2.1 Introduction

Successful designers of interactive systems know that they can and must go beyond intuitive judgments made hastily when a design problem emerges. Fortunately, guidance for designers is available in the form of (1) specific and practical guidelines, (2) middle-level principles, and (3) high-level theories and models. The practical guidelines prescribe cures for design problems, caution against dangers, and provide helpful reminders based on accumulated wisdom. The middle-level principles help in analyzing and comparing design alternatives. For developers of high-level theories and models, the goal is to describe objects and actions with consistent terminology so that comprehensible explanations can be made to support communication and teaching. Other theories are predictive, such as those for reading, typing, or pointing times.

In many contemporary systems, there is a grand opportunity to improve the user interface. The cluttered displays, complex and tedious procedures, inadequate functionality, inconsistent sequences of actions, and insufficient informative feedback can generate debilitating stress and anxiety. It is perfectly understandable that users whose network connections drop as they are completing lengthy online purchase orders may become frustrated and even angry. These experiences can lead to poor performance, frequent minor slips, and occasional serious errors, all contributing to job dissatisfaction and consumer frustration. Guidelines, principles, and theories—which can provide remedies and preventive medicine for these problems—have matured in recent years. Reliable methods for predicting pointing and input times (Chapter 9), helpful cognitive theories (Chapter 11), and better frameworks for online help (Chapter 13) now shape research and guide design.
This chapter begins with a sampling of guidelines for navigating, organizing displays, getting user attention, and facilitating data entry (Section 2.2). Then Section 2.3 covers some fundamental principles of interface design, such as coping with user skill levels, task profiles, and interaction styles. It offers eight golden rules of interface design, explores ways of preventing user errors, and closes with a section on the controversial strategies for integrating automation with human control. Section 2.4 reviews several theories of interface design, and Section 2.5 concentrates on the object-action interface model.

2.2 Guidelines

From the earliest days, interface designers have tried to write down guidelines to record their insights and guide the efforts of future designers. The early Apple and Microsoft guidelines, which were influential for desktop-interface designers, have been followed by dozens of guidelines documents for the Web and a few for newer mobile devices (see the list at the end of Chapter 1). A guidelines document helps by developing a shared language and then promoting consistency among multiple designers in terminology, appearance, and action sequences. It records best practices derived from practical experience or empirical studies with appropriate examples and counterexamples. The creation of a guidelines document engages the design community in a lively discussion of input or output formats, action sequences, terminology, and hardware devices (see Section 3.3.1). Another source of design guidelines is the work of graphics designers (Mullet and Sano, 1995; Lynch and Horton, 1999; Galitz, 2003), whose dos and don’ts record current styles.

Critics complain that guidelines can be too specific, incomplete, hard to apply, and sometimes wrong. Proponents argue that building on experience from design leaders contributes to steady improvements. Both groups recognize the value of lively discussions in promoting awareness. The following four sections provide examples of guidelines, and Section 3.3 discusses how they can be integrated into the design process. The examples address some key topics, but they merely sample the thousands of guidelines that have been written.

2.2.1 Navigating the interface

Since navigation can be difficult for many users, providing clear rules is helpful. This sample of guidelines comes from the National Cancer Institute's effort to assist government agencies with design of informative web pages, but these guidelines have widespread application (Koyani et al., 2003; http://www.usability.gov). Most are stated positively ("reduce the user's workload"), but some are negative ("do not display unsolicited windows or graphics"). The National Cancer Institute's 388 guidelines, which are backed by research findings, cover the design process,
general principles, and specific rules. This sample of the guidelines gives useful advice and a taste of their style:

- **Standardize task sequences.** Allow users to perform tasks in the same sequence and manner across similar conditions.
- **Ensure that embedded links are descriptive.** When using embedded links, the link text should accurately describe the link's destination.
- **Use unique and descriptive headings.** Use headings that are unique from one another and conceptually related to the content they describe.
- **Use check boxes for binary choices.** Provide a check box control for users to make a choice between two clearly distinguishable states, such as “on” or “off.”
- **Develop pages that will print properly.** If users are likely to print one or more pages, develop pages with widths that print properly.
- **Use thumbnail images to preview larger images.** When viewing full-size images is not critical, first provide a thumbnail of the image.

These guidelines are clarified by examples and supported by research studies. A goal for guidelines writers is to be clear and comprehensible, using meaningful examples. However, controversies over guidelines are lively, often leading to revisions and the creation of alternatives.

Guidelines to promote accessibility for users with disabilities were included in the U. S. Rehabilitation Act Amendments of 1998. Its Section 508, with guidelines for web design, are published by the Access Board (http://www.access-board.gov/508.htm), an independent U. S. government agency devoted to accessibility for people with disabilities. In 1999, the World Wide Web Consortium (W3C) adapted these guidelines (http://www.w3.org/TR/WCAG10/) and organized them into three priority levels for which they provided automated checking tools. A few of the Priority 1 Accessibility Guidelines are:

- Provide a text equivalent for every non-text element (for example, via “alt”, “longdesc”, or in the element content), including images, graphical representations of text (including symbols), image map regions, animations (such as animated GIFs), applets and programmatic objects, ASCII art, frames, scripts, images used as list bullets, spacers, graphical buttons, sounds (played with or without user interaction), stand-alone audio files, audio tracks of video, and video.
- For any time-based multimedia presentation (for example, a movie or animation), synchronize equivalent alternatives, such as captions or auditory descriptions of the visual track, with the presentation.
- Ensure that all information conveyed with color is also available without color—for example, from context or markup.
- Title each frame to facilitate frame identification and navigation.
The goal of these guidelines is to have web-page designers use features that permit users with disabilities to employ screen readers or other special technologies to give them access to web-page content.

2.2.2 Organizing the display

Display design is a large topic with many special cases. Smith and Mosier (1986) offer five high-level goals as part of their guidelines for data display:

1. **Consistency of data display.** During the design process, the terminology, abbreviations, formats, colors, capitalization, and so on should all be standardized and controlled by use of a written (or computer-managed) dictionary of these items.

2. **Efficient information assimilation by the user.** The format should be familiar to the operator and should be related to the tasks required to be performed with the data. This objective is served by rules for neat columns of data, left justification for alphanumeric data, right justification of integers, lining up of decimal points, proper spacing, use of comprehensible labels, and appropriate measurement units and numbers of decimal digits.

3. **Minimal memory load on the user.** Users should not be required to remember information from one screen for use on another screen. Tasks should be arranged such that completion occurs with few actions, minimizing the chance of forgetting to perform a step. Labels and common formats should be provided for novice or intermittent users.

4. **Compatibility of data display with data entry.** The format of displayed information should be linked clearly to the format of the data entry. Where possible and appropriate, the output fields should also act as editable input fields.

5. **Flexibility for user control of data display.** Users should be able to get the information from the display in the form most convenient for the task on which they are working. For example, the order of columns and sorting of rows should be easily changeable by the users.

This compact set of high-level objectives is a useful starting point, but each project needs to expand these into application-specific and hardware-dependent standards and practices. For example, these generic guidelines, which emerged from a report on design of control rooms for electric-power utilities (Lockheed, 1981), remain valid:

- Be consistent in labeling and graphic conventions.
- Standardize abbreviations.
- Use consistent formatting in all displays (headers, footers, paging, menus, and so on).
Chapter 2  Guidelines, Principles, and Theories

- Present data only if they assist the operator.
- Present information graphically where appropriate by using widths of lines, positions of markers on scales, and other techniques that relieve the need to read and interpret alphanumeric data.
- Present digital values only when knowledge of numerical values is necessary and useful.
- Use high-resolution monitors and maintain them to provide maximum display quality.
- Design a display in monochromatic form using spacing and arrangement for organization and then judiciously add color where it will aid the operator.
- Involve users in the development of new displays and procedures.

Chapter 12 further discusses data-display issues.

2.2.3 Getting the user’s attention

Since substantial information may be presented to users for the normal performance of their work, exceptional conditions or time-dependent information must be presented so as to attract attention (Wickens and Hollands, 2000). These guidelines detail several techniques for getting the user’s attention:

- **Intensity.** Use two levels only, with limited use of high intensity to draw attention.
- **Marking.** Underline the item, enclose it in a box, point to it with an arrow, or use an indicator such as an asterisk, bullet, dash, plus sign, or X.
- **Size.** Use up to four sizes, with larger sizes attracting more attention.
- **Choice of fonts.** Use up to three fonts.
- **Inverse video.** Use inverse coloring.
- **Blinking.** Use blinking displays (2–4 Hz) or blinking color changes with great care and in limited areas.
- **Color.** Use up to four standard colors, with additional colors reserved for occasional use.
- **Audio.** Use soft tones for regular positive feedback and harsh sounds for rare emergency conditions.

A few words of caution are necessary. There is a danger in creating cluttered displays by overusing these techniques. Some web designers use blinking advertisements or animated icons to attract attention, but users almost universally disapprove. Animation is appreciated primarily when it provides meaningful information, such as for a progress indicator. Novices need simple, logically organized, and well-labeled displays that guide their actions. Expert users prefer limited labels on fields so data values are easier to extract; subtle
highlighting of changed values or positional presentation is sufficient. Display formats must be tested with users for comprehensibility.

Similarly, highlighted items will be perceived as being related. Color-coding is especially powerful in linking related items, but this use makes it more difficult to cluster items across color codes (see Section 12.6). User control over highlighting—for example, allowing the operator in an air-traffic-control environment to assign orange to images of aircraft above 18,000 feet—may provide a useful resolution to concerns about personal preferences.

Audio tones, like the clicks in keyboards or ringing sounds in telephones, can provide informative feedback about progress. Alarms for emergency conditions do alert users rapidly, but a mechanism to suppress alarms must be provided. If several types of alarms are used, testing is necessary to ensure that users can distinguish between the alarm levels. Prerecorded or synthesized voice messages are an intriguing alternative, but since they may interfere with communications between operators, they should be used cautiously (see Section 9.4).

2.2.4 Facilitating data entry

Data-entry tasks can occupy a substantial fraction of the users' time and can be the source of frustrating and potentially dangerous errors. Smith and Mosier (1986) offer five high-level objectives as part of their guidelines for data entry:

1. **Consistency of data-entry transactions.** Similar sequences of actions should be used under all conditions; similar delimiters, abbreviations, and so on should be used.

2. **Minimal input actions by user.** Fewer input actions mean greater operator productivity and—usually—fewer chances for error. Making a choice by a single keystroke, mouse selection, or finger press, rather than by typing in a lengthy string of characters, is potentially advantageous. Selecting from a list of choices eliminates the need for memorization, structures the decision-making task, and eliminates the possibility of typographic errors. However, if users must move their hands from a keyboard to a separate input device, the advantage is negated, because home-row position is lost. Expert users often prefer to type six to eight characters instead of moving to a mouse, joystick, or other selection device.

A second aspect of this guideline is that redundant data entry should be avoided. It is annoying for users to enter the same information in two locations, since the double entry is perceived as a waste of effort and an opportunity for error. When the same information is required in two places, the system should copy the information for the user, who should still have the option of overriding by retying.

3. **Minimal memory load on users.** When doing data entry, users should not be required to remember lengthy lists of codes and complex syntactic command strings.
4. **Compatibility of data entry with data display.** The format of data-entry information should be linked closely to the format of displayed information.

5. **Flexibility for user control of data entry.** Experienced data-entry operators may prefer to enter information in a sequence that they can control. For example, on some occasions in an air-traffic-control environment, the arrival time is the prime field in the controller's mind; on other occasions, the altitude is the prime field. However, flexibility should be used cautiously, since it goes against the consistency principle.

Guidelines documents are a wonderful starting point to give designers the benefit of experience, but they will always need management processes to facilitate education, enforcement, exemption, and enhancement (see Section 3.3.1).

## 2.3 Principles

While guidelines are narrowly focused, principles tend to be more fundamental, widely applicable, and enduring. However, they also tend to need more clarification. For example, the principle of recognizing user diversity makes sense to every designer, but it must be thoughtfully interpreted. A preschooler playing an animated computer game is a long way from a reference librarian doing bibliographic searches for anxious and hurried patrons. Similarly, a grandmother sending a text message is a long way from a highly trained and experienced air-traffic controller. These sketches highlight the differences in users' background knowledge, training in the use of the system, frequency of use, and goals, as well as in the impact of a user error. Since no single design could satisfy all these users and situations, successful designers must characterize their users and the situations in which their products will be used as precisely and completely as possible.

Section 1.5 offered an introduction to the variety of individual differences that a designer must address to work towards universal usability. This section focuses on a few fundamental principles, beginning with accommodating user skill levels and profiling tasks and user needs. We then discuss the five primary interaction styles (direct manipulation, menu selection, form fillin, command language, and natural language) and the "eight golden rules of interface design,” followed by a section on error prevention. Finally, we cover the controversial strategies for integrating automation with human control.

### 2.3.1 Determine users' skill levels

"Know thy user" was the first principle in Hansen's (1971) classic list of user-engineering principles. It is a simple idea but a difficult and, unfortunately, often
undervalued goal. No one would argue against this principle, but many designers assume that they understand the users and users’ tasks. Successful designers are aware that other people learn, think, and solve problems in different ways. Some users really do prefer to deal with tables rather than with graphs, with words instead of numbers, or with a rigid structure rather than an open-ended form.

All design should begin with an understanding of the intended users, including population profiles that reflect age, gender, physical and cognitive abilities, education, cultural or ethnic background, training, motivation, goals, and personality. There are often several communities of users for an interface, especially for web applications and mobile devices, so the design effort is multiplied. Typical user communities—such as nurses, doctors, storekeepers, high-school students, or librarians—can be expected to have various combinations of knowledge and usage patterns. Users from different countries may each deserve special attention, and regional differences often exist within countries. Other variables that characterize users include location (for example, urban versus rural), economic profile, disabilities, and attitudes towards using technology. Users with poor reading skills, limited education, and low motivation require special attention.

In addition to these profiles, an understanding of users’ skills with interfaces and with the application domain is important. Users might be tested for their familiarity with interface features such as traversing hierarchical menus or drawing tools. Other tests might cover domain-specific abilities such as knowledge of airport city codes, stockbrokerage terminology, insurance-claim concepts, or map icons.

The process of getting to know the users is never-ending because there is so much to know and because the users keep changing. Every step in understanding the users and in recognizing them as individuals with outlooks different from the designer’s own is likely to be a step closer to a successful design.

For example, a generic separation into novice or first-time, knowledgeable intermittent, and expert frequent users might lead to these differing design goals:

- **Novice or first-time users.** True novice users—for example, grandparents sending their first e-mail to a grandchild—are assumed to know little of the task or interface concepts. By contrast, first-time users are professionals who know the task concepts but have shallow knowledge of the interface concepts (for example, a business traveler using a rental car’s navigation system). Both groups of users may arrive with learning-inhibiting anxiety about using computers. Overcoming these limitations, via instructions, dialog boxes, and online help, is a serious challenge to the designer of the interface. Restricting vocabulary to a small number of familiar, consistently used concept terms is essential to begin developing the user’s knowledge. The number of actions should also be small, so that novice and first-time users can carry out simple
tasks successfully and thus reduce anxiety, build confidence, and gain positive reinforcement. Informative feedback about the accomplishment of each task is helpful, and constructive, specific error messages should be provided when users make mistakes. Carefully designed user manuals, video demonstrations, and task-oriented online tutorials may be effective.

- **Knowledgeable Intermittent Users.** Many people are knowledgeable but intermittent users of a variety of systems—for example, corporate managers using word processors to create templates for travel reimbursements. They have stable task concepts and broad knowledge of interface concepts, but they may have difficulty retaining the structure of menus or the location of features. The burden on their memories will be lightened by orderly structure in the menus, consistent terminology, and high interface apparenty, which emphasizes recognition rather than recall. Consistent sequences of actions, meaningful messages, and guides to frequent patterns of usage will help knowledgeable intermittent users to rediscover how to perform their tasks properly. These features will also help novices and some experts, but the major beneficiaries are knowledgeable intermittent users. Protection from danger is necessary to support relaxed exploration of features or usage of partially forgotten action sequences. These users will benefit from context-dependent help to fill in missing pieces of task or interface knowledge. Well-organized reference manuals are also useful.

- **Expert Frequent Users.** Expert “power” users are thoroughly familiar with the task and interface concepts and seek to get their work done quickly. They demand rapid response times, brief and nondistracting feedback, and the shortcuts to carry out actions with just a few keystrokes or selections. When a sequence of three or four actions is performed regularly, frequent users are willing to create a macro or other abbreviated form to reduce the number of steps. Strings of commands, shortcuts through menus, abbreviations, and other accelerators are requirements.

The characteristics of these three classes of usage must be refined for each environment. Designing for one class is easy; designing for several is much more difficult.

When multiple usage classes must be accommodated in one system, the basic strategy is to permit a **multi-layer** (sometimes called **level-structured** or **spiral**) approach to learning. Novices can be taught a minimal subset of objects and actions with which to get started. They are most likely to make correct choices when they have only a few options and are protected from making mistakes—that is, when they are given a **training-wheels** interface. After gaining confidence from hands-on experience, these users can choose to progress to ever-greater levels of task concepts and the accompanying interface concepts. The learning plan should be governed by the users’ progress through the task concepts, with
new interface concepts being chosen when they are needed to support a more complex task. For users with strong knowledge of the task and interface concepts, rapid progress is possible.

For example, novice users of a cell phone can quickly learn to make/receive calls first, then to use the menus, and later to store numbers for frequent callees. Their progress is governed by the task domain, rather than by an alphabetical list of commands that are difficult to relate to the tasks. The multi-layer approach must be carried out in the design of not only the software, but also the user manuals, help screens, error messages, and tutorials (McGrenere, Baekker, and Booth, 2002; Shneiderman, 2003). Multi-layer designs seem to be the most promising approach to promoting universal usability.

Another component of accommodating different usage classes is to permit user control of the density of informative feedback that the system provides. Novices want more informative feedback to confirm their actions, whereas frequent users want less distracting feedback. Similarly, it seems that frequent users like displays to be more densely packed than do novices. Finally, the pace of interaction may be varied from slow for novices to fast for frequent users.

### 2.3.2 Identify the tasks

After carefully drawing the user profile, the developers must identify the tasks to be carried out. Every designer would agree that the set of tasks must be determined before design can proceed, but too often the task analysis is done informally or implicitly. Task analysis has a long, mixed history (Bailey, 1996; Hackos and Redish, 1998), but successful strategies usually involve long hours of observing and interviewing users. This helps designers to understand task frequencies and sequences and make the tough decisions about what tasks to support. Some implementers prefer to include all possible actions in the hope that some users will find them helpful, but this can cause unfortunate clutter. The Palm Pilot designers were dramatically successful because they ruthlessly limited functionality (calendar, contacts, to-do list, and notes) to guarantee simplicity.

High-level task actions can be decomposed into multiple middle-level task actions, which can be further refined into atomic actions that users execute with a single command, menu selection, and so on. Choosing the most appropriate set of atomic actions is a difficult task. If the atomic actions are too small, the users will become frustrated by the large number of actions necessary to accomplish a higher-level task. If the atomic actions are too large and elaborate, the users will need many such actions with special options, or they will not be able to get exactly what they want from the system.

The relative task frequencies are important in shaping, for example, a set of commands or a menu tree. Frequent tasks should be simple and quick to carry
out, even at the expense of lengthening some infrequent tasks. Relative frequency of use is one of the bases for making architectural design decisions. For example, in a word processor:

- Frequent actions might be performed by special keys, such as the four cursor arrows, Insert, and Delete.
- Less frequent actions might be performed by a single letter plus the Ctrl key, or by a selection from a pull-down menu—examples include underscore, bold, or save.
- Infrequent actions or complex actions might require going through a sequence of menu selections or form fillins—for example, to change the printing format or to revise network-protocol parameters.

A matrix of users and tasks can help designers sort out these issues (Fig. 2.1). In each box, the designer can put a check mark to indicate that this user carries out this task. A more precise analysis would include frequencies instead of just simple check marks. Such user-needs assessment clarifies what tasks are essential for the design and which ones could be left out to preserve system simplicity and ease of learning.

<table>
<thead>
<tr>
<th>TASK</th>
<th>Query by Patient</th>
<th>Update Data</th>
<th>Query across Patients</th>
<th>Add Relations</th>
<th>Evaluate System</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nurse</td>
<td>0.14</td>
<td>0.11</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Physician</td>
<td>0.06</td>
<td>0.04</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Supervisor</td>
<td>0.01</td>
<td>0.01</td>
<td>0.04</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Appointment personnel</td>
<td>0.26</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Medical-record maintainer</td>
<td>0.07</td>
<td>0.04</td>
<td>0.04</td>
<td>0.01</td>
<td></td>
</tr>
<tr>
<td>Clinical researcher</td>
<td></td>
<td></td>
<td></td>
<td>0.08</td>
<td></td>
</tr>
<tr>
<td>Database programmer</td>
<td></td>
<td></td>
<td>0.02</td>
<td>0.02</td>
<td>0.05</td>
</tr>
</tbody>
</table>

Figure 2.1

FREQUENCY OF TASK BY JOB TITLE
Hypothetical frequency-of-use data for a medical clinic information system. Answering queries from appointments personnel about individual patients is the highest-frequency task.
2.3.3 Choose an interaction style

When the task analysis is complete and the task objects and actions have been identified, the designer can choose from these primary interaction styles: direct manipulation, menu selection, form fillin, command language, and natural language (Box 2.1 and Box 2.2). Chapters 6 through 8 explore these styles in detail; this summary gives a brief comparative overview.

**Direct manipulation.** When a clever designer can create a visual representation of the world of action, the users' tasks can be greatly simplified, because direct manipulation of familiar objects is possible. Examples of such systems include the desktop metaphor, computer-assisted design tools, air-traffic-control systems, and games. By pointing at visual representations of objects and actions, users can carry out tasks rapidly and can observe the results immediately (for example, dragging and dropping an icon into a trash can). Keyboard entry of commands or menu choices is replaced by use of pointing devices to select from a visible set of objects and actions. Direct manipulation is appealing to novices, is easy to remember for intermittent users, and, with careful design, can be rapid for frequent users. Chapter 6 describes direct manipulation and its application.

**Menu selection.** In menu-selection systems, users read a list of items, select the one most appropriate to their task, and observe the effect. If the terminology and meaning of the items are understandable and distinct, users can accomplish their tasks with little learning or memorization and just a few actions. The greatest benefit may be that there is a clear structure to decision making, since all possible choices are presented at one time. This interaction style is appropriate for novice and intermittent users and can be appealing to frequent users if the display and selection mechanisms are rapid. For designers, menu-selection systems require careful task analysis to ensure that all functions are supported conveniently and that terminology is chosen carefully and used consistently. Advanced user-interface building tools to support menu selection provide an enormous benefit by ensuring consistent screen design, validating completeness, and supporting maintenance. Menu selection is discussed in Chapter 7.

**Form fillin.** When data entry is required, menu selection alone usually becomes cumbersome, and form fillin (also called fill in the blanks) is appropriate. Users see a display of related fields, move a cursor among the fields, and enter data where desired. With the form fillin interaction style, users must understand the field labels, know the permissible values and the data-entry method, and be capable of responding to error messages. Since knowledge of the keyboard, labels, and permissible fields is required, some training may be necessary. This interaction style is most appropriate for knowledgeable intermittent users or frequent users. Chapter 7 provides a thorough treatment of form fillin.
Advantages and disadvantages of the five primary interaction styles.

<table>
<thead>
<tr>
<th>Advantages</th>
<th>Disadvantages</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Direct manipulation</strong></td>
<td>May be hard to program</td>
</tr>
<tr>
<td>Visually presents task concepts</td>
<td>May require graphics display and pointing devices</td>
</tr>
<tr>
<td>Allows easy learning</td>
<td></td>
</tr>
<tr>
<td>Allows easy retention</td>
<td></td>
</tr>
<tr>
<td>Allows errors to be avoided</td>
<td></td>
</tr>
<tr>
<td>Encourages exploration</td>
<td></td>
</tr>
<tr>
<td>Affords high subjective satisfaction</td>
<td></td>
</tr>
<tr>
<td><strong>Menu selection</strong></td>
<td></td>
</tr>
<tr>
<td>Shortens learning</td>
<td>Presents danger of many menus</td>
</tr>
<tr>
<td>Reduces keystrokes</td>
<td>May slow frequent users</td>
</tr>
<tr>
<td>Structures decision making</td>
<td>Consumes screen space</td>
</tr>
<tr>
<td>Permits use of dialog-management tools</td>
<td>Requires rapid display rate</td>
</tr>
<tr>
<td>Allows easy support of error handling</td>
<td></td>
</tr>
<tr>
<td><strong>Form fillin</strong></td>
<td></td>
</tr>
<tr>
<td>Simplifies data entry</td>
<td>Consumes screen space</td>
</tr>
<tr>
<td>Requires modest training</td>
<td></td>
</tr>
<tr>
<td>Gives convenient assistance</td>
<td></td>
</tr>
<tr>
<td>Permits use of form-management tools</td>
<td></td>
</tr>
<tr>
<td><strong>Command language</strong></td>
<td></td>
</tr>
<tr>
<td>Is flexible</td>
<td>Has poor error handling</td>
</tr>
<tr>
<td>Appeals to &quot;power&quot; users</td>
<td>Requires substantial training and memorization</td>
</tr>
<tr>
<td>Supports user initiative</td>
<td></td>
</tr>
<tr>
<td>Allows convenient creation of user-defined macros</td>
<td></td>
</tr>
<tr>
<td><strong>Natural language</strong></td>
<td></td>
</tr>
<tr>
<td>Relieves burden of learning syntax</td>
<td>Requires clarification dialog</td>
</tr>
<tr>
<td></td>
<td>May not show context</td>
</tr>
<tr>
<td></td>
<td>May require more keystrokes</td>
</tr>
<tr>
<td></td>
<td>Is unpredictable</td>
</tr>
</tbody>
</table>
Command language. For frequent users, command languages (discussed in Chapter 8) provide a strong feeling of being in control. Users learn the syntax and can often express complex possibilities rapidly, without having to read distracting prompts. However, error rates are typically high, training is necessary, and retention may be poor. Error messages and online assistance are hard to provide because of the diversity of possibilities and the complexity of mapping from tasks to interface concepts and syntax. Command languages and lengthier query or programming languages are the domain of expert frequent users, who often derive great satisfaction from mastering a complex set of semantics and syntax. Powerful advantages include easy history keeping and simple macro creation.

Natural language. The hope that computers will respond properly to arbitrary natural-language sentences or phrases engages many researchers and system developers, in spite of limited success thus far. Natural-language interaction usually provides little context for issuing the next command, frequently requires clarification dialog, and may be slower and more cumbersome than the
alternatives. Still, where users are knowledgeable about a task domain whose scope is limited and where intermittent use inhibits command-language training, there exist opportunities for natural-language interfaces (discussed at the end of Chapter 8).

Blending several interaction styles may be appropriate when the required tasks and users are diverse. For example, commands can lead the user to a form fill-in where data entry is required, or menus can be used to control a direct-manipulation environment when a suitable visualization of actions cannot be found. Also, keyboard commands can provide shortcuts for experts who seek more rapid performance than mouse selection. Chapters 6–8 expand on the constructive guidance for using the interaction styles outlined here, and Chapter 9 describes how input and output devices influence these interaction styles. Chapter 10 deals with the relationship between interaction styles and collaborative interfaces.

2.3.4 Use the eight golden rules of interface design

This section focuses attention on eight principles, called "golden rules," that are applicable in most interactive systems. These principles, derived from experience and refined over two decades, need validation and tuning for specific design domains. No list such as this can be complete, but it has been well received as a useful guide to students and designers.

1. Strive for consistency. This rule is the most frequently violated one, but following it can be tricky because there are many forms of consistency. Consistent sequences of actions should be required in similar situations; identical terminology should be used in prompts, menus, and help screens; and consistent color, layout, capitalization, fonts, and so on should be employed throughout. Exceptions, such as required confirmation of the delete command or no echoing of passwords, should be comprehensible and limited in number.

2. Cater to universal usability. Recognize the needs of diverse users and design for plasticity, facilitating transformation of content. Novice-expect differences, age ranges, disabilities, and technology diversity each enrich the spectrum of requirements that guides design. Adding features for novices, such as explanations, and features for experts, such as shortcuts and faster pacing, can enrich the interface design and improve perceived system quality.

3. Offer informative feedback. For every user action, there should be system feedback. For frequent and minor actions, the response can be modest, whereas for infrequent and major actions, the response should be more substantial. Visual presentation of the objects of interest provides a convenient environment for showing changes explicitly (see the discussion of direct manipulation in Chapter 6).
4. **Design dialogs to yield closure.** Sequences of actions should be organized into groups with a beginning, middle, and end. Informative feedback at the completion of a group of actions gives operators the satisfaction of accomplishment, a sense of relief, the signal to drop contingency plans from their minds, and a signal to prepare for the next group of actions. For example, e-commerce websites move users from selecting products to the checkout, ending with a clear confirmation page that completes the transaction.

5. **Prevent errors.** As much as possible, design the system such that users cannot make serious errors; for example, gray out menu items that are not appropriate and do not allow alphabetic characters in numeric entry fields (see Section 2.3.5). If a user makes an error, the interface should detect the error and offer simple, constructive, and specific instructions for recovery. For example, users should not have to retype an entire name-address form if they enter an invalid zip code, but rather should be guided to repair only the faulty part. Erroneous actions should leave the system state unchanged, or the interface should give instructions about restoring the state.

6. **Permit easy reversal of actions.** As much as possible, actions should be reversible. This feature relieves anxiety, since the user knows that errors can be undone, thus encouraging exploration of unfamiliar options. The units of reversibility may be a single action, a data-entry task, or a complete group of actions, such as entry of a name and address block.

7. **Support internal locus of control.** Experienced operators strongly desire the sense that they are in charge of the interface and that the interface responds to their actions. Surprising interface actions, tedious sequences of data entries, inability to obtain or difficulty in obtaining necessary information, and inability to produce the action desired all build anxiety and dissatisfaction. Gaines (1981) captured part of this principle with his rule *avoid inaccessibility* and his encouragement to make users the initiators of actions rather than the responders to actions.

8. **Reduce short-term memory load.** The limitation of human information processing in short-term memory (the rule of thumb is that humans can remember “seven plus or minus two chunks” of information) requires that displays be kept simple, multiple-page displays be consolidated, window-motion frequency be reduced, and sufficient training time be allotted for codes, mnemonics, and sequences of actions. Where appropriate, online access to command-syntax forms, abbreviations, codes, and other information should be provided.

These underlying principles must be interpreted, refined, and extended for each environment. They have their limitations, but they provide a good starting point for mobile, desktop, or web designers. The principles presented in the ensuing sections focus on increasing the productivity of users by providing simplified
data-entry procedures, comprehensible displays, and rapid informative feedback to increase feelings of competence, mastery, and control over the system.

2.3.5 Prevent errors

There is no medicine against death, and against error no rule has been found.

Sigmund Freud
(Inscription he wrote on his portrait)

The importance of error prevention (the fifth golden rule) is so strong that it deserves its own section. Users of cell phones, e-mail, spreadsheets, air-traffic-control systems, and other interactive systems make mistakes far more frequently than might be expected. Even experienced analysts make errors in almost half their spreadsheets (Brown and Gould, 1987; Galleta et al., 1993). Other studies reveal the magnitude of the problem of and the loss of productivity due to user errors (Panko and Halverson, 1996).

One way to reduce the loss in productivity due to errors is to improve the error messages provided by the interface. Better error messages can raise success rates in repairing the errors, lowering future error rates, and increasing subjective satisfaction (Shneiderman, 1982). Superior error messages are more specific, positive in tone, and constructive (telling the user what to do, rather than merely reporting the problem). Rather than using vague (“?” or “What?”) or hostile (“Illegal Operation” or “Syntax Error”) messages, designers are encouraged to use informative messages, such as “Printer is off, please turn it on” or “Month range from 1 to 12.”

Improved error messages, however, are only helpful medicine. A more effective approach is to prevent the errors from occurring. This goal is more attainable than it may seem in many interfaces.

The first step is to understand the nature of errors. One perspective is that people make mistakes or “slips” (Norman, 1983) that designers can help them to avoid by organizing screens and menus functionally, designing commands or menu choices to be distinctive, and making it difficult for users to take irreversible actions. Norman offers other guidelines, such as providing feedback about the state of the interface (changing the cursor to show whether a map interface is in zoom-in or select mode) and designing for consistency of actions (ensuring that Yes/No buttons are always in the same order). Norman’s analysis provides practical examples and a useful theory. Additional design techniques to reduce errors include correct actions and complete sequences.

Correct actions. Industrial designers recognize that successful products must be safe and must prevent users from making dangerously incorrect use of
the products. Airplane engines cannot be put into reverse until the landing gear has touched down, and cars cannot be put into reverse while traveling forward at faster than five miles per hour. Similar principles can be applied to interactive systems—for example, graying out inappropriate menu items so they can’t be inadvertently selected, or allowing web users to simply click on the date on a calendar instead of having to type a month and date for a desired airline flight departure. Likewise, instead of having to enter a 10-digit phone number, cell phone users can scroll through a list of frequently or recently dialed phone numbers and select one with a single button. Another option used by some systems, such as the Visual Basic programming environment, is to offer automatic command completion to reduce the likelihood of user errors: the user types the first few letters of a command and the computer completes it as soon as the input is sufficient to distinguish the command from others. Techniques such as these do some of the work for the user, thereby reducing opportunities for user errors.

Complete sequences. Sometimes, an action requires several steps to reach completion. Since people may forget to complete every step of an action, designers attempt to offer a sequence of steps as a single action. In an automobile, the driver does not have to set two switches to signal a left turn. A single switch causes both (front and rear) turn-signal lights on the left side of the car to flash. When a pilot throws a switch to lower the landing gear, hundreds of steps and checks are invoked automatically. This same concept can be applied to interactive uses of computers. For example, the sequence of dialing up, setting communication parameters, logging on, and loading files is frequently executed by many users. Fortunately, most communications-software packages enable users to specify these processes once and then to execute them by simply selecting the appropriate process name.

As another example, users of a word processor can indicate that all section titles are to be centered, set in uppercase letters, and underlined, without having to issue a series of commands each time they enter a section title. Then, if users want to change the title style—for example, to eliminate underlining—a single command will guarantee that all section titles are revised consistently. As a final example, air-traffic controllers may formulate plans to change the altitude of a plane from 14,000 feet to 18,000 feet in two increments; after raising the plane to 16,000 feet, however, the controller may get distracted and may thus fail to complete the action. The controller should be able to record the plan and then have the computer prompt for completion. The notion of complete sequences of actions may be difficult to implement because users may need to issue atomic actions as well as complete sequences. In this case, users should be allowed to define sequences of their own; the macro or subroutine concept should be available at every level of usage. Designers can gather information about potential complete sequences by studying sequences of commands that people actually issue and the patterns of errors that people actually make.
Chapter 2  Guidelines, Principles, and Theories

Thinking about universal usability also contributes to reducing errors—for example, a design with too many small buttons may cause unacceptably high error rates among older users or others with limited motor control, but enlarging the buttons will benefit all users. Section 4.6.2 addresses the idea of logging user errors so designers can continuously improve designs.

2.3.6  Integrating automation while preserving human control

The guidelines and principles described in the previous sections are often devoted to simplifying the users' tasks. Users can then avoid routine, tedious, and error-prone tasks and can concentrate on making critical decisions, coping with unexpected situations, and planning future actions (Sanders and McCormick, 1993). (Box 2.3 provides a detailed comparison of human and machine capabilities.)

The degree of automation increases over time as procedures become more standardized and the pressure for productivity grows. With routine tasks, automation is desirable, since the potential for errors and the users' workload are reduced. However, even with increased automation, designers can still offer the predictable and controllable interfaces that users often prefer. The human supervisory role needs to be maintained because the real world is an open system (that is, there is a nondenumerable number of unpredictable events and system failures). By contrast, computers constitute a closed system (there is only a denumerable number of normal and failure situations that can be accommodated in hardware and software). Human judgment is necessary for unpredictable events in which some action must be taken to preserve safety, to avoid expensive failures, or to increase product quality (Hancock and Scallen, 1996).

For example, in air-traffic control, common actions include changes to altitude, heading, or speed. These actions are well understood and can potentially be automatable by a scheduling and route-allocation algorithm, but the controllers must be present to deal with the highly variable and unpredictable emergency situations. An automated system might deal successfully with high volumes of traffic, but what would happen if the airport manager closes runways because of turbulent weather? The controllers would have to reroute planes quickly. Now suppose that one pilot requests clearance for an emergency landing because of a failed engine, while another pilot reports a passen-
Box 2.3


**Humans Generally Better**
- Sense low-level stimuli
- Detect stimuli in noisy background
- Recognize constant patterns in varying situations
- Sense unusual and unexpected events
- Remember principles and strategies
- Retrieve pertinent details without *a priori* connection
- Draw on experience and adapt decisions to situation
- Select alternatives if original approach fails
- Reason inductively; generalize from observations
- Act in unanticipated emergencies and novel situations
- Apply principles to solve varied problems
- Make subjective evaluations
- Develop new solutions
- Concentrate on important tasks when overload occurs
- Adapt physical response to changes in situation

**Machines Generally Better**
- Sense stimuli outside human's range
- Count or measure physical quantities
- Store quantities of coded information accurately
- Monitor prespecified events, especially infrequent ones
- Make rapid and consistent responses to input signals
- Recall quantities of detailed information accurately
- Process quantitative data in prespecified ways
- Reason deductively; infer from a general principle
- Perform repetitive preprogrammed actions reliably
- Exert great, highly controlled physical force
- Perform several activities simultaneously
- Maintain operations under heavy information load
- Maintain performance over extended periods of time

Another example of the complexity of life-critical situations in air-traffic control emerges from an incident on a plane that had a fire on board. The controller cleared other traffic from the flight path and began to guide the plane in for a landing. The smoke was so thick that the pilot had trouble reading his instruments. Then the onboard transponder burned out, so the air-traffic controller could no longer read the plane's altitude from the situation display. In spite of these multiple failures, the controller and the pilot managed to bring down the plane quickly enough to save the lives of many—but not all—of the passengers. A computer could not have been programmed to deal with this particular unexpected series of events.
A tragic outcome of excess automation occurred during a 1995 flight to Cali, Colombia. The pilots relied on the automatic pilot and failed to realize that the plane was making a wide turn to return to a location that they had already passed. When the ground-collision alarm sounded, the pilots were too disoriented to pull up in time; they crashed 200 feet below a mountain peak, killing all but four people on board.

The goal of system design in many applications is to give operators sufficient information about current status and activities so that, when intervention is necessary, they have the knowledge and the capacity to perform correctly, even under partial failures (Sheridan, 1997; Billings, 1997). The U.S. Federal Aviation Agency stresses that designs should place the user in control and automate only to "improve system performance, without reducing human involvement" (FAA, 2003). These standards also encourage managers to "train users when to question automation."

The entire system must be designed and tested, not only for normal situations, but also for as wide a range of anomalous situations as can be anticipated. An extensive set of test conditions might be included as part of the requirements document. Operators need to have enough information that they can take responsibility for their actions. Beyond supervision of decision making and handling of failures, the role of the human operator is to improve the design of the system.

Questions of integrating automation with human control also emerge in systems for home and office automation. Many designers are eager to create an autonomous agent that knows people's likes and dislikes, makes proper inferences, responds to novel situations, and performs competently with little guidance. They believe that human-human interaction is a good model for human-computer interaction, and they seek to create computer-based partners, assistants, or agents (Berners-Lee, Hendler, and Laszlo, 2001).

The controversy is over whether to create tool-like interfaces or to pursue autonomous, adaptive, or anthropomorphic agents that carry out the users' intents and anticipate needs (Cassell et al., 2000; Gratch et al., 2002). The agent scenarios often show a responsive, butler-like human, such as the bow-tied, helpful young man in Apple Computer's 1987 video on the Knowledge Navigator. Microsoft's 1995 BOB program, which used cartoon characters to create onscreen partners, was unsuccessful; their much-criticized Clippie character was also withdrawn. Web-based characters (such as Ananova) to read the news have also faded. On the other hand, avatars representing users, not computers, in game-playing and three-dimensional social environments (see Section 6.6) have remained popular, possibly because they have a puppet-like theatrical quality.

To succeed in this path, promoters of anthropomorphic representations (see Section 12.3) of computers will have to understand and overcome the history of their unsuccessful application in the products mentioned above, as well as in bank terminals, computer-assisted instruction, talking cars, and postal-service stations. Hopeful scenarios include anthropomorphic pedagogical agents that
instruct, respond to, or guide students using natural-language interaction (Rickel and Johnson, 1997; Graesser et al., 2001; Moreno et al., 2001; and see Section 8.6.5).

A variant of the agent scenario, which does not include an anthropomorphic realization, is that the computer employs a user model to guide an adaptive interface. The system keeps track of user performance and adapts the interface to suit the users' needs. For example, when users begin to make menu selections rapidly, indicating proficiency, advanced menu items or a command-line interface should appear. Automatic adaptations have been proposed for interface features such as content of menus, order of menu items (see Section 7.5.2 for evidence against the helpfulness of this strategy), type of feedback (graphic or tabular), and content of help screens. Advocates point to video games that increase the speed or number of dangers as users progress through stages of the game. However, games are notably different from most work situations, where users have goals and motivations to accomplish their tasks.

There are some opportunities for adaptive user models to tailor system designs (such as e-mail spam filters), but even occasional unexpected behavior has serious negative effects that discourage use. If adaptive systems make surprising changes, users must pause to see what has happened. Then users may become anxious, because they may not be able to predict the next change, interpret what has happened, or restore the system to the previous state. Suggestions that users could be consulted before a change is made are helpful, but such intrusions may still disrupt problem-solving processes and annoy users. Empirical evidence has begun to clarify that the more acceptable direction is content adaptation, such as allowing users to specify that more sports stories be shown in a newspaper website (Kobsa, 2004).

An extension of user modeling is the notion of recommender systems or collaborative filtering in distributed World Wide Web applications. There is no agent or adaptation in the interface, but the system aggregates information from multiple sources in some (often proprietary) way. Such approaches have great entertainment and practical value in cases such as selecting movies, books, or music; users are often intrigued and amused to see what suggestions emerge from aggregated patterns of preferences or purchases (Riedl, Konstan, and Vrooman, 2002).

The philosophical alternative to agents and user modeling is comprehensible systems that provide consistent interfaces, user control, and predictable behavior. Designers who emphasize a direct-manipulation style believe that users have a strong desire to be in control and to gain mastery over the system, which allows them to accept responsibility for their actions and derive feelings of accomplishment (Lanier, 1995; Schneiderman, 1995). Historical evidence suggests that users seek comprehensible and predictable systems and shy away from those that are complex or unpredictable; for example, pilots may disengage automatic piloting devices if they perceive that these systems are not performing as they expect.
Another resolution of the controversy is to accept user control at the interface, but consider agent-like or multi-agent programming to automate internal processes such as disk-space allocation or network routing based on current loads. However, these are adaptations based on system features, not user profiles.

Since agent advocates promote autonomy, it seems they must take on the issue of responsibility for failures. Who is responsible when an agent violates copyright, invades privacy, or destroys data? Agent designs might be better received if they supported performance monitoring while allowing users to examine and revise the current user model.

An alternative to agents with user models may be to expand the control-panel model. Computer control panels, like automobile cruise-control mechanisms and television remote controls, are designed to convey the sense of control that users seem to expect. Users employ control panels to set physical parameters, such as the cursor blinking speed or speaker volume, and to establish personal preferences such as time/date formats or color schemes (Figs. 2.2 and 2.3). Some software packages allow users to set parameters such as the speed of play in games—users start at layer 1 and can then choose when to progress to higher levels; often they are content remaining experts at layer 1 of a complex interface rather than dealing with the uncertainties of higher layers. More elaborate control panels exist in style sheets of word processors, specification boxes of query facilities, and information-visualization tools. Similarly, scheduling software may have elaborate controls to allow users to execute planned procedures at regular intervals or when triggered by events.

2.4 Theories

One goal for the discipline of human-computer interaction is to go beyond the specifics of guidelines and build on the breadth of principles to develop tested, reliable, and broadly useful theories. Of course, for a topic as large as user-interface design, many theories are needed. Some theories are descriptive and explanatory; these theories are helpful in developing consistent terminology for objects and actions, thereby supporting collaboration and training. Some theories are predictive; these theories enable designers to compare proposed designs for execution time or error rates.

Another way to group theories is according to motor-task performance (pointing with a mouse), perceptual activities (finding an item on a display), or cognitive aspects (planning the conversion of a boldfaced character to an italic one). Motor-task performance predictions are well established and accurate for predicting keystroking or pointing times (see Fitts’s Law, Section 9.3.5). Perceptual theories have been successful in predicting reading times for free text, lists, for-
Another resolution of the controversy is to accept user control at the interface, but consider agent-like or multi-agent programming to automate internal processes such as disk-space allocation or network routing based on current loads. However, these are adaptations based on system features, not user profiles.

Since agent advocates promote autonomy, it seems they must take on the issue of responsibility for failures. Who is responsible when an agent violates copyright, invades privacy, or destroys data? Agent designs might be better received if they supported performance monitoring while allowing users to examine and revise the current user model.

An alternative to agents with user models may be to expand the control-panel model. Computer control panels, like automobile cruise-control mechanisms and television remote controls, are designed to convey the sense of control that users seem to expect. Users employ control panels to set physical parameters, such as the cursor blinking speed or speaker volume, and to establish personal preferences such as time/date formats or color schemes (Figs. 2.2 and 2.3). Some software packages allow users to set parameters such as the speed of play in games—users start at layer 1 and can then choose when to progress to higher levels; often they are content remaining experts at layer 1 of a complex interface rather than dealing with the uncertainties of higher layers. More elaborate control panels exist in style sheets of word processors, specification boxes of query facilities, and information-visualization tools. Similarly, scheduling software may have elaborate controls to allow users to execute planned procedures at regular intervals or when triggered by events.

2.4 Theories

One goal for the discipline of human-computer interaction is to go beyond the specifics of guidelines and build on the breadth of principles to develop tested, reliable, and broadly useful theories. Of course, for a topic as large as user-interface design, many theories are needed. Some theories are descriptive and explanatory; these theories are helpful in developing consistent terminology for objects and actions, thereby supporting collaboration and training. Some theories are predictive; these theories enable designers to compare proposed designs for execution time or error rates.

Another way to group theories is according to motor-task performance (pointing with a mouse), perceptual activities (finding an item on a display), or cognitive aspects (planning the conversion of a boldfaced character to an italic one). Motor-task performance predictions are well established and accurate for predicting keystroking or pointing times (see Fitts's Law, Section 9.3.5). Perceptual theories have been successful in predicting reading times for free text, lists, for-