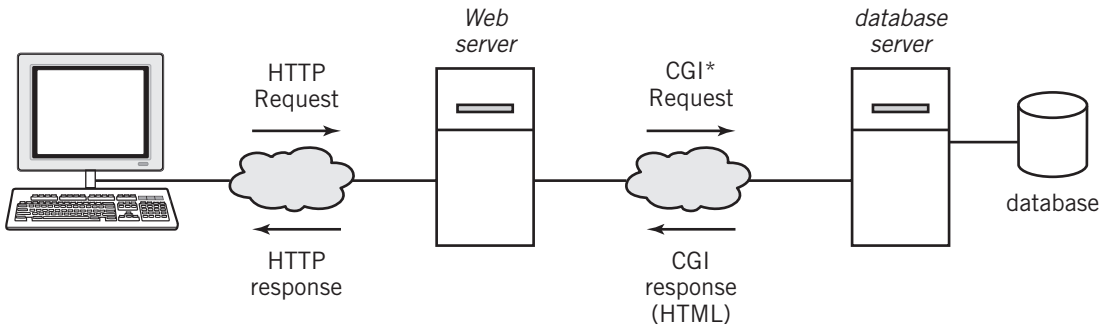
Client-server architecture is a distributed processing methodology, in which some of the processing is performed on the client system and some is performed on the server system. To see this more clearly, consider the distribution of processing between the client and server in a database application, in which the client requests specific information from a database stored on a database server.

At one extreme, the client application provides little more than a request form and a means to display the results. All of the processing is performed on the server. This might be appropriate if there is little computing power in the client. Certain so-called ''thin'' clients or ''end-user'' terminals might meet this criterion, but this situation is increasingly rare. Because this extreme case puts the entire processing load on the server, the system designer will have to specify a more powerful computer for the server; additionally, the requirements of the database server may limit the capability of the server computer system to perform other tasks or to scale for increased usage.

At the other extreme, the database server application simply accesses data from the database and passes all of the data to the client. The client application performs all of the processing. This relieves the load on the server, and it is reasonable to assume that modern client computers would be able to handle most database processing tasks relatively easily. However, the potential transfer of large amounts of raw data from the server to

FIGURE 2.8

Three-Tier Database Architecture



*CGI: Common Gateway Interface

the client for processing may put an extra burden on the network instead, requiring the system designer to specify higher speed network components at potentially higher cost and additional implementation difficulty.

A well-designed system analysis will consider the different factors, the complexity of the applications, expected network traffic, usage patterns, and the like. The optimum solution is likely to fall somewhere in the middle, with some pieces of applications on the server, others on the client.

One of the strengths of client-server architecture is its ability to enable different computer hardware and software to work together. This provides flexibility in the selection of server and client equipment tailored to the needs of both the organization and the individual users. One difficulty that sometimes arises when different computers have to work together is potential incompatibilities between the application software that resides on different equipment. This problem is commonly solved with software called **middleware**. Middleware resides logically between the servers and the clients. Typically, the middleware will reside physically on a server with other applications, but on a large system it might be installed on its own server. Either way, both clients and servers send all request and response messages to the middleware. The middleware resolves problems between incompatible message and data formats before forwarding the messages. It also manages system changes, such as the movement of a server application program from one server to another. In this case, the middleware would forward the message to the new server transparently. The middleware thus assures continued system access and stability. In general, the use of middleware can improve system performance and administration.

WEB-BASED COMPUTING The widespread success of the World Wide Web has resulted in a large base of computer users familiar with Web techniques, powerful development tools for creating Web sites and Web pages and for linking them with other applications, and protocols and standards that offer a wide and flexible variety of techniques for the collection, manipulation, and display of data and information. In addition, a powerful website is already a critical component in the system strategy of most modern organizations. Much of the data provided for the website is provided by architectural components of the organization's systems that are already in place.

Not surprisingly, these factors have led system designers to retrofit and integrate Web technology into new and existing systems, creating modern systems which take advantage of Web technology to collect, process, and present data more effectively to the users of the system.

The user of a Web-based system interacts with the system using a standard Web browser, enters data into the system by filling out Web-style forms, and accesses data using Web pages created by the system in a manner essentially identical to those used for the Internet. The organization's internal network, commonly called an **intranet**, is implemented using Web technology. To the user, integration between the intranet and the Internet is relatively seamless, limited only by the security measures designed into the system. This system architecture offers a consistent and familiar interface to users; Web-enabled applications offer access to the organization's traditional applications through the Web. Web technology can even extend the reach of these applications to employees in other parts of the world, using the Internet as the communication channel.

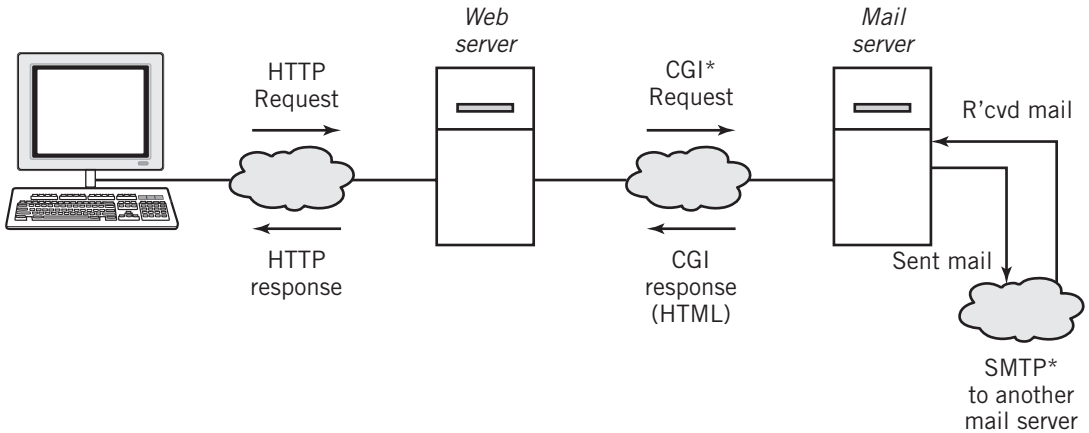
Since Web technology is based on a client-server model, it requires only a simple extension of the *n*-tier architecture to implement Web-based applications. As an example, Figure 2.9 shows a possible system architecture to implement Web-based e-mail. Note the similarity between this example and the three-tier database application shown in Figure 2.8.

Many organizations also now find it possible and advantageous to create system architectures that integrate parts of their systems with other organizations using Web technology and Web standards as the medium of communication. For example, an organization can integrate and automate its purchasing system with the order system of its suppliers to automate control of its inventory, leading to reduced inventory costs, as well as to rapid replacement and establishment of reliable stocks of inventory when they are needed. Internet standards such as XML allow the easy identification of relevant data within data streams between interconnected systems, making these applications possible and practical. This type of automation is a fundamental component of modern business-to-business operations.

PEER-TO-PEER COMPUTING An alternative to client-server architecture is **peer-to-peer architecture**. Peer-to-peer architecture treats the computers in a network as equals, with the ability to share files and other resources and to move them between computers. With appropriate permissions, any computer on the network can view the resources of any other computer on the network, and can share those resources. Since every computer is essentially independent, it is difficult or impossible to establish centralized control to restrict inappropriate access and to ensure data integrity. Even where the integrity of the system can be assured, it can be difficult to know where a particular file is located and no assurance that the resource holding that file is actually accessible when the file is needed. (The particular computer that holds the file may be turned off.) The system also may have several versions of the file, each stored on a different computer. Synchronization of different file versions is difficult to control and difficult to maintain. Finally, since

FIGURE 2.9

Three-Tier Web-Based E-Mail Architecture



*SMTP: Simple Mail Transfer Protocol
*CGI: Common Gateway Interface

data may pass openly through many different machines, the users of those machines may be able to steal data or inject viruses as the data passes through. All of these reasons are sufficient to eliminate peer-to-peer computing from consideration in any organizational situation where the computers in the network are controlled by more than one individual or group. In other words, nearly always.

There is one exception: peer-to-peer computing is adequate, appropriate, and useful for the movement of files between personal computers or to share a printer in a small office or home network.

Peer-to-peer technology has also proven viable as an Internet file sharing methodology outside the organizational structure, particularly for the downloading of music and video. The perceived advantage is that the heavy loads and network traffic associated with a server are eliminated. (There are legal ramifications, also, for a server that is sharing copyrighted material illegally.) This technique operates on the assumption that the computer searching for a file is able to find another computer somewhere by broadcasting a request across the Internet and establishing a connection with a nearby computer that can supply the file. Presumably, that computer already has established connections with other systems. All of these systems join together into a peer-to-peer network that can then share files. One serious downside to this approach, noted above, is the fact that the computers in an open, essentially random, peer-to-peer network can also be manipulated to spread viruses and steal identities. There are several serious documented cases of both.

An alternative, hybrid model uses client-server technology to locate systems and files that can then participate in peer-to-peer transactions. The hybrid model is used for instant messaging, for Skype and other online phone systems, and for Napster and other legal file download systems.

Although there have been research studies to determine if there is a place for peer-to-peer technology in organizational computing, the security risks are high, the amount of control low, and the overall usefulness limited. The results to date have been disappointing.

The Role of the System Architect

In Section 2.1, we suggested that there are different ways of viewing systems. From the discussion within this section, you can see that the IT system architect must consider the system from the perspectives of application architecture, data architecture, network architecture, and computer architecture. Each of these addresses different aspects of the IT system as a whole. For example, our consideration of different general application architectures—client-server, web-based architecture, peer-to-peer architecture—ignored the networking that links the various computers together. Similarly, we attempted to minimize the effects due to the specifics of individual computer systems when exploring the various requirements of a system from the perspective of application architecture.

Ultimately, it is the responsibility of the system architect to assess the particular needs of an organization and create a system that meets those needs while attempting to achieve an optimum balance of computer power, network capability, user convenience, and budget. To do so, the architect will consider each aspect of the system: application architecture, network requirements, specification of computer systems, and data requirements, just as the architect designing a house considers flow of people through the house, overall use of

space and room layout, individual room layouts, mechanical systems, and aesthetic design as different views of the overall architecture of the house.

Although the infrastructure design as defined by the computer hardware, system software, and communication channels is subordinate to the fundamental business requirements that determine a basic IT system architecture, the system architect must understand the features and constraints that establish the feasibility and desirability of a particular infrastructure configuration.

Google: A System Architecture Example

So far, we have considered basic system concepts and simple system architectures as examples. Most IT business systems operate primarily within an organization, with limited collaboration with other, partnered organizations and carefully controlled public access. At the opposite extreme are massive systems that are widely open to the public. Google offers a primary example of such a system.

The primary mission of Google is to provide powerful, fast search capability of material on the Internet for billions of users all over the world. Income to the organization is provided from advertising that is targeted to each user based on the specific nature of the user's search. The design of Google's IT system architecture is obviously fundamental to Google's ability to achieve its mission and to meet reasonable income goals. In keeping with the focus of this book, our primary interest is in the computer and network architectures that Google uses to meet its system requirements; however we will use this example to explore the relationship between the basic system requirements, the IT system architecture created to meet those requirements, and the specific computer and network architectures that evolved from the system architecture.

Some of the basic requirements that the Google IT system must satisfy include the following:

- It must be capable of responding to millions of simultaneous requests from all over the world with pertinent, ranked search results and appropriately targeted advertising. Most desirably, the results and advertising would be matched in language, geographic suitability, and culture as much as possible to the location of the user.
- The system must be able to troll the Internet systematically and thoroughly to retrieve data and to organize the data in such a way as to make it readily available for response to user requests. There must be a processing mechanism to establish a ranking of the results to a request.
- The system must respond to requests with a reliability as near to 100 percent as is technically possible. Individual hardware and software component failures within the system must not affect system performance adversely.
- The system must be easily scalable to handle ever-increasing numbers of requests and must be cost effective.

At the application level, the requirements identify three specific processing tasks that the system must fulfill:

1. The system must accept search requests from users, identify and rank matches, create a Web page, and serve it to the user.

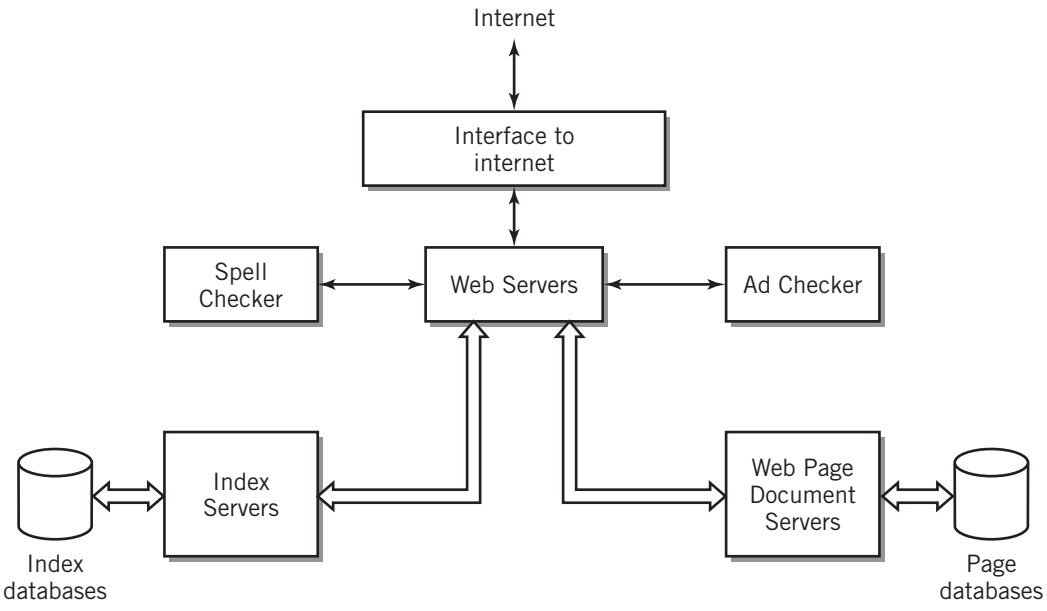
2. The system must collect data—lots of data! This task "crawls the Web", identifies the search terms (every significant word) on every Web page it encounters, and maintains an index database connecting each term to the corresponding page. It likewise stores every Web page in a Web page database and assigns a ranking value to each entry.
3. The system must manage advertisements, identify appropriate advertisements in response to user search requests, and make the advertisements available to the Web page creation application mentioned in 1.

For this discussion we will focus on the processing of search requests. When a user types the Google URL www.google.com into her browser, the Web browser uses a service called *Domain Name Service (DNS)* to identify the IP address of the Web server to which the request is to be sent. Because Google must be able to handle several milllion requests per hour, Google provides a number of alternative IP addresses representing different sites to which the request may be redirected. Based on the approximate location from which the request was sent, the request is routed by DNS to a Google data center near that location. Google maintains more than forty separate data centers around the world to serve user requests.

A simplified system diagram of the application architecture for a Google data center is shown in Figure 2.10. All of the data centers are architecturally identical, differing only in such details as the number of processors and the hardware specifications for each processor. Each data center processes requests independently. Multiple copies of all of the

FIGURE 2.10

Google Data Center Search Application Architecture



index word data and Web page data are stored locally at every data center, and updated from master data at regular intervals.

A request enters the system from the Internet and is distributed to a Google Web server for processing. A request consists of words and phrases. There are many separate Web servers available so that many requests can be processed in parallel. The words are passed to a spell checker, to an ad server, and to a pool consisting of a large number of index servers.

The spell checker checks each word and considers possible alternatives if it believes that the user may have intended something different. When appropriate, the output of the spell checker will become part of the response sent to the user. ("*Did you mean . . .* " is familiar to most Google users.) The ad checker searches for words in the advertising database that match the user's request and adds the corresponding advertisement(s) to the material that will be used to create the response page.

The index servers look up each word from the request in the index database and compile a list of matching pages for each word. The list is then adjusted for multiple words and phrases and sorted in order of relevance, based on Google's ranking algorithms. This list is then passed back to the Web server.

Next, the Web server calls upon the document servers to look up each matching page in the Web page database. The document servers return a URL, a title, and a short snippet of text for each document to the Web server. Finally, the Web server creates an HTML document from the spelling, ad, and matching page results and returns the page to the user's Web browser.

Although the application processing just described is relatively straightforward, the implementation of this system presented a number of challenges to the system architects, The index and document databases are both massive in size. Many searches will result in a large number of "hits"; each hit must be evaluated and ranked. Each hit requires retrieval and processing of a separate page from the document database. All of this processing must occur very quickly. And the numbers of searches occurring simultaneously may also be extremely large.

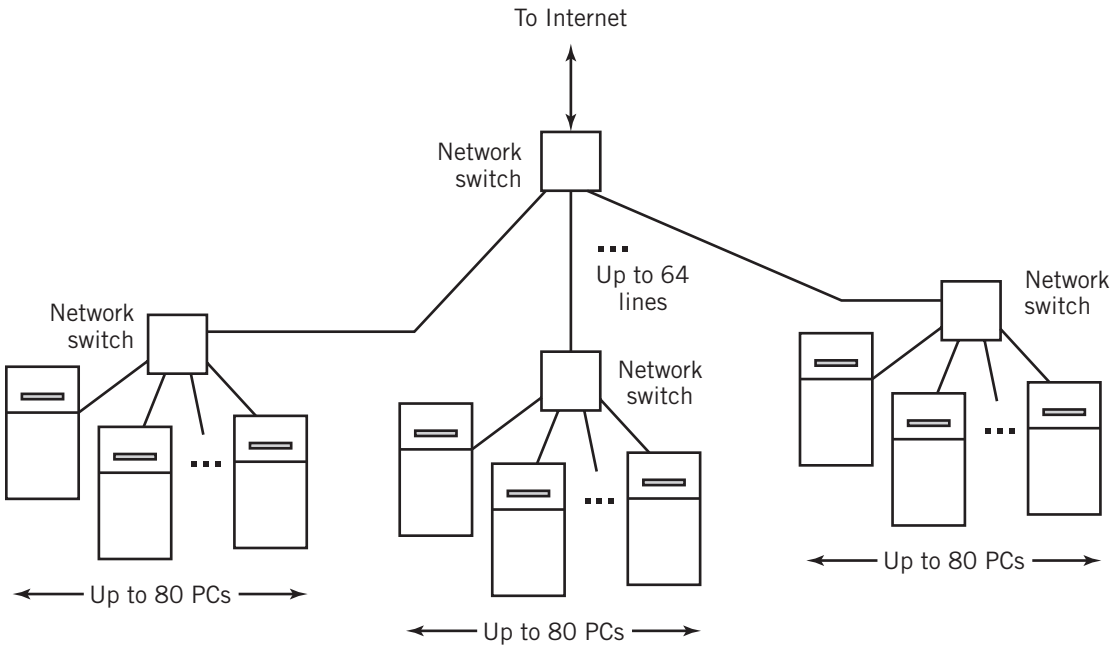
Google's system architects responded to these challenges by recognizing that each search could be processed independently on a separate computer, except for certain bottlenecks. For example, each search request arriving from the Internet could be steered by a computer to a different Web browser. They also observed that the major bottleneck was the time required to access the databases on disks, which had to be shared among all the searches taking place. Since the data in the databases never changed as a result of a search, however, they reasoned that the databases could also be replicated and accessed in parallel.

A simplified hardware representation of their solution is shown in Figure 2.11. Groups of up to eighty computers are connected together in a network, then these networks, up to sixty-four of them, are, themselves, connected together to form a larger network, sort of like a miniature Internet of up to 5,120 computers. (There are additional switches and connections built in for reliability that are not shown in the diagram.) Each computer acts as a server, with different computers assigned to different pieces of the application architecture. Each data center is equipped similarly.

Although the computers are manufactured specifically for Google, they are essentially inexpensive commodity PCs, similar to standard, medium power, non-state-of-the-art,

FIGURE 2.11

Simplified Google System Hardware Architecture



off-the-shelf PCs. Each computer has a fairly large, but still off-the-shelf, hard disk. The index and document databases are divided up among the hard disks on many computers. (Google calls these partial databases *shards*.) This design allows different searches to access different parts of the databases simultaneously. There are multiple copies of each database, so that the failure of a PC or hard disk does not affect the overall ability of the system to conduct searches. Each computer runs standard Linux operating system software, but the application software was specially written by Google programmers.

Overall, this design allows a large number of searches to progress in parallel. The use of inexpensive PC hardware makes the solution cost-effective. The system can be scaled easily by adding more computers. Finally, the failure of a PC does not result in failure and, in fact, has minimal effect on the performance of the system overall. Thus, this solution meets the original requirements admirably. It is worth noting that a fundamental understanding of computer infrastructure was key to the system architects' solution.

This discussion provides a simple overview of the Google system. Hopefully you found even this brief look at the Google system interesting and informative. There are a number of other considerations in the Google system architecture that we have glossed over for now. However, to understand the Google architecture better, it is first necessary to continue our exploration of the hardware, software, and network components that make up the Google system, as well as every other IT system. We will return for a more in-depth discussion of the Google system architecture in Supplementary Chapter 2.

SUMMARY AND REVIEW

When working with large concepts with defined boundaries, it is often easiest to think of them in terms of systems. A system can be defined as a collection of components, linked together and organized in such a way as to be recognizable as a single unit. The components themselves may also be recognized as subsystems, to be further reduced into components, when appropriate. The area outside the boundaries of a system is its environment. The system affects and is affected by various elements of the environment. In many situations, the environment supplies inputs to the system and receives outputs from the system. The patterns of relationships, connections, constraints, and linkages among the components of a system and between a system and its environment are known collectively as the architecture of the system.

Information technology systems are systems that support the strategy and operations of organizations. The technological components of an IT system include computer hardware, application software, operating system software, networks, and data. Other components include personnel, policies, and more.

There are a number of different ways of viewing an IT system, including application architecture, network architecture, software architecture, and hardware architecture. The general architecture for an IT system includes all of these considerations.

Nearly all modern IT systems rely on distributed processing. Data comes from many sources and information is required by users distributed throughout an organization and beyond. The most common application architecture to support distributed processing is client-server architecture, in which server computer systems provide various services—Web, database, file, print, processing—to client computer systems. Client-server systems are convenient for users and offer centralized control for the organization. Client-server architecture is commonly organized in tiers, ranging from two-tier to *n*-tier. The alternative architecture to client-server computing, peer-to-peer computing, is used outside of organizations as a means for sharing files over the Internet, but is of limited use in organizational settings due to difficulties in establishing stable data sources, security risks, and lack of central control. It is also possible to create a hybrid architecture, with features from both client-server and peer-to-peer computing.

A specific type of client-server architecture, Web-based computing, predominates the IT scene, primarily because users are generally familiar with the use of Web browsers, the technology is standardized and already in use in most organizations, and good development tools for designing Web pages and accessing data are readily available. Both intranets and the Internet provide user access.

Protocols are the means used to communicate between computers. IT system protocols of interest to us include network protocols such as TCP/IP, I/O protocols such as USB and PCI-Express, and application protocols such as HTTP. Standards make it possible for different system components to work together. Most modern standards are global. There are standards that are defined by interested groups and de facto standards that arise from common usage.

The first step in IT system analysis and design is about finding an appropriate architecture for a particular business situation. The task can be difficult and challenging. It is easy to see why system architects need a deep understanding of the computer system

and network components that comprise the modern IT system to make the appropriate design, selections, and tradeoffs.

Hopefully this short but concentrated chapter has prepared you for the remainder of the book, which considers in detail the data, computer system hardware, operating systems, and networks that make up the technological infrastructure of an IT system.

FOR FURTHER READING

Surprisingly, there are few books that discuss system concepts and system architecture in a truly general way. Most books that claim to be about system architecture are actually specific to a particular field, usually the field of information systems. One general book about systems is by Laszlo [LASZ96]. Some IS systems design and analysis textbooks provide a brief introduction to general system concepts. (Unfortunately, many don't!) One example of a book that provides a good introduction to system concepts is Stampf [STAM05]. Chapter 1 of Stampf covers the topics in this chapter well. Wikipedia offers other references under the topic *system*.

KEY CONCEPTS AND TERMS

abstraction
application architecture
architecture
client-server (model)
decomposition
environment
interface
intranet
middleware
n-tier architecture
peer-to-peer architecture
shared server
subsystem
system
three-tier architecture
top-down approach
two-tier architecture

READING REVIEW QUESTIONS

2.1 What are the most important ideas, keywords, and phrases that are stated in the definition of a system?

2.2 Explain the relationships among the following words: system, environment, boundary, interface.

2.3 Explain the following statement about systems: ''Decomposition is inherently hierarchical.''

2.4 Explain what is meant by the architecture of a system.

2.5 What does the top-down approach allow a system architect to do that might be more difficult otherwise?

2.6 What is the primary concern of application architecture? Give an example of application architecture, either your own, or one from the examples in the book. Explain how this example fulfills the features and requirements of the concept of application architecture.

2.7 Most modern computing in organizations is based on client-server models. Explain why this tends to be the case. Give an example of client-server computing that you

are familiar with and explain the characteristics of your example that fulfill the concept of client-server computing.

2.8 Web-based system architecture is a popular approach to many organizational systems because it offers a number of advantages to the users and to the organization over other types of systems. Discuss the primary advantages to this approach.

2.9 What are the principal responsibilities of a system architect?

2.10 Many system architects base their IT system designs on an *n*-tier architecture, where *n* is a number with value 2 or greater. Explain the difference between a single-tier architecture and an *n*-tier architecture. What are the main advantages claimed for an *n*-tier architecture?

EXERCISES

2.1 The human body is an example of an object that can be represented as a system. Consider the various ways in which you could represent the human body as a system. Select a representation and identify the components that constitute the system. Select one component and decompose it to the next subsystem level. Now consider a completely different system representation of the human body and repeat this exercise.

2.2 Consider a representation of a work organization or school with which you are familiar. Identify the major components that characterize the primary operations within the organization and draw a diagram that represents the system's organization. Show and identify the links that connect the various components. Identify the major environmental factors that impact the organization.

2.3 Consider this textbook. Using the detailed table of contents as a reference, we can represent this textbook as a hierarchical system. As a first pass, we can define this book by the five component *parts* that make up the body of the text. Identify by general name the objects that constitute the next level of decomposition below the *parts* components. Continue to do this for at least three more levels of the hierarchy.

2.4 Thinking in terms of systems allows us to analyze situations that are too complicated for us to understand as a whole. What specific characteristics and features of system thinking make this possible?

2.5 Figure 2.8 illustrates the basic architecture for a three-tier database system. This system can be viewed as an IPO (input-processing-output) system. What is the input for this system? What environmental element generates the input? (Hint: the Web browser computer is within the system boundary.) What is the expected output from this system? What environmental element receives the output? Briefly describe the processing that takes place in this system.

2.6 Based on the illustration of an iPhone shown in Figure 2.4, draw a system model for an iPhone.

2.7 It is common to represent network connections in IT systems as a cloud. (See, for example, Figures 2.6, 2.7, 2.8, and 2.9). The cloud is obviously an *abstraction* as we defined abstraction in this chapter. What does the cloud abstraction actually represent?

2.8 Suppose that you have been hired to develop a website-based sales system for a large international retail sales firm. Discuss some environmental issues that are specific to the Web design of your system that you must consider if your system is to be successful at attracting and keeping purchasing customers.

2.9 Consider a home theatre system consisting of a television set, a receiver, a DVD player, speakers, and any other components you wish to include. Draw a system diagram for this system. Include both components and links. What are the inputs to this system? What are the outputs? (Remember that the DVD player and receiver are both components *within* the system.) Now draw a system diagram for the receiver subsystem. Include both its primary components and the links between them. What are the inputs and outputs for the receiver subsystem? Do these inputs and outputs conform to the links connected to the receiver in your original system diagram?