A user interface is the part of the system with which the users interact. It includes the screen displays that provide navigation through the system, the screens and forms that capture data, and the reports that the system produces (whether on paper, on the Web, or via some other media). This chapter introduces the basic principles and processes of interface design and discusses how to design the interface structure and standards.

OBJECTIVES

■ Describe several fundamental user interface design principles.
■ Explain the process of user interface design.
■ Discuss how to design the user interface structure.
■ Explain how to design the user interface standards.
■ Be able to design a user interface.

CHAPTER OUTLINE

Introduction
Principles for User Interface Design
  Layout
  Content Awareness
  Aesthetics
  User Experience
  Consistency
  Minimize User Effort
User Interface Design Process
  Use Scenario Development
  Interface Structure Design
  Interface Standards Design
  Interface Design Prototyping
  Interface Evaluation
Navigation Design
  Basic Principles
  Types of Navigation Controls
  Messages
Input Design
  Basic Principles
  Types of Inputs
  Input Validation
Output Design
  Basic Principles
  Types of Outputs
  Media
Applying the Concepts at Tune Source
  Use Scenario Development
  Interface Structure Design
  Interface Standards Design
  Interface Template Design
  Design Prototyping
  Interface Evaluation
Summary
INTRODUCTION

Interface design is the process of defining how the system will interact with external entities (e.g., customers, suppliers, other systems). In this chapter, we focus on the design of user interfaces, but it is also important to remember that sometimes there are system interfaces that exchange information with other systems. System interfaces are typically designed as part of a systems integration effort. They are defined in general terms as part of the physical data flow diagrams (DFDs) and in the non-functional requirements (operational requirements), and are designed in detail during program design (see Chapter 10) and data storage design (see Chapter 11).

The user interface design defines the way in which the users will interact with the system and the nature of the inputs and outputs that the system accepts and produces. The user interface includes three fundamental parts. The first is the navigation mechanism, the way in which the user gives instructions to the system and tells it what to do (e.g., buttons, menus). The second is the input mechanism, the way in which the system captures information (e.g., forms for adding new customers). The third is the output mechanism, the way in which the system provides information to the user or to other systems (e.g., reports, Web pages). Each of these is conceptually different, but all are closely intertwined: All computer displays contain navigation mechanisms, and most contain input and output mechanisms. Therefore, navigation design, input design, and output design are tightly coupled.

The study of human-computer interaction (HCI) focuses on improving the interactions between users and computers by making computers more usable and receptive to the user’s needs. Some organizations employ professional HCI designers, who specialize in applying design processes to the creation of graphical user interfaces and Web interfaces.

This chapter first introduces several fundamental user interface design principles. Second, it provides an overview of the user interface design process. It then provides an overview of the navigation, input, and output components that are used in interface design. This chapter focuses on the design of Web-based interfaces and graphical user interfaces (GUI) that use windows, menus, icons, and a mouse (e.g., Windows, Macintosh). Although text-based interfaces are still commonly used on mainframes and UNIX systems, GUIs are probably the most common type of interfaces that you will use, with the possible exception of printed reports.

PRINCIPLES FOR USER INTERFACE DESIGN

In many ways, user interface design is an art. The goal is to make the interface pleasing to the eye and simple to use, while minimizing the effort users expend to accomplish their work. The system is never an end in itself; it is merely a means to accomplish the business of the organization.

1 Many people attribute the origin of GUI interfaces to Apple or Microsoft. Some people know that Microsoft copied from Apple, which in turn “borrowed” the whole idea from a system developed at the Xerox Palo Alto Research Center (PARC) in the 1970s. Very few know that the Xerox system was based on one developed by Doug Englebart that was first demonstrated at the Western Computer Conference in 1968. Around the same time, Doug also invented the mouse, desktop videoconferencing, groupware, and a host of other things we now take for granted. Doug is a legend in the computer science community and has won too many awards to count, but is relatively unknown by the general public.

We have found that the greatest problem facing experienced designers is using space effectively. Simply put, there is more information to present than room to present it. Analysts must balance the need for simplicity and pleasant appearance against the need to present the information across multiple pages or screens, which decreases simplicity. In this section, we discuss some fundamental interface design principles, which are common for navigation design, input design, and output design\(^3\) (Figure 9-1).

**Layout**

The first principle of user interface design deals with the layout of the screen, form, or report. Layout refers to organizing areas of the screen or document for different purposes and using those areas consistently throughout the user interface. Most software designed for personal computers follows the standard Windows or Macintosh approach for screen layout. This approach divides the screen into three main areas: The top area provides the user with ways to navigate through the system; the middle (and largest) area is for display of the user’s work; and the bottom area contains status information about what the user is doing.

In many cases (particularly on the Web), multiple layout areas are used. Figure 9-2 shows a screen with five navigation areas, each of which is organized to provide different functions and navigation within different parts of the system.

---

<table>
<thead>
<tr>
<th>Principle</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Layout</strong></td>
<td>The interface should be a series of areas on the screen that are used consistently for different purposes—for example, a top area for commands and navigation, a middle area for information to be input or output, and a bottom area for status information.</td>
</tr>
<tr>
<td><strong>Content awareness</strong></td>
<td>Users should always be aware of where they are in the system and what information is being displayed.</td>
</tr>
<tr>
<td><strong>Aesthetics</strong></td>
<td>Interfaces should be functional and inviting to users through careful use of white space, colors, and fonts. There is often a trade-off between including enough white space to make the interface look pleasing and losing so much space that important information does not fit on the screen.</td>
</tr>
<tr>
<td><strong>User experience</strong></td>
<td>Although ease of use and ease of learning often lead to similar design decisions, there is sometimes a trade-off between the two. Novice users or infrequent users of software will prefer ease of learning, whereas frequent users will prefer ease of use.</td>
</tr>
<tr>
<td><strong>Consistency</strong></td>
<td>Consistency in interface design enables users to predict what will happen before they perform a function. It is one of the most important elements in ease of learning, ease of use, and aesthetics.</td>
</tr>
<tr>
<td><strong>Minimize user effort</strong></td>
<td>The interface should be simple to use. Most designers plan on having no more than three mouse clicks from the starting menu until users perform work.</td>
</tr>
</tbody>
</table>
The top area provides the standard web browser navigation and command controls that change the contents of the entire system. The navigation area on the left edge maneuvers between sections and changes all content to its right. The other two section navigation areas at the top and bottom of the page provide other ways to navigate between sections. The content in the middle of the page displays the results (i.e., software review articles) and provides additional navigation within the page about these reviews.

This use of multiple layout areas for navigation also applies to inputs and outputs. Data areas on reports and forms are often subdivided into subareas, each of which is used for different types of information. These areas are almost always rectangular in shape, although sometimes space constraints will require odd shapes. Nonetheless, the margins on the edges of the screen should be consistent. Each of
the areas within the report or form is designed to hold different information. For example, on an order form (or order report), one part may be used for customer information (e.g., name, address), one part for information about the order in general (e.g., date, payment information), and one part for the order details (e.g., how many units of which items at what price each). Each area is self-contained so that information in one area does not run into another.

The areas and information within areas should have a natural intuitive flow to minimize users’ movement from one area to the next. People in Western nations (e.g., Europe, North America) tend to read top to bottom, left to right, so that related information should be placed so that it is used in this order (e.g., address lines, followed by city, state/province, and then ZIP code/postal code.) Sometimes, the sequence is in chronological order, or from the general to the specific, or from most frequently to least frequently used. In any event, before the areas are placed on a form or report, the analyst should have a clear understanding of what arrangement makes the most sense for how the form or report will be used. The flow between sections should also be consistent, whether horizontal or vertical (Figure 9-3). Ideally, the areas will remain consistent in size, shape, and placement for the forms used to enter information (whether on paper or on a screen) and the reports used to present it.

**Content Awareness**

Content awareness refers to the ability of an interface to make the user aware of the information it contains with the least amount of effort by the user. All parts of the interface, whether navigation, input, or output, should provide as much content awareness as possible, but it is particularly important for forms or reports that are used quickly or irregularly (e.g., a Web site).

Content awareness applies to the interface in general. All interfaces should have titles (on the screen frame, for example). Menus should show where the user is and, if possible, where the user came from to get there. For example, in Figure 9-2, the top line in the center site navigation bar shows that the user is in the Small Business Computing Channel section of the winplanet.com site.

Content awareness also applies to the area within forms and reports. All areas should be clear and well defined (with titles if space permits) to reduce the chances that users become confused about the information in any area. Then users can quickly locate the part of the form or report that is likely to contain the information they need. Sometimes the areas are marked by lines, colors, or headings (e.g., the site navigation links on the left side in Figure 9-2); in other cases, the areas are only implied (e.g., the page links in the center of Figure 9-2).

Content awareness also applies to the fields within each area. Fields are the individual elements of data that are input or output. The field labels that identify the fields on the interface should be short and specific—objectives that often conflict. There should be no uncertainty about the format of information within fields, whether for entry or display. For example, a date of 10/5/12 means different things, depending on whether you are in the United States (October 5, 2012) or in Canada (May 10, 2012). Any fields for which there is the possibility of uncertainty or multiple interpretations should provide explicit explanations.

Content awareness also applies to the information that a form or report contains. In general, all forms and reports should contain a preparation date (i.e., the date printed or the date data were completed) so that the age of the information is
FIGURE 9-3
Interface Flow between Sections
obvious. Likewise, all printed forms and software should provide version numbers so that users, analysts, and programmers can identify outdated materials.

Figure 9-4, a form from the University of Georgia, illustrates the logical grouping of fields into areas with an explicit box (top left), as well as an implied area with no box (lower left). The address fields within the address area follow a clear, natural order. Field labels are short where possible (see the top left), but long where more information is needed to prevent misinterpretation (see the bottom left).

Aesthetics

*Aesthetics* refers to designing interfaces that are pleasing to the eye. Interfaces do not have to be works of art, but they do need to be functional and inviting to use. In most cases, “less is more,” meaning that a simple, minimalist design is the best.

Space usually is at a premium on forms and reports, and often there is the temptation to squeeze as much information as possible onto a page or a screen. Unfortunately, this can make a form or report so unpleasant that users do not want to complete it. In general, all forms and reports need at least a minimum amount of white space that is intentionally left blank.

What was your first reaction when you looked at Figure 9-4? This is the most unpleasant form at the University of Georgia, according to staff members. Its density is too high; it has too much information packed into too small a space with too little white space. Although it may be efficient in saving paper by being one page instead of two, it is not effective for many users.

In general, novice or infrequent users of an interface, whether on a screen or on paper, prefer interfaces with low density, often one with a density of less than 50% (i.e., less than 50% of the interface occupied by information). More experienced users prefer higher densities, sometimes approaching 90% occupied, because they know where information is located and high densities reduce the amount of physical movement through the interface. We suspect that the form in Figure 9-4 was designed for the experienced staff in the personnel office, who use it daily, rather than for the clerical staff in academic departments, who have less personnel experience and use the form only a few times a year.

The design of text is equally important. In general, there should be one font for the entire form or report and no more than two sizes of that font on the form or report. A larger font size may be used for titles, section headings, etc., and a smaller font for the report or form content. If the form or report will be printed, the smaller font should be at least 8 points in size. A minimum of 10 points is preferred if the users will be older people. For forms or reports displayed on the screen, consider a minimum of a 12-point font size if the display monitor is set for a high screen resolution. Italics and underlining should be avoided because they make text harder to read.

Serif fonts (i.e., those having letters with serifs, or “tails,” such as Times Roman or the font you are reading right now) are the most readable for printed reports, particularly for small letters. Sans serif fonts (i.e., those without serifs, such as Tahoma or Arial or the ones used for the chapter titles in this book) are the most readable for computer screens and are often used for headings in printed reports. Never use all capital letters, except possibly for titles—all-caps text “shouts” and is harder to read.

Color and patterns should be used carefully and sparingly and only when they serve a purpose. (About 10% of men are color blind, so the improper use of color
**FIGURE 9-4**

*Form Example*
can impair their ability to read information.) A quick trip around the Web will
demonstrate the problems caused by indiscriminate use of colors and patterns.
Remember, the goal is pleasant readability, not art; colors and patterns should be
used to strengthen the message, not overwhelm it. Color is best used to separate and
categorize items, such as showing the difference between headings and regular text,
or to highlight important information. Therefore, colors with high contrast should
be used (e.g., black and white). In general, black text on a white background is the
most readable, with blue on red the least readable. (Most experts agree that back-
ground patterns on Web pages should be avoided.) Color has been shown to affect
emotion, with red provoking intense emotion (e.g., anger) and blue provoking low-
ered emotions (e.g., drowsiness).

User Experience

*User experience* refers to designing the user interface with the users’ level of
computer experience in mind. A computer system will be used by people with
experience and by people with no experience; the user interface should be
designed for both types. Novice users usually are most concerned with *ease of
learning*—how quickly and easily they can learn to use the system. Expert users
are typically more concerned with *ease of use*—how quickly and easily they can
complete a task with the system once they have learned how to use it. Often, these
two objectives are complementary and lead to similar design decisions, but some-
times, there are trade-offs. Novices, for example, often prefer menus that show all
available system functions, because these promote ease of learning. Experts, on
the other hand, sometimes prefer fewer menus that are organized around the most
commonly used functions.

Systems that will end up being used by many people on a daily basis are more
likely to have a majority of expert users (e.g., order entry systems). Although inter-
faces should try to balance ease of use and ease of learning, these types of systems
should put more emphasis on ease of use rather than on ease of learning. Users
should be able to access the commonly employed functions quickly, with few key-
strokes or a small number of menu selections.

In many other systems (e.g., decision support systems), most people will
remain occasional users for the lifetime of the system. In this case, greater empha-
sis may be placed on ease of learning rather than on ease of use.

Although ease of use and ease of learning often go hand in hand, sometimes
they don’t. Research shows that expert and novice users have different require-
ments and behavior patterns in some cases. For example, novices virtually never
look at the bottom area of a screen that presents status information, but experts
refer to the status bar when they need information. Most systems should be
designed to support frequent users, except for systems that are to be used infre-
quently or those for which many new users or occasional users are expected (e.g.,
the Web). Likewise, systems that contain functionality that is used only occasion-
ally must contain a highly intuitive interface, or an interface that contains explicit
guidance regarding its use.

The balance between quick access to commonly used and well-known func-
tions and guidance through new and less-well-known functions is challenging to the
interface designer, and this balance often requires elegant solutions. Microsoft
Office, for example, addresses this issue through the use of the “show me” func-
tions that demonstrate the menus and buttons for specific functions. These features
remain in the background until they are needed by novice users (or even experienced users when they use an unfamiliar part of the system).

**Consistency**

*Consistency* in design is probably the single most important factor in making a system simple to use, because it enables users to predict what will happen. When interfaces are consistent, users can interact with one part of the system and then know how to interact with the rest—aside, of course, from elements unique to those parts. Consistency usually refers to the interface within one computer system, so that all parts of the same system work in the same way. Ideally, however, the system also should be consistent with other computer systems in the organization and with whatever commercial software is used (e.g., Windows). For example, many users are familiar with the Web, so the use of Web-like interfaces can reduce the amount of learning required by the user. In this way, the user can reuse Web knowledge, thus significantly reducing the learning curve for a new system.

Consistency occurs at many different levels. Consistency in the navigation controls conveys how actions in the system should be performed. For example, using the same icon or command to change an item clearly communicates how changes are made throughout the system. Consistency in terminology is also important. This refers to using the same words for elements on forms and reports (e.g., not “customer” in one place and “client” in another). We also believe that consistency in report and form design is important, although one study suggests that being too consistent can cause problems. When reports and forms are very similar except for minor changes in titles, users sometimes mistakenly use the wrong report or form and either enter incorrect data or misinterpret its information. The implication for design is to make the reports and forms similar, but give them some distinctive elements (e.g., color, size of titles) that enable users to immediately detect differences.

**Minimize User Effort**

Finally, interfaces should be designed to minimize the amount of effort needed to accomplish tasks. This means using the fewest possible mouse clicks or keystrokes to move from one part of the system to another. Most interface designers follow the *three-clicks rule*: Users should be able to go from the start or main menu of a system to the information or action they want in no more than three mouse clicks or three keystrokes.

---

**9-1 Web Page Critique**

Visit the Web home page for your university and navigate through several of its Web pages. Evaluate the extent to which they meet the six design principles.

---

USER INTERFACE DESIGN PROCESS

User interface design is a five-step process that is iterative—analysts often move back and forth between steps rather than proceed sequentially from step 1 to step 5 (Figure 9-5). First, the analysts examine the DFDs and use cases developed in the analysis phase (see Chapters 4 and 5) and interview users to develop use scenarios that describe users’ commonly employed patterns of actions so that the interface can enable users to quickly and smoothly perform these scenarios. Second, the analysts develop the interface structure diagram (ISD) that defines the basic structure of the interface. This diagram (or set of diagrams) shows all the interfaces (e.g., screens, forms, and reports) in the system and how they are connected. Third, the analysts design interface standards, which are the basic design elements on which interfaces in the system are based. Fourth, the analysts create an interface design prototype for each of the individual interfaces in the system, such as navigation controls, input screens, output screens, forms (including preprinted paper forms), and reports. Finally, the individual interfaces are subjected to interface evaluation to determine whether they are satisfactory and how they can be improved.

Interface evaluations almost always identify improvements, so the interface design process is repeated in a cyclical process until no new improvements are identified. In practice, most analysts interact closely with the users during the interface design process, so that users have many chances to see the interface as it evolves rather than waiting for one overall interface evaluation at the end of the interface design process. It is better for all concerned (both analysts and users) if

---

5 One of the best books on user interface design is Ben Schneiderman, Designing the User Interface: Strategies for Effective Human-Computer Interaction, 3d ed., Reading, MA: Addison-Wesley, 1998.
changes are identified sooner rather than later. For example, if the interface structure or standards need improvement, it is better to identify changes before most of the screens that use the standards have been designed.

**Use Scenario Development**

A *use scenario* is an outline of the steps that the users perform to accomplish some part of their work. A use scenario is one commonly used path through a use case. Recall that use cases and data flow diagrams may include multiple ways in which the response to the event can be completed. For example, think back to the Search and Browse Tunes use case from Figure 4-14 in Chapter 4 that was modeled in a level 1 DFD shown in Figure 5-18 in Chapter 5. This figure shows process 1.2 (Process Search Requests) as being distinct from process 1.3 (Process Tune Selection). We model the two processes separately and write the programs separately because they are separate processes within process 1 (Search and Browse Tunes).

The DFD was designed to model all possible uses of the system—that is, its complete functionality or all possible paths through the use case. But use scenarios are just one path through the use case. In one use scenario, for example, a user will browse through many tunes, much like someone browsing through a real music store looking for interesting music. He or she will search for a tune, listen to a sample, perhaps add it to the shopping cart, browse for more, and so on. Eventually, the user will want to purchase the download(s), perhaps removing some selections from the shopping cart beforehand.

In another use scenario, a user will want to buy one specific tune. He or she will go directly to the tune, price it, and buy it immediately, much like someone running into a store, making a beeline for the one item he or she wants, and immediately paying and leaving the store. This user will enter the tune information in the search portion of the system, look at the resulting cost information, and immediately buy the download or leave. Anything that slows him or her down will risk losing the sale. For this use scenario, we need to ensure that the path through the DFD as presented by the interface is short and simple, with very few menus and mouse clicks.

Use scenarios are presented in a simple narrative description that is tied to the DFD. Figure 9-6 shows the two use scenarios just described. The key point in using use scenarios for interface design is not to document all possible use scenarios within a use case, because then you end up just repeating the DFD in a different form. The goal is to describe the handful of most commonly occurring use scenarios so that the interface can be designed to enable the most common uses to be performed simply and easily.

---

**YOUR TURN**

Visit the Web site for your university and navigate through several of its Web pages. Develop two use scenarios for it.
Interface Structure Design

The *interface structure design* defines the basic components of the interface and how they work together to provide functionality to users. An interface structure diagram (ISD) is used to show how all the screens, forms, and reports used by the system are related and how the user moves from one to another. Most systems have several ISDs, one for each major part of the system.

An ISD is somewhat similar to a DFD in that it uses boxes and lines to show structure. However, unlike DFDs, there are no commonly used rules or standards for their development. With one approach, each interface element (e.g., screen, form, report) on an ISD is drawn as a box and is given a unique number (at the top) and a unique name (in the middle). The numbers usually follow a tree-type structure, although this is not always done. Unlike the DFDs, however, the numbers do not mean that all the screens belong to “parents” higher in the tree; instead, they usually imply relationships between a menu and a submenu. The lines denote the ability to navigate from one menu to another.

Each box on the ISD also shows (at the bottom) the DFD process that is supported by the interface (Figure 9-7). Sometimes, there is more than one interface for a given process (e.g., in Figure 9-7, interfaces 1.1 through 1.3 support process 1.1.1); in other cases, there is only one interface for each process (e.g., interfaces 3.1 through 3.3 support processes 1.1.3.1 through 1.1.3.3).

**FIGURE 9-6**
Two Use Scenarios for the Search and Browse Tunes Use Case

<table>
<thead>
<tr>
<th>Use Scenario: The Browsing Shopper</th>
</tr>
</thead>
<tbody>
<tr>
<td>User is not sure what they want to buy and will browse through several tunes.</td>
</tr>
<tr>
<td>1. User may search for a specific artist or browse through a music category (1.2).</td>
</tr>
<tr>
<td>2. User will likely read the basic information for several tunes, as well as the marketing material for some. He or she will likely listen to music samples and browse related tunes (1.3).</td>
</tr>
<tr>
<td>3. User will put the tune in the shopping cart (1.3) and will continue browsing (1.2).</td>
</tr>
<tr>
<td>4. Eventually, the user will want to purchase the download, but will probably want to look through the shopping cart, possibly discarding some tunes first (1.3).</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Use Scenario: The Hurry-up Shopper</th>
</tr>
</thead>
<tbody>
<tr>
<td>User knows exactly what he or she wants and wants it quickly.</td>
</tr>
<tr>
<td>1. User will search for a specific artist or tune (1.2).</td>
</tr>
<tr>
<td>2. User will look at the price and enough other information to confirm that the tune is the desired tune (1.3).</td>
</tr>
<tr>
<td>3. User will want to buy the download (process 2) or move on to other Web sites.</td>
</tr>
</tbody>
</table>

The numbers in parentheses refer to process numbers in the DFD.

**Interface Structure Design**

The *interface structure design* defines the basic components of the interface and how they work together to provide functionality to users. An interface structure diagram (ISD) is used to show how all the screens, forms, and reports used by the system are related and how the user moves from one to another. Most systems have several ISDs, one for each major part of the system.

An ISD is somewhat similar to a DFD in that it uses boxes and lines to show structure. However, unlike DFDs, there are no commonly used rules or standards for their development. With one approach, each interface element (e.g., screen, form, report) on an ISD is drawn as a box and is given a unique number (at the top) and a unique name (in the middle). The numbers usually follow a tree-type structure, although this is not always done. Unlike the DFDs, however, the numbers do not mean that all the screens belong to “parents” higher in the tree; instead, they usually imply relationships between a menu and a submenu. The lines denote the ability to navigate from one menu to another.

Each box on the ISD also shows (at the bottom) the DFD process that is supported by the interface (Figure 9-7). Sometimes, there is more than one interface for a given process (e.g., in Figure 9-7, interfaces 1.1 through 1.3 support process 1.1.1); in other cases, there is only one interface for each process (e.g., interfaces 3.1 through 3.3 support processes 1.1.3.1 through 1.1.3.3).

**9-3 Use Scenario Development for an Automated Teller Machine**

**Your Turn**

Pretend that you have been charged with the task of redesigning the interface for the ATM at your local bank. Develop two use scenarios for it.
Each interface is linked to other interfaces by lines that show how users can transition from one interface to the next. In most cases, the interfaces form a hierarchy, or a tree; but sometimes, an interface is linked to one outside of the hierarchy, as shown by the link from Form J to Form B (e.g., the ability to update customer information, such as address, while entering a new order).

The basic structure of the interface follows the basic structure of the business process itself as defined in the process model. The analyst starts with the DFD and develops the fundamental flow of control of the system as it moves from process to process. There are usually several major parts to an information
system, each of them distinct, in the same way that there are several high-level processes in a DFD. In general—but not always—there is one ISD for each process on the level 0 DFD.

The analyst then examines the use scenarios to see how well the ISD supports them. Quite often, the use scenarios identify paths through the ISD that are more complicated than they should be. The analyst then reworks the ISD to simplify the ability of the interface to support the use scenarios, sometimes by making major changes to the menu structure, sometimes by adding shortcuts.

**Interface Standards Design**

The interface standards are the basic design elements that are common across the individual screens, forms, and reports within the system. Depending on the application, there may be several sets of interface standards for different parts of the system (e.g., one for Web screens, one for paper reports, one for input forms). For example, the part of the system used by data-entry operators may mirror other data-entry applications in the company, whereas a Web interface for displaying information from the same system may adhere to some standardized Web format. Likewise, each individual interface may not contain all the elements in the standards (e.g., a report screen may not have an “edit” capability), and they may contain additional characteristics beyond the standard ones, but the standards serve as the touchstone which ensures that the interfaces are consistent across the system.

**Interface Metaphor**

First of all, the analysts must develop the fundamental interface metaphor(s) that defines how the interface will work. An interface metaphor is a concept from the real world that is used as a model for the computer system. The metaphor helps the user to understand the system and enables the user to predict what features the interface might provide, even without actually using the system. Sometimes systems have one metaphor, whereas in other cases there are several metaphors in different parts of the system.

In many cases, the metaphor is explicit. Quicken, for example, uses a checkbook metaphor for its interface, even to the point of having the users type information into an on-screen form that looks like a real check register. In other cases, the metaphor is implicit, or unstated, but it is there nonetheless. Many Windows systems use the paper form or table as a metaphor.

In some cases, the metaphor is so obvious that it requires no thought. The Tune Source Digital Music Download system, for example, will use the retail music store as the metaphor (e.g., shopping cart). In other cases, a metaphor is hard to identify. In general, it is better not to force a metaphor that really doesn’t fit a system, because an ill-fitting metaphor will confuse users by promoting incorrect assumptions.

**Interface Templates**

The interface template defines the general appearance of all screens in the information system and the paper-based forms and reports that are

---

**YOUR TURN**

Pretend that you have been charged with the task of redesigning the interface for the ATM at your local bank. Design an ISD that shows how a user would navigate among the screens.
used. The template design, for example, specifies the basic layout of the screens (e.g., where the navigation area[s], status area, and form/report area[s] will be placed) and the color scheme(s) that will be applied. It defines whether windows will replace one another on the screen or will cascade on top of each other. The template defines a standard placement and order for common interface actions (e.g., “File, Edit, View” rather than “File, View, Edit”). In short, the template draws together the other major interface design elements: metaphors, objects, actions, and icons.

**Interface Objects** The template specifies the names that the interface will use for the major interface objects, the fundamental building blocks of the system such as the entities and data stores. In many cases, the object names are straightforward, such as calling the shopping cart the “shopping cart.” In other cases, it is not simple. For example, Tune Source has chosen to call its digital music downloads “tunes.” Some people may refer to individual music selections as “tracks” or “cuts.” Obviously, the object names should be easily understood and should help promote the interface metaphor.

In general, in cases of disagreements between the users and the analysts over names, whether for objects or actions (discussed later), the users should win. A more understandable name always beats a more precise or more accurate name.

**Interface Actions** The template also specifies the navigation and command language style (e.g., menus) and grammar (e.g., object–action order; see “Navigation Design” later in this chapter). The template gives names to the most commonly used interface actions in the navigation design (e.g., “buy” versus “purchase,” or “exit” versus “quit”).

**Interface Icons** The interface objects and actions, and also their status (e.g., deleted, error), may be represented by interface icons. Icons are pictures that will appear on command buttons as well as in reports and forms to highlight important information. Icon design is very challenging because it means developing a simple picture less than half the size of a postage stamp that needs to convey an often-complex meaning. The simplest and best approach is to adopt icons developed by others (e.g., a blank page to indicate “create a new file,” a diskette to indicate “save”). This has the advantage of quick icon development, and the icons may already be well understood by users because users have seen them in other software.

Commands are actions that are especially difficult to represent with icons because they are in motion, not static. Many icons have become well known from widespread use, but icons are not as well understood as it was at first believed that they would be. The use of icons can sometimes cause more confusion than insight. (For example, did you know that a picture of a sweeping paintbrush in Microsoft Word means “format painter”? Icon meanings become clearer with use, but because they are often cryptic, many applications now provide text tool tips that appear when the pointer hovers over an icon. This feature explains the purpose of the icon in words.

---

**YOUR TURN**

Pretend that you have been charged with the task of redesigning the interface for the ATM at your local bank. Develop an interface standard that includes metaphors, objects, actions, icons, and a template.
Interface Design Prototyping

An interface design prototype is a mock-up or a simulation of a computer screen, form, or report. A prototype is prepared for each interface in the system to show the users and the programmers how the system will perform. In the “old days,” an interface design prototype was usually specified on a paper form that showed what would be displayed on each part of the screen. Paper forms are still used today, but more and more interface design prototypes are being built with computer tools instead of on paper. The three most common approaches to interface design prototyping are storyboards, HTML prototypes, and language prototypes.

Storyboard At its simplest, an interface design prototype is a paper-based storyboard. The storyboard shows hand-drawn pictures of what the screens will look like and how they will flow from one screen to another, in the same way that a storyboard for a cartoon shows how the action will flow from one scene to the next (Fig. 9-8). Storyboards are the simplest technique because all they require is paper (often on a flip chart) and a pen—and someone with some artistic ability.
HTML Prototype  One of the most common types of interface design prototypes used today is the *HTML prototype*. As the name suggests, an HTML prototype is built with the use of Web pages created in HTML (hypertext mark-up language). The designer uses HTML to create a series of Web pages that show the fundamental parts of the system. The users can interact with the pages by clicking on buttons and entering pretend data into forms (but because there is no system behind the pages, the data are never processed). The pages are linked together so that, as the user clicks on buttons, the requested part of the system appears. HTML prototypes are superior to storyboards in that they enable users to interact with the system and gain a better sense of how to navigate among the different screens. However, HTML has limitations—the screens shown in HTML will never appear exactly like the real screens in the system (unless, of course, the real system will be a Web system in HTML).

Language Prototype  A *language prototype* is an interface design prototype built in the actual language or by the actual tool that will be used to build the system. Language prototypes are designed in the same ways as HTML prototypes. (They enable the user to move from screen to screen, but they perform no real processing). For example, in Visual Basic, it is possible to create and view screens without actually attaching program code to the screens. See Figure 9-9 for a sample Visual Basic Language prototype. Language prototypes take longer to develop.
FIGURE 9-9
Sample Language Prototype — Part B

FIGURE 9-9
Sample Language Prototype — Part C
than do storyboards or HTML prototypes, but they have the distinct advantage of showing exactly what the screens will look like. The user does not have to guess about the shape or position of the elements on the screen.

Selecting the Appropriate Techniques Projects often use a combination of different interface design prototyping techniques for different parts of the system. Storyboarding is the fastest and least expensive, but provides the least amount of detail. Language prototyping is the slowest, most expensive, and most detailed approach. HTML prototyping falls between the two extremes. Therefore, storyboarding is used for parts of the system in which the interface is well understood and when more expensive prototypes are thought to be unnecessary. HTML prototypes and language prototypes are used for parts of the system that are critical, yet not well understood.

Interface Evaluation

The objective of interface evaluation is to understand how to improve the interface design. Interface design is subjective; there are no formulas that guarantee a great user interface. Most interface designers intentionally or unintentionally design an interface that meets their personal preferences, which may or may not match the preferences of the users. The key message, therefore, is to have as many people as possible evaluate the interface—and the more users, the better. Most experts recommend involving at least 10 potential users in the evaluation process.

Many organizations save interface evaluation for the very last step in the SDLC before the system is installed. Ideally, however, interface evaluation should be performed while the system is being designed—before it is built—so that any major design problems can be identified and corrected before the time and cost of programming have been spent on a weak design. It is not uncommon for the system to undergo one or two major changes after the users see the first interface design prototype, because they identify problems that are overlooked by the project team.
As with interface design prototyping, interface evaluation can take many different forms, each with different costs and different levels of detail. Four common approaches are heuristic evaluation, walk-through evaluation, interactive evaluation, and formal usability testing. As with interface design prototyping, the different parts of a system can be evaluated by different techniques.

**Heuristic Evaluation**  A heuristic evaluation examines the interface by comparing it to a set of heuristics, or principles, for interface design. The project team develops a checklist of interface design principles—from the list at the start of this chapter, for example, as well as the lists of principles in the navigation, input, and output design sections. At least three members of the project team then individually work through the interface design prototype, examining each interface to ensure that it satisfies each design principle on the formal checklist. After each member has gone through the prototype separately, they all meet as a team to discuss their evaluation and identify specific improvements that are required. Because this technique does not involve the users, it is considered the weakest type of evaluation.

**Walk-through Evaluation**  An interface design walk-through evaluation is a meeting conducted with the users who will ultimately have to operate the system. The project team presents the prototype to the users and walks them through the various parts of the interface. The project team shows the storyboard or actually demonstrates the HTML or language prototype and explains how the interface will be used. The users identify improvements to each of the interfaces that are presented.

**Interactive Evaluation**  With an interactive evaluation, the users themselves actually work with the HTML or language prototype in one-on-one sessions with members of the project team. (An interactive evaluation cannot be used with a storyboard.) As the user works with the prototype (often by going through the use scenarios or just navigating at will through the system), he or she tells the project team members what he or she likes and doesn’t like and what additional information or functionality is needed. As the user interacts with the prototype, team members record the situation when the user appears to be unsure of what to do, makes mistakes, or misinterprets the meaning of an interface component. If the pattern of uncertainty, mistakes, or misinterpretations recurs across several evaluation sessions with several users, it is a clear indication that those parts of the interface need improvement.

**Formal Usability Testing**  Formal usability testing is commonly done with commercial software products and products developed by large organizations that will be widely used through the organization. As the name suggests, it a very formal—almost scientific—process that can be used only with language prototypes (and systems that have been completely built and are awaiting installation or shipping). As with interactive evaluation, usability testing is done in one-on-one sessions in which a user works directly with the software. However, it is typically done in a special lab equipped with video cameras and special software that records each and every keystroke and mouse operation so that they can be replayed to help in understanding exactly what the user did.

---

Pretend that you have been charged with the task of redesigning the interface for the ATM at your local bank. What type of prototyping and interface evaluation approach would you recommend?

The user is given a specific set of tasks to accomplish (usually the use scenarios), and after some initial instructions the project teams member(s) are not permitted to interact with the user to provide assistance. The user must work with the software without help, which can be hard on the user if he or she becomes confused with the system. It is critical that users understand that the goal is to test the interface, not their abilities, and if they are unable to complete the task, the interface—not the user—has failed the test. Several performance measures are used, such as time to complete the task, error rate, and user satisfaction.

Formal usability testing is very expensive. Each one-user session (which typically lasts one to two hours) can take one to two days to analyze due to the volume of detail collected in the computer logs and videotapes. Most usability testing involves 5 to 10 users. Fewer than 5 users makes the results depend too much on the specific individual users who participated, but more than 10 users is often too expensive to justify (unless you work for a large commercial software developer).

**NAVIGATION DESIGN**

The navigation component of the interface enables the user to enter commands to navigate through the system and perform actions to enter and review information it contains. The navigation component also presents messages to the user about the success or failure of his or her actions. The goal of the navigation system is to make the system as simple as possible to use. A good navigation component is one that the user never really notices. It simply functions the way the user expects, and thus the user gives it little thought.

**Basic Principles**

One of the hardest things about using a computer system is learning how to manipulate the navigation controls to make the system do what you want. Analysts usually must assume that users have not read the manual, have not attended training, and do not have external help readily at hand. All controls should be clear and understandable and placed in an intuitive location on the screen. Ideally, the controls should anticipate what the user will do and simplify his or her efforts. For example, many set-up programs are designed so that, for a typical installation, the user can simply keep pressing the “Next” button.

**Prevent Mistakes**  The first principle of designing navigation controls is to prevent the user from making mistakes. A mistake costs time and creates frustration. Worse still, a series of mistakes can cause the user to discard the system. Mistakes can be
reduced by labeling commands and actions appropriately and by limiting choices. Too many choices can confuse the user, particularly when they are similar and hard to describe in the short space available on the screen. When there are many similar choices on a menu, consider creating a second level of menu or a series of options for basic commands.

Never display a command that cannot be used. For example, many Windows applications gray-out commands that cannot be used; they are displayed on pull-down menus in a very light-colored font, but they cannot be selected. This shows that they are available, but that they cannot be used in the current context. It also keeps all menu items in the same place.

When the user is about to perform a critical function that is difficult or impossible to undo (e.g., deleting a file), it is important to confirm the action with the user (and make sure the selection was not made by mistake). This is usually done by having the user respond to a confirmation message that explains what the user has requested and asks the user to confirm that this action is correct.

**Simplify Recovery from Mistakes** No matter what the system designer does, users will make mistakes. The system should make it as easy as possible to correct these errors. Ideally, the system will have an “Undo” button that makes mistakes easy to override; however, writing the software for such buttons can be very complicated.

**Use Consistent Grammar Order** One of the most fundamental decisions is the grammar order. Most commands require the user to specify an object (e.g., file, record, word) and the action to be performed on that object (e.g., copy, delete). The interface can require the user to first choose the object and then the action (an object–action order) or first choose the action and then the object (an action–object order). Most Windows applications use an object–action grammar order (e.g., think about copying a block of text in your word processor).

The grammar order should be consistent throughout the system, both at the data element level and at the overall menu level. Experts debate about the advantages of one approach over the other, but because most users are familiar with the object–action order, most systems today are designed with that approach.

**Types of Navigation Controls**

There are two traditional hardware devices that can be used to control the user interface: the keyboard and a pointing device, such as a mouse, trackball, or touch screen. In recent years, voice recognition systems have made an appearance, but they are not yet common. There are three basic software approaches for defining user commands: languages, menus, and direct manipulation.

**Languages** With a command language, the user enters commands in a special language developed for the computer system (e.g., UNIX and SQL both use command languages). Command languages sometimes provide greater flexibility than do other approaches, because the user can combine language elements in ways not predetermined by developers. However, they put a greater burden on users because users must learn syntax and type commands rather than select from a well-defined, limited number of choices. Systems today use command languages sparingly, except in cases in which there are an extremely large number of command combinations, making it impractical to try to build all combinations into a menu (e.g., SQL queries for databases).
Design a navigation system for a form into which users must enter information about customers, products, and orders. For all three information categories, users will want to change, delete, find one specific record, and list all records.

Natural language interfaces are designed to understand the user’s own language (e.g., English, French, Spanish). These interfaces attempt to interpret what the user means, and often they present back to the user a list of interpretations from which to choose. Many “help” systems today enable the user to ask free-form questions to get help.

Menus The most common type of navigation system today is the menu. A menu presents the user with a list of choices, each of which can be selected. Menus are easier to learn than languages because the user sees an organized, but limited, set of choices. Clicking on an item with a pointing device or pressing a key that matches the menu choice (e.g., a function key) takes very little effort. Therefore, menus are usually preferred to languages.

Menus should be designed with care, because the submenus behind a main menu are hidden from users until they click on the menu item. It is better to make menus broad and shallow (i.e., with each menu containing many items and each item containing only one or two layers of menus) rather than narrow and deep (i.e., with each menu containing only a few items, but each item leading to three or more layers of menus). A broad and shallow menu presents the user with the most information initially, so that he or she can see many options, and requires only a few mouse clicks or keystrokes to perform an action. A narrow and deep menu makes users hunt and seek for items hidden behind menu items and requires many more clicks or keystrokes to perform an action.

Research suggests that in an ideal world, any one menu should contain no more than eight items and it should take no more than two mouse clicks or keystrokes from any menu to perform an action (or three from the main menu that starts a system).7 However, analysts sometimes must break this guideline in the design of complex systems. In this case, menu items are often grouped together and separated by a horizontal line (Fig. 9-10). Often, menu items have hot keys that enable experienced users to quickly invoke a command with keystrokes in lieu of a menu choice (e.g., Control-F in Word invokes the Find command, whereas Alt-F opens the File menu).

Menus should put together similar categories of items so that the user can intuitively guess what each menu contains. Most designers recommend grouping menu items by interface objects (e.g., Customers, Purchase Orders, Inventory) rather than by interaction actions (e.g., New, Update, Format), so that all actions pertaining to one object are in one menu, all actions for another object are in a different menu, and so on. However, this is highly dependent on the specific interface. As Figure 9-10 shows, Microsoft Visual Studio groups menu items by interface.

---

objects (e.g., File, Project, Window) and by interface actions (e.g., Edit, View, Build) on the same menu. Some of the more common types of menus include menu bars, drop-down menus, pop-up menus, tab menus, tool bars, and image maps. (See Figures 9-10 and 9-11.)

Direct Manipulation With direct manipulation, the user enters commands by working directly with interface objects. For example, users can change the size of objects in Microsoft PowerPoint by clicking on objects and moving the sides, or they can move files in Windows Explorer by dragging the file names from one folder to another. Direct manipulation can be simple, but it suffers from two problems. First,
users familiar with language- or menu-based interfaces don’t always expect it. Second, not all commands are intuitive. (For example, how do you copy [not move] files in Windows Explorer?)

**Messages**

Messages are the way in which the system responds to a user and informs him or her of the status of the interaction. There are many different types of messages, such as error messages, confirmation messages, acknowledgment messages, delay messages, and help messages (Figure 9-12). In general, messages should be clear, concise, and complete, which are sometimes conflicting objectives. All messages should be grammatically correct and free of jargon and abbreviations (unless they
Navigation Design

<table>
<thead>
<tr>
<th>Type of Menu</th>
<th>When to Use</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Menu Bar</strong></td>
<td>Main menu for system</td>
<td>• Use the same organization as the operating system and other packages (e.g., File, Edit, View).</td>
</tr>
<tr>
<td></td>
<td>List of commands at the top of the screen. Always on screen.</td>
<td>• Menu items are always one word, never two.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Menu items lead to other menus, rather than performing action.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Never allow users to select actions they can’t perform. (Instead, use grayed-out items).</td>
</tr>
<tr>
<td><strong>Drop-down Menu</strong></td>
<td>Second-level menu, often from menu bar</td>
<td>• Menu items are often multiple words.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Avoid abbreviations.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Menu items perform action or lead to another cascading drop-down menu, pop-up menu, or tab menu.</td>
</tr>
<tr>
<td><strong>Hyperlink Menu</strong></td>
<td>Main menu for Web-based system</td>
<td>• Most users are familiar with hyperlink menus on the left edge of the screen, although they can be placed along any edge.</td>
</tr>
<tr>
<td></td>
<td>A set of items arranged as a menu, usually along one edge of the screen.</td>
<td>• Menu items are usually only one or two words.</td>
</tr>
<tr>
<td><strong>Embedded Hyperlinks</strong></td>
<td>As a link to ancillary, optional information</td>
<td>• Used sparingly to provide additional information, because they can complicate navigation.</td>
</tr>
<tr>
<td><strong>Pop-up Menu</strong></td>
<td>As a shortcut to commands for experienced users</td>
<td>• Usually open a new window that is closed once the action is complete so that the user can return to the original use scenario.</td>
</tr>
<tr>
<td></td>
<td>Menu that pops up and floats over the screen. Disappears after one use.</td>
<td>• Often (not always) invoked by a right click in Windows-based systems.</td>
</tr>
<tr>
<td><strong>Tab Menu</strong></td>
<td>When user needs to change several settings or perform several related commands</td>
<td>• Menu choices vary depending on pointer position.</td>
</tr>
<tr>
<td></td>
<td>Menu items should be short to fit on the tab label.</td>
<td>• Often overlooked by novice users, so usually should duplicate functionality provided in other menus.</td>
</tr>
<tr>
<td><strong>Tool Bar</strong></td>
<td>As a shortcut to commands for experienced users</td>
<td>• Menu items should be short to fit on the tab label.</td>
</tr>
<tr>
<td></td>
<td>Menu of buttons (often with icons) that remains on the screen until closed.</td>
<td>• Avoid more than one row of tabs because clicking on a tab to open it can change the order of the tabs, and in virtually no other case does selecting from a menu rearrange the menu itself.</td>
</tr>
<tr>
<td><strong>Image Map</strong></td>
<td>Only when the graphical image adds meaning to the menu</td>
<td>• All buttons on the same tool bar should be the same size.</td>
</tr>
<tr>
<td></td>
<td>Graphical image in which certain areas are linked to actions or other menus.</td>
<td>• If the labels vary dramatically in size, then use two different sizes (small and large).</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Buttons with icons should have a tool tip—an area that displays a text phrase explaining the button when the user pauses the pointer over it.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Image should convey meaning to show which parts perform an action when clicked.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Tool tips can be helpful.</td>
</tr>
</tbody>
</table>

**FIGURE 9-11**
Types of Menus

are users’ jargon and abbreviations). Avoid negatives, because they can be confusing (e.g., replace “Are you sure you do not want to continue?” with “Do you want to quit?”). Likewise, avoid humor, because it wears off quickly after the same message appears dozens of times.

Messages should require the user to acknowledge them (by clicking, for example), rather than being displayed for a few seconds and then disappearing. The exceptions are messages that inform the user of delays in processing, which should disappear once the delay has passed. In general, messages are text, but sometimes standard icons are used. For example, Windows 7 displays a revolving circle when the system is busy.
### Types of Messages

<table>
<thead>
<tr>
<th>Type of Messages</th>
<th>When to Use</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Error message</td>
<td>When user does something that is not permitted or not possible</td>
<td>Always explain the reason and suggest corrective action. Traditionally, error messages have been accompanied by a beep, but many applications now omit it or permit users to remove it.</td>
</tr>
<tr>
<td>Confirmation message</td>
<td>When user selects a potentially dangerous choice, such as deleting a file</td>
<td>Always explain the cause and suggest possible action. Often include several choices other than “OK” and “cancel.”</td>
</tr>
<tr>
<td>Acknowledgment message</td>
<td>Seldom or never; users quickly become annoyed with all the unnecessary mouse clicks</td>
<td>Acknowledgment messages are typically included because novice users often like to be reassured that an action has taken place. The best approach is to provide acknowledgment information without a separate message on which the user must click. For example, if the user is viewing items in a list and adds one, then the updated list on the screen showing the added item is sufficient acknowledgment.</td>
</tr>
<tr>
<td>Delay message</td>
<td>When an activity takes more than seven seconds</td>
<td>This message should permit the user to cancel the operation in case he or she does not want to wait for its completion. The message should provide some indication of how long the delay may last.</td>
</tr>
<tr>
<td>Help message</td>
<td>In all systems</td>
<td>Help information is organized by table of contents and/or keyword search. Context-sensitive help provides information that is dependent on what the user was doing when help was requested. Help messages and online documentation are discussed in Chapter 12.</td>
</tr>
</tbody>
</table>

All messages should be carefully crafted, but error messages and help messages require particular care. Messages (and especially error messages) should always explain the problem in polite, succinct terms (e.g., what the user did incorrectly) and explain corrective action as clearly and as explicitly as possible so that the user knows exactly what needs to be done. In the case of complicated errors, the error message should display what the user entered, suggest probable causes for the error, and propose possible user responses. When in doubt, provide either more information than the user needs or the ability to get additional information. Error messages should provide a message number. Message numbers are not intended for users, but their presence makes it simpler for those staffing help desks and customer support lines to identify problems and help users, because many messages use similar wording.

### INPUT DESIGN

Input mechanisms facilitate the entry of data into the computer system, whether highly structured data, such as order information (e.g., item numbers, quantities, costs), or unstructured information (e.g., comments). Input design means designing the screens used to enter the information, as well as any forms on which users write or type information (e.g., time cards, expense claims).
Basic Principles

The goal of input design is to capture accurate information for the system simply and easily. The fundamental principles for input design reflect the nature of the inputs (whether batch or online) and ways to simplify their collection.

Use Online and Batch Processing Appropriately

There are two general approaches for entering inputs into a computer system: online processing and batch processing. With online processing (sometimes called transaction processing), each input item (e.g., a customer order, a purchase order) is entered into the system individually, usually at the same time as the event or transaction prompting the input. For example, when you borrow a book from the library, buy an item at the store, or make an airline reservation, the computer system that supports each process uses online processing to immediately record the transaction in the appropriate database(s). Online processing is most commonly used when it is important to have real-time information about the business process. For example, when you reserve an airline seat, the seat is no longer available for someone else to use, so that piece of information must be recorded immediately.

With batch processing, all the inputs collected over some period are gathered together and entered into the system at one time in a batch. Some business processes naturally generate information in batches. For example, most hourly payrolls are done by batch processing because time cards are gathered together in batches and processed at once. Batch processing also is used for transaction processing systems that do not require real-time information. For example, most stores send sales information to district offices so that new replacement inventory can be ordered. This information could be sent in real time as it is captured in the store, so that the district offices are aware within a second or two that a product is sold. If stores do not need up-to-the-second real-time data, they will collect sales data throughout the day and transmit the data every evening in a batch to the district office. This batching simplifies the data communications process and often cuts communications costs. It does mean, however, that inventories are not accurate in real time, but rather are accurate only at the end of the day after the batch has been processed.

Capture Data at the Source

Perhaps the most important principle of input design is to capture the data in an electronic format at the original source or as close to the original source as possible. In the early days of computing, computer systems replaced traditional manual systems that were based on paper forms. As these business processes were automated, many of the original paper forms remained, either because no one thought to replace them or because it was too expensive to do so. Instead, the business process continued to contain manual forms that were taken to the computer center in batches to be typed into the computer system by a data-entry operator.

Many business processes still operate this way today. For example, many organizations have expense claim forms that are completed by hand and submitted to an accounting department, which approves them and enters them into the system in batches. There are three problems with this approach. First, it is expensive because it duplicates work. (The form is filled out twice, once by hand and once by keyboard.) Second, it increases processing time because the paper forms must be physically moved through the process. Third, it increases the cost and the probability of error, because it separates the entry from the processing of information; someone may misread the handwriting on the input form, data could be entered incorrectly, or the original input may contain an error that invalidates the information.
Most transaction processing systems today are designed to capture data at its source. *Source data automation* refers to using special hardware devices to automatically capture data without requiring anyone to type it. Stores commonly use *bar code readers* that automatically scan products and enter data directly into the computer system. No intermediate formats, such as paper forms, are used. Similar technologies include *optical character recognition*, which can read printed numbers and text (e.g., on checks); *magnetic stripe readers*, which can read information encoded on a stripe of magnetic material similar to a diskette (e.g., credit cards); and *smart cards* that contain microprocessors, memory chips, and batteries (much like credit card-size calculators). A recent development is the *RFID (radio frequency identification) tag*, combining a microprocessor chip with an antenna to broadcast its information to electronic readers. Information can be read from or written to the RFID tag. As well as reducing the time and cost of data entry, these systems reduce errors because they are far less likely to capture data incorrectly. Today, portable computers and scanners allow data to be captured at the source even in mobile settings (e.g., air courier deliveries, use of rental cars).

A lot of information, however, cannot be collected by these automatic systems. Today, with the widespread use of the Web, much data is captured directly from the customer. Consequently, the forms for capturing information on-screen should provide a logical flow and should allow the user to easily complete the forms and check their entries before submitting them. Since data entered by the user is prone to inaccuracies, validation checks (see Figure 9-15) should be used whenever possible.

**Minimize Keystrokes** Another important principle is to minimize keystrokes. Keystrokes cost time and money, whether they are performed by a customer, user, or trained data-entry operator. The system should never ask for information that can be obtained in another way (e.g., by retrieving it from a database or by performing a calculation). Likewise, a system should not require a user to type information that can be selected from a list; selecting reduces errors and speeds entry.

In many cases, data have values that often recur. These frequent values should be used as the *default value* for the data so that the user can simply accept the value and not have to retype it time and time again. Examples of default values are the current date, the area code held by the majority of a company’s customers, and a billing address that is based on the customer’s residence. Most systems permit changes to default values to handle data entry exceptions as they occur.

**9-8 Career Services**

Pretend that you are designing the new interface for a career services system at your university that accepts student résumés and presents them in a standard format to recruiters. Describe how you could incorporate the basic principles of input design into your interface design. Remember to include the use of online versus batch data input, the capture of information, and plans to minimize keystrokes.
**Types of Inputs**

Each data item that has to be input is linked to a field, on the form into which its value is typed. Each field also has a field label, which is the text beside, above, or below the field, that tells the user what type of information belongs in the field. Often, the field label is similar to the name of the data element, but the two do not have to have identical wording. In some cases, a field will display a template over the entry box to show the user exactly how data should be typed. There are many different types of inputs, in the same way that there are many different types of fields. (See Fig. 9-13.)

**Text**  As the name suggests, a text box is used to enter text. Text boxes can be defined to have a fixed length or can be scrollable and accept a virtually unlimited amount of text. In either case, boxes can contain single or multiple lines of textual information. Never use a text box if you can use a selection box.

Text boxes should have field labels placed to the left of the entry area, with their size clearly delimited by a box (or a set of underlines in a non-GUI interface). If there are multiple text boxes, their field labels and the left edges of their entry boxes should be aligned. Text boxes should permit standard GUI functions such as cut, copy, and paste.
Numbers A *number box* is used to enter numbers. Some software can automatically format numbers as they are entered, so that 3452478 becomes $34,524.78. Dates are a special form of numbers that sometimes have their own type of number box. Never use a number box if you can use a selection box.

Selection Box A *selection box* enables the user to select a value from a predefined list. The items in the list should be arranged in some meaningful order, such as alphabetical for long lists, or in order of most frequently used. The default selection value should be chosen with care. A selection box can be initialized as “unselected” or, better still, start with the most commonly used item already selected.

There are six commonly used types of selection boxes: *check boxes, radio buttons, on-screen list boxes, drop-down list boxes, combo boxes*, and *scroll bars* (Figs. 9-13, 9-14). The choice among the types of text selection boxes generally comes down to one of screen space and the number of choices the user can select. If screen space is limited and only one item can be selected, then a drop-down list box is the best choice, because not all list items need to be displayed on the screen. If screen space is limited, but the user can select multiple items, an on-screen list box that displays only a few items can be used. Check boxes (for multiple selections) and radio buttons (for single selections) both require all list items to be displayed at all times, thus requiring more screen space, but since they display all choices, they are often simpler for novice users.
Input Validation

All data entered into the system must be validated in order to ensure accuracy. Input validation (also called edit checks) can take many forms. Ideally, to prevent invalid information from entering the system, computer systems should not accept data that fail any important validation check. However, this can be very difficult, and invalid data often slip by data-entry operators and the users providing the information. It is up the system to identify invalid data and either make changes or notify someone who can resolve the information problem.

There are six different types of validation checks: completeness check, format check, range check, check digit check, consistency check, and database check. (See Figure 9-15.) Every system should use at least one validation check on all entered data and, ideally, will perform all appropriate checks where possible.
Consider a Web form that a student would use to input student information and résumé information into a career services application at your university. Sketch out how this form would look and identify the fields that the form would include. What types of validity checks would you use to make sure that the correct information is entered into the system?

<table>
<thead>
<tr>
<th>Type of Validation</th>
<th>When to Use</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Completeness check</td>
<td>When several fields must be entered before the form can be processed</td>
<td>If required information is missing, the form is returned unprocessed to the user.</td>
</tr>
<tr>
<td>Format check</td>
<td>When fields are numeric or contain coded data</td>
<td>Ideally, numeric fields should not permit users to type text data, but if this is not possible, the entered data must be checked to ensure that it is numeric. Some fields use special codes or formats (e.g., license plates with three letters and three numbers) that must be checked.</td>
</tr>
<tr>
<td>Range check</td>
<td>With all numeric data, if possible</td>
<td>A range check permits only numbers between correct values. Such a system can also be used to screen data for “reasonableness”—e.g., rejecting birthdates prior to 1890 because people do not live to be a great deal over 100 years old (most likely, 1990 was intended).</td>
</tr>
<tr>
<td>Check digit check</td>
<td>When numeric codes are used</td>
<td>Check digits are numbers added to a code as a way of enabling the system to quickly validate correctness. For example, U.S. Social Security Numbers and Canadian Social Insurance Numbers assign only eight of the nine digits in the number. The ninth number—the check digit—is calculated by a mathematical formula from the first eight numbers. When the identification number is typed into a computer system, the system uses the formula and compares the result with the check digit. If the numbers don’t match, then an error has occurred.</td>
</tr>
<tr>
<td>Consistency checks</td>
<td>When data are related</td>
<td>Data fields are often related. For example, someone’s birth year should precede the year in which he or she was married. Although it is impossible for the system to know which data are incorrect, it can report the error to the user for correction.</td>
</tr>
<tr>
<td>Database checks</td>
<td>When data are available to be checked</td>
<td>Data are compared against information in a database (or file) to ensure that they are correct. For example, before an identification number is accepted, the database is queried to ensure that the number is valid. Because database checks are more “expensive” than the other types of checks (they require the system to do more work), most systems perform the other checks first and perform database checks only after the data have passed the previous checks.</td>
</tr>
</tbody>
</table>
OUTPUT DESIGN

Outputs are the reports that the system produces, whether on the screen, on paper, or in other media, such as the Web. Outputs are perhaps the most visible part of any system, because a primary reason for using an information system is to access the information that it produces.

Basic Principles

The goal of the output mechanism is to present information to users so that they can accurately understand it with the least effort. The fundamental principles for output design reflect how the outputs are used and ways to make it simpler for users to understand them.

Understand Report Usage

The first principle in designing reports is to understand how they are used. Reports can be used for many different purposes. In some cases—but not very often—reports are read cover to cover because all information is needed. In most cases, reports are used to identify specific items or are used as references to find information, so the order in which items are sorted on the report or grouped within categories is critical. This is particularly important for the design of electronic or Web-based reports. Web reports that are intended to be read end to end should be presented in one long scrollable page, whereas reports that are primarily used to find specific information should be broken into multiple pages, each with a separate link. Page numbers and the date on which the report was prepared also are important for reference reports.

The frequency of the report may also play an important role in its design and distribution. Real-time reports provide data that are accurate to the second or minute at which they were produced (e.g., stock market quotes). Batch reports are those that report historical information that may be months, days, or hours old, and they often provide additional information beyond the reported information (e.g., totals, summaries, historical averages).

There are no inherent advantages to real-time reports over batch reports. The only advantages lie in the time value of the information. If the information in a report is time critical (e.g., stock prices, air traffic control information), then real-time reports have value. This is particularly important because real-time reports often are expensive to produce; unless they offer some clear business value, they may not be worth the extra cost.

Manage Information Load

Most managers get too much information, not too little (i.e., the information load confronting the manager is too great). The goal of a well-designed report is to provide all the information needed to support the task for which it was designed. This does not mean that the report should provide all the information available on the subject—just what the users decide they need to perform their jobs. In some cases, this may result in the production of several different reports on the same topics for the same users, because they are used in different ways. This is not bad design.

For users in Westernized countries, the most important information generally should be presented first, in the top left corner of the screen or paper report. Information should be provided in a format that is usable without modification. The user should not need to re-sort the report’s information, highlight critical information to find it more easily amid a mass of data, or perform additional mathematical calculations.
Minimize Bias  No analyst sets out to design a biased report. The problem with bias is that it can be very subtle; analysts can introduce it unintentionally. Bias can be introduced by the way in which lists of data are sorted, because entries that appear first in a list may receive more attention than those appearing later in the list. Data often are sorted in alphabetic order, making those entries starting with the letter A more prominent. Data can be sorted in chronological order (or reverse chronological order), placing more emphasis on older (or most recent) entries. Data may be sorted by numeric value, placing more emphasis on higher or lower values. For example, consider a monthly sales report by state. Should the report be listed in alphabetic order by state name, in descending order by the amount sold, or in some other order (e.g., geographic region)? There are no easy answers to this, except to say that the order of presentation should match the way in which the information is used.

Graphic displays and reports can present particularly challenging design issues.8 The scale on the axes in graphs is particularly subject to bias. For most types of graphs, the scale should always begin at zero; otherwise, comparisons among values can be misleading. For example, in Fig. 9-16, have sales increased by very much since 2006? The numbers in both charts are the same, but the visual images the two present are quite different. A glance at Fig. 9-16a would suggest only minor changes, whereas a glance at Fig. 9-16b might suggest that there have been some significant increases. In fact, sales have increased by a total of 15% over five years, or 3% per year. Fig. 9-16a presents the most accurate picture; Fig. 9-16b is biased because the scale starts very close to the lowest value in the graph and misleads the eye into inferring that there have been major changes (i.e., more than doubling from “two lines” in 2006 to “five lines” in 2011). Fig. 9-16b is the default graph produced by Microsoft Excel, so be aware of how easy it is to unintentionally introduce bias in graphs.

Types of Outputs

There are many different types of reports, such as detail reports, summary reports, exception reports, turnaround documents, and graphs (Fig. 9-17). Classifying reports is challenging because many reports have characteristics of several different types. For example, some detail reports also produce summary totals, making them summary reports.

9-10 Finding Bias

Read through recent copies of a newspaper or popular press magazine such as Time, Newsweek, or BusinessWeek and find four graphs. How many are biased and how many are unbiased?8

---

There are many different types of media used to produce reports. The two dominant media today are paper and electronic. Paper is the more traditional medium and is relatively permanent, easy to use, and accessible in most situations. It also is highly portable, at least for short reports.

Paper also has several rather significant drawbacks. It is inflexible. Once the report is printed, it cannot be sorted or reformatted to present a different view of the information. Likewise, if the information on the report changes, the entire report must be reprinted. Paper reports are expensive, are hard to duplicate, and require considerable supplies (paper, ink) and storage space. Paper reports are also hard to quickly move long distances (e.g., from a head office in Toronto to a regional office in Bermuda).

Many organizations are therefore moving to electronic production of reports, whereby reports are “printed,” but stored in electronic format on file servers or Web servers so that users can easily access them. Often, the reports are available in more
I helped a university department develop a small decision support system to analyze and rank students who applied to a specialized program. Some of the information was numeric and could easily be processed directly by the system (e.g., grade point average, standardized test scores). Other information required the faculty to make subjective judgments among the students (e.g., extracurricular activities, work experience). The users entered their evaluations of the subjective information via several data analysis screens in which the students were listed in alphabetical order.

In order to make the system “easier to use,” the reports listing the results of the analysis were also presented in alphabetical order by student name, rather than in order from the highest ranked student to the lowest ranked student. In a series of tests prior to installation, the users selected the wrong students to admit in 20 percent of the cases. They assumed, wrongly, that the students listed first were the highest ranked students and simply selected the first students on the list for admission. Neither the title on the report, nor the fact that all the students’ names were in alphabetical order made them realize that they had read the report incorrectly.

**Question:**
What concerns could this problem raise about the rest of the system?
Applying the Concepts at Tune Source

9-D Cutting Paper to Save Money

One of the Fortune 500 firms with which I have worked had an 18-story office building for its world headquarters. It devoted two full floors of this building to nothing more than storing “current” paper reports (a separate warehouse was maintained outside the city for “archived” reports such as tax documents). Imagine the annual cost of office space in the headquarters building tied up in these paper reports. Now imagine how a staff member would gain access to the reports, and you can quickly understand the driving force behind electronic reports, even if most users end up printing them. Within one year of switching to electronic reports (for as many reports as practical) the paper report storage area was reduced to one small storage room.

Alan Dennis

QUESTION:
What types of reports are most suited to electronic format? What types of reports are less suited to electronic reports?

APPLYING THE CONCEPTS AT TUNE SOURCE

In the Tune Source case, there are three different user interfaces to be designed: Process 1: Search and Browse Tunes; Process 2: Purchase Tunes; and Process 3: Promote Tunes. (See Fig. 5-17 in Chapter 5.) In this section, we focus only on Process 1, the Web portion used by customers to find tunes of interest.

Use Scenario Development

The first step in the interface design process was to develop the key use scenarios for the Digital Music Download system. Jason Wells, senior systems analyst at Tune Source and project manager for the Digital Music Download system, began by examining the DFD and thinking about the types of users and how they would interact with the system. As discussed previously, Jason identified two use scenarios: the browsing shopper and the hurry-up shopper. (See Fig. 9-6.) Jason also thought of several other use scenarios for the Web site in general, but he omitted them because they were not common. Likewise, he thought of several use scenarios that did not lead to sales (e.g., fans looking for information about their favorite artists and music), and he omitted them as well, as they were not important in the design of the Web site.

Interface Structure Design

Next, Jason created an ISD for the Web system. He began with the DFDs to ensure that all functionality defined for the system was included in the ISD. Fig. 9-18 shows predefined formats than are their paper-based counterparts, because the cost of producing and storing different formats is minimal. Electronic reports also can be produced on demand as needed, and they enable the user to search more easily for certain words. Furthermore, electronic reports can provide a means to support ad hoc reports when users customize the contents of the report at the time the report is generated. Some users may still print the electronic report on their own printers. The reduced cost of electronic delivery over distance and improved user access to the reports usually offsets the cost of local printing.
the ISD for the Web portion of process 1. In practice, some of the processes on the level 1 DFD for this part of the system (Fig. 5-18) might be decomposed into several level 2 DFDs. However, to keep things simple, in this chapter we show an ISD that links to the level 1 DFD in Fig. 5-18, rather than attempting to create more DFDs and link the ISD to them.

The Initial Interface Structure Design The Digital Music Download system will have a main menu or home page (interface number 0) that will enable the user to initiate a search (1), which would allow the user to enter search criteria to produce a list of tunes based on tune (1.1), artist (1.2), or composer (1.3), or to view the user’s Favorites List (1.4). The home page would also have a browse feature (2) that would enable the user to select a music genre and produce a list of tunes within that genre (2.1). The home page would also have links to Web promotions (3) that would lead to lists of most popular tunes (3.1), newly added tunes (3.2), or Web special promotions (3.3).

Each of these lists of tunes would enable the user to click on a specific tune title and view detailed information about it (4), and listen to music samples. Jason decided to provide the additional marketing materials (e.g., reviews) on a separate page (4.1), rather than including them on the main entry for each tune, to prevent overcrowding and long time delays on the Web. Both the tune page (4) and additional material page (4.1) would let the user listen to a sample (4.2) and then put the tune on the user’s Favorites List (4.3) and add to it the shopping cart (4.4). The user could then return to the home page (0) to enter a new search.

Reviewing the Interface Structure Design Jason then examined the use scenarios to see how well the ISD enabled different types of users to work through the system. He started with the “browsing shopper” scenario and followed it through the ISD, imagining what would appear on each screen and pretending to navigate through the system. He found that the ISD worked well.

Jason next explored the “hurry-up shopper” scenario. In this case, the ISD did not work as well. Moving from the home page to the search page to the list of matching tunes to the tune info page with price and other information takes three mouse clicks. This falls within the three-clicks rule, but for someone in a hurry, this may be too many. Jason decided to add a “quick search” option to the home page (interface 0) that would enable the user to enter one search criterion (e.g., searching by just artist name or title, rather than doing a more detailed search, as would be possible on the search page) that, with one click, would take the user to the one tune that matched the criterion (interface 4) or to a list of tunes if there were more than one (interfaces 1.1, 1.2, 1.3). This would enable an impatient user to get to the tune of interest in one or two clicks.

Interface Standards Design Once the ISD was complete, Jason moved on to develop the interface standards for the system. The interface metaphor was straightforward: a Tune Source retail music store. The key interface objects and actions were equally straightforward, as was the use of the Tune Source logo icon (Fig. 9-19).

Interface Template Design For the interface template, Jason decided on a simple, clean design that had the Tune Source logo in the upper left corner. The template had a navigation menu
across the top for navigation within the entire Web site (e.g., overall Web site home page, store locations) and one menu down the side of the page for navigation within the Digital Music Download system. The left-edge menu contained the links to the three top-level screens (interfaces 1, 2 and 3 in Fig. 9-18), as well as the “quick search” option. The center area of the screen is used for displaying forms and reports when the appropriate link is clicked. See Fig. 9-20. At this point, Jason decided to seek some quick feedback on the interface structure and standards before investing time in prototyping the interface designs. Therefore, he met with Carly Edwards, the project sponsor, to discuss the emerging design. Making changes at this point would be much simpler than waiting until after doing the prototype. Carly had a few suggestions, so, after the meeting, Jason made the changes and moved into the design prototyping step.

**Design Prototyping**

Jason decided to use Visio to create prototypes of the system. The Digital Music Download system was new territory for Tune Source and a strategic investment in a new business model, so it was important to make sure that no key issues were overlooked. The prototypes would provide the most detailed information and could be developed rapidly. Once satisfied with the layout, Jason planned to create HTML prototypes that could be used to test the interface interactively.

Jason began designing the prototype, starting with the home screen, and gradually worked his way through all the screens. The process was very iterative, and he
made many changes to the screens as he worked. Once he had an initial prototype designed, he posted it on Tune Source’s internal intranet and solicited comments from several friends with lots of Web experience. He revised it according to the comments he received. Fig. 9-21 presents some screens from the prototype.

**Interface Evaluation**

The next step was interface evaluation. Jason decided on a two-phase evaluation. The first evaluation was to be an interactive one conducted by Carly Edwards, her marketing managers, selected staff members, and selected store managers. They worked hands-on with the prototype and identified several ways to improve it. Jason modified the HTML prototype to reflect the changes suggested by the group and asked Carly to review it again.
FIGURE 9-21
Sample interfaces from the Tune Source design prototype:
(a) list by artist (interface 1.2)
(b) information for one tune (interface 4)

<table>
<thead>
<tr>
<th>Song Title</th>
<th>Artist</th>
<th>Album</th>
<th>Time</th>
<th>Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Don't Know Why</td>
<td>Norah Jones</td>
<td>Come Away With Me</td>
<td>3:06</td>
<td>$0.88</td>
</tr>
<tr>
<td>2. Come Away With Me</td>
<td>Norah Jones</td>
<td>Come Away With Me</td>
<td>3:18</td>
<td>$0.88</td>
</tr>
</tbody>
</table>

Composer: Norah Jones
Release Date: Feb, 26, 2002
Format(s): MP3, WMA
Size: 3.6 MB

Listen: Add to Favorites Add to Cart Buy Now

(a)

(b)
The second evaluation was again interactive, this time performed by a series of two focus groups of potential customers, one with little Internet experience, the other with extensive Internet experience. Once again, several minor changes were identified. Jason again modified the HTML prototype and asked Carly to review it once more. When she was satisfied, the interface design was complete.

**SUMMARY**

*User Interface Design Principles*

The first element of the user interface design is the layout of the screen, form, or report, which is usually depicted by rectangular shapes with a top area for navigation, a central area for inputs and outputs, and a status line at the bottom. The design should help the user be aware of content and context, both between different parts of the system as they navigate through it and within any one form or report. All interfaces should be aesthetically pleasing (not necessarily works of art) and need to include significant white space, use colors carefully, and be consistent with fonts. Most interfaces should be designed to support both novice or first-time users as well as experienced users. Consistency in design (both within the system and across other systems used by the users) is important for the navigation controls, terminology, and layout of forms and reports. Finally, all interfaces should attempt to minimize user effort, for example, by requiring no more than three clicks from the main menu to perform an action.

*The User Interface Design Process*

First, analysts develop use scenarios describing commonly used patterns of actions that the users will perform. Second, they design the interface structure via an ISD based on the DFD. The ISD is then tested with the use scenarios to ensure that it enables users to quickly and smoothly perform these scenarios. Third, analysts define the interface standards in terms of interface metaphor(s), objects, actions, and icons. These elements are drawn together by the design of a basic interface template for each major section of the system. Fourth, the designs of the individual interfaces are prototyped, either through a simple storyboard, an HTML prototype, or a prototype in the development language of the system itself (e.g., Visual Basic). Finally, interface evaluation is conducted by heuristic evaluation, walk-through evaluation, interactive evaluation, or formal usability testing. This evaluation almost always identifies improvements, so the interfaces are redesigned and evaluated further.

*Navigation Design*

The fundamental goal of the navigation design is to make the system as simple to use as possible by preventing the user from making mistakes, simplifying the recovery from mistakes, and using a consistent grammar order (usually object–action order). Command languages, natural languages, and direct manipulation are used in navigation, but the most common approach is menus, (menu bar, drop-down menu, hyperlink menu, embedded hyperlinks, pop-up menu, tab menu, buttons and toolbars, and image maps). Error messages, confirmation messages, acknowledgment messages, delay messages, and help messages are common types of messages.

*Input Design*

The goal of input design is to simply and easily capture accurate information for the system, typically by using online or batch processing, capturing data at the source,
and minimizing keystrokes. Input design includes both the design of input screens and all preprinted forms that are used to collect data before they are entered into the information system. There are many types of inputs, such as text fields, number fields, check boxes, radio buttons, on-screen list boxes, drop-down list boxes, and sliders. Most inputs are validated by some combination of completeness checks, format checks, range checks, check digits, consistency checks, and database checks.

Output Design
The goal of output design is to present information to users so that they can accurately understand it with the least effort, usually by understanding how reports will be used and designing them to minimize information overload and bias. Output design means designing both screens and reports in other media, such as paper and the Web. There are many types of reports, including detail reports, summary reports, exception reports, turnaround documents, and graphs.

KEY TERMS

Acknowledgment message
Action–object order
Aesthetics
Bar code reader
Batch processing
Batch report
Bias
Button
Check box
Check digit check
Combo box
Command language
Completeness check
Confirmation message
Consistency
Consistency check
Content awareness
Context-sensitive help
Database check
Data-entry operator
Default value
Delay message
Density
Detail report
Direct manipulation
Drop-down list box
Drop-down menu
Ease of learning
Ease of use
Edit check
Error message
Exception message
Field
Field label

Form
Format check
Grammar order
Graph
Graphical user interface (GUI)
Help message
Heuristic evaluation
Hot key
HTML prototype
Human-computer interaction (HCI)
Image map
Information load
Input mechanism
Interactive evaluation
Interface action
Interface design prototype
Interface evaluation
Interface icon
Interface metaphor
Interface object
Interface standards
Interface structure design
Interface structure diagram (ISD)
Interface template
Language prototype
Layout
Magnetic stripe readers
Menu
Menu bar
Natural language
Navigation mechanism
Number box
Object-action order
Online processing

On-screen list box
Optical character recognition
Output mechanism
Pop-up menu
Radio button
Range check
Real-time information
Real-time report
Report
RFID tag
Screen
Selection box
Slider
Smart card
Source data automation
Storyboard
Summary report
System interface
Tab menu
Text box
Three-clicks rule
Toolbar
Tool tip
Transaction processing
Turnaround document
Usability testing
Use scenario
User experience
User interface
Validation
Walk-through evaluation
Web page
White space
**QUESTIONS**

1. Explain three important user interface design principles.
2. What are three fundamental parts of most user interfaces?
3. Why is content awareness important?
4. What is white space, and why is it important?
5. Under what circumstances should densities be low? high?
6. How can a system be designed to be used by both experienced and first-time users?
7. Why is consistency in design important? Why can too much consistency cause problems?
8. How can different parts of the interface be consistent?
9. Describe the basic process of user interface design.
10. What are use scenarios, and why are they important?
11. What is an interface structure diagram (ISD), and why is it used?
12. Why are interface standards important?
13. Explain the purpose and contents of interface metaphors, interface objects, interface actions, interface icons, and interface templates.
14. Why do we prototype the user interface design?
15. Compare and contrast the three types of interface design prototypes.
16. Why is it important to perform an interface evaluation before the system is built?
17. Compare and contrast the four types of interface evaluation.
18. Under what conditions is heuristic evaluation justified?
19. What type of interface evaluation did you perform in the “Your Turn 9.1”?
20. Describe three basic principles of navigation design.
21. How can you prevent mistakes?
22. Explain the differences between object–action order and action–object order.
23. Describe four types of navigation controls.
24. Why are menus the most commonly used navigation control?
25. Compare and contrast four types of menus.
26. Under what circumstances would you use a drop-down menu versus a tab menu?
27. Describe five types of messages.
28. What are the key factors in designing an error message?
29. What is context-sensitive help? Does your word processor have context-sensitive help?
30. Explain three principles in the design of inputs.
31. Compare and contrast batch processing and online processing. Describe one application that would use batch processing and one that would use online processing.
32. Why is capturing data at the source important?
33. Describe four devices that can be used for source data automation.
34. Describe five types of inputs.
35. Compare and contrast check boxes and radio buttons. When would you use one versus the other?
36. Compare and contrast on-screen list boxes and drop-down list boxes. When would you use one versus the other?
37. Why is input validation important?
38. Describe five types of input validation methods.
39. Explain three principles in the design of outputs.
40. Describe five types of outputs.
41. When would you use electronic reports rather than paper reports, and vice versa?
42. What do you think are three common mistakes that novice analysts make in interface design?
43. How would you improve the form in Fig. 9-4?

**EXERCISES**

A. Develop two use scenarios for a Web site that sells some retail products (e.g., books, music, clothes).
B. Draw an ISD for a Web site that sells some retail products (e.g., books, music, clothes).
C. Describe the primary components of the interface standards for a Web site that sells some retail products (metaphors, objects, actions, icons, and templates).
D. Develop two use scenarios for the DFD in exercise C in Chapter 5.
E. Draw an ISD for the DFD in exercise C in Chapter 5.
F. Develop the interface standards (omitting the interface template) for the DFD in exercise C in Chapter 5.
G. Develop two use scenarios for the DFD in exercise E in Chapter 5.
H. Develop the interface standards (omitting the interface template) for the DFD in exercise E in Chapter 5.
I. Design an interface template for Exercise E.
J. Draw an ISD for the DFD in Exercise E in Chapter 5.
K. Design a storyboard for Exercise E in Chapter 5.
L. Develop an HTML, prototype for Exercise E in this chapter.
M. Develop an HTML prototype for Exercise F in this chapter.
N. Develop the interface standards (omitting the interface template) for the DFD in Exercise I in Chapter 5.
O. Draw an ISD for the DFD in Exercise I in Chapter 5.
P. Design a storyboard for Exercise I in Chapter 5.
Q. Develop two use scenarios for the DFD in Exercise I in Chapter 5.
R. Ask Jeeves (http://www.askjeeves.com) is an Internet search engine that uses natural language. Experiment with it and compare it with search engines that use key words.
S. Draw an ISD for “Your Turn 9.7,” using the opposite grammar order from your original design. (If you didn’t do the “Your Turn” exercise, draw two ISDs, one in each grammar order.) Which is “best”? Why?
T. In “Your Turn 9.7” you probably used menus. Design the navigation system again, using a command language.

MINICASES

1. Tots to Teens is a catalog retailer specializing in children’s clothing. A project has been underway to develop a new order-entry system for the company’s catalog clerks. The old system had a character-based user interface that corresponded to the system’s COBOL underpinnings. The new system will feature a graphical user interface more in keeping with up-to-date PC products in use today. The company hopes that this new user interface will help reduce the turnover they have experienced with their order-entry clerks. Many newly hired order entry staff found the old system very difficult to learn and were overwhelmed by the numerous mysterious codes that had to be used to communicate with the system.

   A user interface walk-through evaluation was scheduled for today to give the users a first look at the new system’s interface. The project team was careful to invite several key users from the order-entry department. In particular, Norma was included because of her years of experience with the order-entry system. Norma was known to be an informal leader in the department; her opinion influenced many of her associates. Norma had let it be known that she was less than thrilled with the ideas she had heard for the new system. Due to her experience and good memory, Norma worked very effectively with the character-based system and was able to breeze through even the most convoluted transactions with ease. Norma had trouble suppressing a sneer when she heard talk of such things as “icons” and “buttons” in the new user interface.

   Cindy was also invited to the walk-through because of her influence in the order-entry department. Cindy has been with the department for just one year, but she quickly became known because of her successful organization of a sick-child day-care service for the children of the department workers. Sick children are the number-one cause of absenteeism in the department, and many of the workers could not afford to miss workdays. Never one to keep quiet when a situation needed improvement, Cindy has been a vocal supporter of the new system.

   a. Drawing upon the design principles presented in the text, describe the features of the user interface that will be most important to experienced users like Norma.
   b. Drawing upon the design principles presented in the text, describe the features of the user interface that will be most important to novice users like Cindy.

2. The members of a systems development project team have gone out for lunch together, and as often happens, the conversation has turned to work. The team has been working on the development of the user interface design, and so far, work has been progressing smoothly. The team should be completing work on the interface prototypes early next week. A combination of storyboards and language prototypes has been used in this project. The storyboards depict the overall structure and flow of the system, but the team developed language prototypes of the actual screens because they felt that seeing the actual screens would be valuable for the users.

   Chris (the youngest member of the project team): I read an article last night about a really cool way to evaluate a user interface design. It’s called usability testing, and it’s done by all the major software vendors. I think we should use it to evaluate our interface design.

   Heather (system analyst): I’ve heard of that, too, but isn’t it really expensive?

   Mark (project manager): I’m afraid it is expensive, and I’m not sure we can justify the expense for this project.
Chris: But we really need to know that the interface works. I thought this usability testing technique would help us prove we have a good design.

Amy (systems analyst): It would, Chris, but there are other ways, too. I assumed we'd do a thorough walkthrough with our users and present the interface to them at a meeting. We can project each interface screen so that the users can see it and give us their reaction. This is probably the most efficient way to get the users' response to our work.

Heather: That's true, but I'd sure like to see the users sit down and work with the system. I've always learned a lot by watching what they do, seeing where they get confused, and hearing their comments and feedback.

Ryan (systems analyst): It seems to me that we've put so much work into this interface design that all we really need to do is review it ourselves. Let's just make a list of the design principles we're most concerned about and check it ourselves to make sure we've followed them consistently. If we have, we should be fine. We want to get moving on the implementation, you know.

Mark: These are all good ideas. It seems like we've all got a different view of how to evaluate the interface design. Let's try and sort out the technique that is best for our project.

Develop a set of guidelines that can help a project team like the one discussed here select the most appropriate interface evaluation technique for their project.

3. The menu structure for Holiday Travel Vehicle's existing character-based system is shown here. Develop and prototype a new interface design for the system's functions, using a graphical user interface. Assume that the new system will need to include the same functions as those shown in the menus provided. Include any messages that will be produced as a user interacts with your interface (error, confirmation, status, etc.). Also, prepare a written summary that describes how your interface implements the principles of good interface design as presented in the textbook.
4. One aspect of the new system under development at Holiday Travel Vehicles will be the direct entry of the sales invoice into the computer system by the salesperson as the purchase transaction is being completed. In the current system, the salesperson fills out the paper form shown here.

Design and prototype an input screen that will permit the salesperson to enter all the necessary information for the sales invoice. The following information may be helpful in your design process: Assume that Holiday Travel Vehicles sells recreational vehicles and trailers from four different manufacturers. Each manufacturer has a fixed number of names and models of RVs and trailers. For the purposes of your prototype, use this format:

<table>
<thead>
<tr>
<th>Mfg-A Name-1 Model-X</th>
<th>Mfg-C Name-1 Model-Z</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mfg-A Name-1 Model-Y</td>
<td>Mfg-C Name-1 Model-Z</td>
</tr>
<tr>
<td>Mfg-A Name-1 Model-Z</td>
<td>Mfg-C Name-1 Model-Z</td>
</tr>
<tr>
<td>Mfg-B Name-1 Model-X</td>
<td>Mfg-C Name-2 Model-X</td>
</tr>
<tr>
<td>Mfg-B Name-1 Model-Y</td>
<td>Mfg-C Name-3 Model-X</td>
</tr>
<tr>
<td>Mfg-B Name-1 Model-Z</td>
<td>Mfg-D Name-1 Model-X</td>
</tr>
<tr>
<td>Mfg-B Name-2 Model-Y</td>
<td>Mfg-D Name-2 Model-X</td>
</tr>
<tr>
<td>Mfg-B Name-2 Model-Z</td>
<td>Mfg-D Name-2 Model-X</td>
</tr>
</tbody>
</table>

Also, assume that there are 10 different dealer options which could be installed on a vehicle at the customer’s request. The company currently has 10 salespeople on staff.

---

**Holiday Travel Vehicles**  
**Sales Invoice**  
Invoice #: ____________  
Invoice Date: ____________

Customer Name: ______________________________________
Address: ______________________________________
City: ______________________________________
State: ______________________________________
Zip: ______________________________________
Phone: ______________________________________

New RV/TRAILER  
(Circle one)  
Name: ______________________________________
Model: ______________________________________
Serial #: ____________________ Year: _________
Manufacturer: ______________________________________

Trade-in RV/TRAILER  
(Circle one)  
Name: ______________________________________
Model: ______________________________________
Year: ______________________________________
Manufacturer: ______________________________________

Options:  
<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
<th>Price</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Vehicle Base Cost: ____________________  
Trade-in Allowance: ____________________  
(Salesperson Name)
Total Options: ____________________
Tax: ____________________
License Fee: ____________________
Final Cost: ____________________  
(Customer Signature)