



Future Vision

# FUTURE VISION BIE

By K B Hemanth Raj

Visit : <https://hemanthrajhemu.github.io>

## A Small Contribution Would Support Us.

Dear Viewer,

**Future Vision BIE** is a free service and so that any Student/Research Personal **Can Access Free of Cost.**

If you would like to say **thanks**, you can make a **small contribution** to the author of this site.

Contribute whatever you feel this is worth to you. This gives **us support** & to bring **Latest Study Material** to you. After the Contribution Fill out this Form (<https://forms.gle/tw3T3bUVpLXL8omX7>). To Receive a **Paid E-Course for Free**, from our End within 7 Working Days.

Regards

- K B Hemanth Raj (Admin)

### Contribution Methods

#### UPI ID

1. futurevisionbie@oksbi
2. futurevisionbie@paytm

#### Scan & Pay

#### Account Transfer

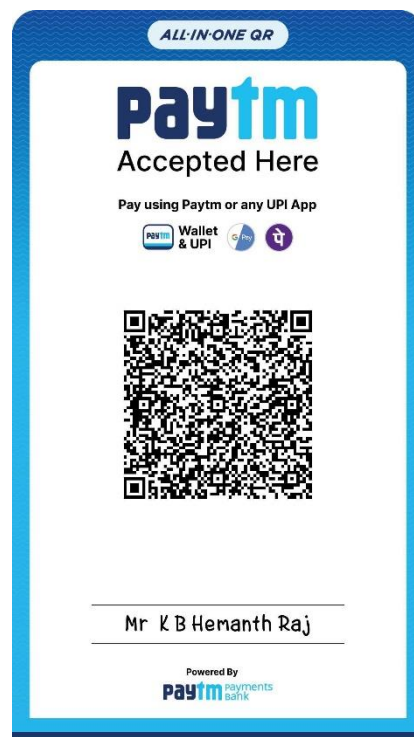
Account Holder's Name: K B Hemanth Raj

Account Number: 39979402438

IFSC Code: SBIN0003982

MICR Code: 560002017

More Info: <https://hemanthrajhemu.github.io/Contribution/>



**Gain Access to All Study Materials according to VTU,  
CSE – Computer Science Engineering,  
ISE – Information Science Engineering,  
ECE - Electronics and Communication Engineering & MORE...**

**Stay Connected... get Updated... ask your queries...**

Join Telegram to get Instant Updates: [https://bit.ly/VTU\\_TELEGRAM](https://bit.ly/VTU_TELEGRAM)

Contact: MAIL: [futurevisionbie@gmail.com](mailto:futurevisionbie@gmail.com)

INSTAGRAM: [www.instagram.com/futurevisionbie/](https://www.instagram.com/futurevisionbie/)

WHATSAPP SHARE: <https://bit.ly/FVBIESHARE>



# **The Essential Guide to User Interface Design**

## **An Introduction to GUI Design Principles and Techniques**

---

**Third Edition**

Wilbert O. Galitz



Wiley Publishing, Inc.

	Consistency	713
	Considerations for People with Color-Viewing Deficiencies	713
	Cultural, Disciplinary, and Accessibility Considerations	714
	Choosing Colors for Textual Graphic Screens	714
	Effective Foreground/Background Combinations	714
	Choose the Background First	717
	Maximum of Four Colors	717
	Use Colors in Toolbars Sparingly	718
	Test the Colors	718
	Choosing Colors for Statistical Graphics Screens	718
	Emphasis	718
	Number of Colors	718
	Backgrounds	719
	Size	719
	Status	719
	Measurements and Area-Fill Patterns	719
	Physical Impressions	720
	Choosing Colors for Web Pages	721
	Uses of Color to Avoid	723
	Step 12 Exercise	725
<b>Step 13</b>	<b>Organize and Layout Windows and Pages</b>	<b>727</b>
	Organizing and Laying Out Screens	728
	General Guidelines	728
	Organization Guidelines	729
	Control Navigation	748
	Window Guidelines	749
	Web Page Guidelines	750
	Screen Examples	761
	Example 1	761
	Example 2	762
<b>Step 14</b>	<b>Test, Test, and Retest</b>	<b>767</b>
	Usability	768
	The Purpose of Usability Testing	768
	The Importance of Usability Testing	769
	Scope of Testing	770
	Prototypes	771
	Hand Sketches and Scenarios	772
	Interactive Paper Prototypes	774
	Programmed Facades	775
	Prototype-Oriented Languages	776
	Comparisons of Prototypes	776
	Kinds of Tests	777
	Guidelines and Standards Review	779
	Heuristic Evaluation	780
	Cognitive Walk-Throughs	786

Think-Aloud Evaluations	788
Usability Test	789
Classic Experiments	790
Focus Groups	791
Choosing a Testing Method	792
Developing and Conducting a Test	795
The Test Plan	795
Test Conduct and Data Collection	803
Analyze, Modify, and Retest	806
Evaluate the Working System	807
Additional Reading	809
A Final Word	810
<b>References</b>	<b>811</b>
<b>Index</b>	<b>835</b>

# Organize and Layout Windows and Pages

During the design process to this point, the individual elements, or building blocks, of screens will have been identified, and each element's content and wording established. A logical flow of information will also have been determined. The next step is to organize and lay out individual windows and Web pages clearly and meaningfully. Proper screen presentation and structure will encourage quick and correct information comprehension, the fastest possible execution of tasks and functions, and enhanced user acceptance. In this step, we will address

- Organizing and laying out graphical and Web screens to encourage quick and accurate information comprehension and control execution.
- Organizing meaningfully and efficiently.
- Creating groupings.
- Providing alignment and balance.

In graphical user interfaces, components to be included on windows include a title, screen controls, headings, other screen content, and possibly instructional messages. On Web pages, components to be included consist of elements such as the page title, textual content, graphics, headings, screen controls, links, and other necessary components.

## Organizing and Laying Out Screens

---

How a screen is organized and how its information is actually presented are crucial to achieving the design goals of fast and accurate comprehension and control execution. Following is a summary of numerous design principles that can be applied toward these ends. They have all been fully addressed in earlier chapters, but are restated here for quick referral and as a reminder of their importance.

### General Guidelines

---

- Amount of information:
  - Present the proper amount of information on each screen.
    - Too little is inefficient.
    - Too much is confusing.
  - Present all information necessary for performing an action or making a decision on one screen, whenever possible.
- Organization and flow:
  - Divide information into units that are logical, meaningful, and sensible.
- Organize by the degree interrelationship between data or information.
- Provide an ordering that
  - Is prioritized according to the user's expectations and needs.
  - Is logical and sequential.
  - Is rhythmic, guiding a person's eye through the display.
  - Encourages natural movement sequences.
  - Minimizes pointer and eye movement distances.
- Control placement:
  - Position the most important and frequently used controls at the top left.
  - Maintain a top-to-bottom, left-to-right flow.
  - If one control enables or affects another, the enabling control should be above or to the left of the enabled control.
  - Place the command buttons that affect the entire window horizontally, and centered, at the window's bottom.
- Navigation:
  - The flow of interaction should
    - Require as little cursor and pointer travel as possible.
    - Minimize the number of times a person's hand has to travel between the keyboard and the mouse.
  - Assist users in navigating through a screen by
    - Aligning elements.
    - Grouping elements.
    - Using line borders.
- Aesthetics:
  - Provide a visually pleasing composition through
    - Adequate use of white space.
    - Balance.

- Groupings.
  - Alignment of elements.
  - Visual clutter:
    - Avoid visual clutter by
      - Maintaining low screen density levels.
      - Maintaining distinctiveness of elements.
  - Focus and emphasis:
    - Provide visual emphasis to the most important screen elements, its data or information.
    - Sequentially, direct attention to items that are
      - Critical.
      - Important.
      - Secondary.
      - Peripheral.
  - Consistency:
    - Provide consistency
      - With a person's experiences and cultural conventions.
      - Internally within a system, including
        - Operational and navigational procedures.
        - Visual identity or theme.
        - Component.
        - Organization.
        - Presentation.
        - Usage.
        - Locations.
      - Externally across systems.
- 

These guidelines, along with many others, are discussed in detail in Step 3. In brief, present the proper amount of information in each window. Never cram information into it. Keep the proportion of the window devoted to information or “ink” to no more than 30 to 40 percent of the window's entire area. Always leave a sufficient margin around all screen elements and between elements and the screen border. The window will look much more appealing to the viewer. Provide an ordering that is logical, sequential, and rhythmic to guide a person's eye through the display. Other important factors include maintaining a top-to-bottom, left-to-right flow; efficiency in navigation; a visually pleasing composition, achieved by means of balance, groupings, and alignment; the proper emphasis of elements; and being consistent.

## Organization Guidelines

Organization guidelines to be addressed include those relating to groupings, borders, dependent controls, alignment, and balance.

## ***Creating Groupings***

---

- General:
    - Provide groupings of associated elements.
      - Elements of a radio button or check box control.
      - Two or more related fields or controls.
    - Create groupings as close as possible to 5 degrees of visual angle.
  - White space:
    - Provide adequate separation of groupings through the liberal use of white space.
    - Leave adequate space
      - Around groups of related controls.
      - Between groupings and window borders.
    - The space between groupings should be greater than the space between fields within a grouping.
  - Headings:
    - Provide section headings and subsection headings for multiple control groupings.
    - Provide headings that meaningfully and concisely describe the nature of the group of related fields.
  - Borders:
    - Enhance groupings through incorporation of borders around
      - Elements of a single control.
      - Groups of related controls or fields.
    - Individual control borders should be visually differentiable from borders delineating groupings of fields or controls.
      - Provide a border consisting of a thin line around *single* controls.
      - Provide a border consisting of a slightly thicker line around *groups* of fields or controls.
    - Do not place individual field or control borders around
      - Single entry fields.
      - Single list boxes.
      - Single combination boxes.
      - Single spin boxes.
      - Single sliders.
    - Do not place borders around command buttons.
- 

Individual controls with multiple parts, such as radio buttons or check boxes, should be identifiable as a single entity. A series of related controls should also be presented as related. Create groupings to do this as often as possible. Groupings aid learning and provide visual appeal. The optimum group size is 5 degrees of visual angle. At the normal viewing distance of a screen, this is a circle 1.67 inches in diameter. On a text-based screen this is equivalent to about six to seven lines at a width of 12 to 14 characters. Examples of groupings are illustrated in Figure 13.1.



Municipality: ☐ City  
☐ Township  
☐ County  
☐ State

Building:

Floor:

Telephone:

**Figure 13.1:** Groupings.

Groupings can be made visually obvious through liberal use of white space. Sufficient space should be left between all groupings of controls, and groupings and the window borders, as illustrated in Figure 13.2.

Headings should also be used to give groupings of controls or information an identity. This aids comprehension and learning of what is presented. See Figure 13.3.

Municipality: ☐ City  
☐ Township  
☐ County  
☐ State

Building:

Floor:

Telephone:

Department: ☐ Administration  
☐ Finance  
☐ Fire  
☐ Police  
☐ Public Works  
☐ Social Services

Manager:

Employees:

Payroll:

**Figure 13.2:** Groupings using white space.

JURISDICTION	LOCATION
Municipality: <input type="radio"/> City	Building: <input type="text"/>
<input type="radio"/> Township	Floor: <input type="text"/>
<input type="radio"/> County	Telephone: <input type="text"/> <input type="text"/> <input type="text"/>
<input type="radio"/> State	
PERSONNEL	
Department: <input type="radio"/> Administration	Manager: <input type="text"/>
<input type="radio"/> Finance	Employees: <input type="text"/>
<input type="radio"/> Fire	Payroll: <input type="text"/>
<input type="radio"/> Police	
<input type="radio"/> Public Works	
<input type="radio"/> Social Services	

**Figure 13.3:** Groupings with section headings.

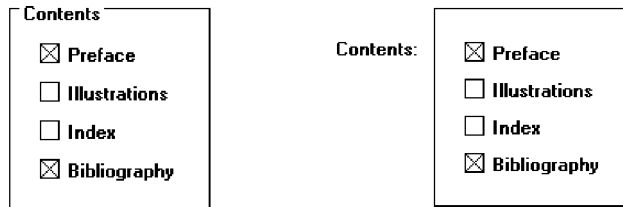
## ***Borders***

Groupings can be further enhanced through the use of borders. Inscribe line borders around elements of a single control such as a radio button or check box and/or groups of related controls or fields. Individual control borders should be visually differentiable from borders delineating groupings of fields or controls. Provide a border consisting of a thin line around single controls and a slightly thicker line around groups of fields or controls.

### **Control Borders**

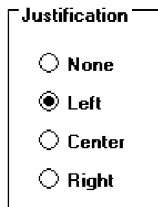
---

- Incorporate a thin single-line border around the elements of a selection control.
- For spacing,
  - Vertically, leave one line space above and below the control elements.
  - Horizontally:
    - Leave at least two character positions between the border and the left side of the control elements.
    - Leave at least two character positions between the border and the right side of the longest control element.
  - Locate the control caption in the top border, indented one character position from the left border.
    - Alternately, locate the caption at the upper left of the box.



**Figure 13.4**

- If the control caption exceeds the length of the choice descriptions, extend the border two character positions to the right of the caption.



**Figure 13.5**

Thin line borders may be used to surround some boxed-in controls, particularly radio buttons and check boxes. Control captions should be located upper left within the border itself, or to the left of the box. The spacing guidelines are to prevent cramming the text within the border. Some examples of control borders are illustrated in Figures 13.6 and 13.7.

### Section Borders

- Incorporate a thicker single-line border around groups of related entry or selection controls.
- For spacing,
  - Vertically, leave one line space between the top and bottom row of the entry or selection control elements.
  - Horizontally, leave at least four character positions to the left and right of the longest caption and/or entry field.
- Locate the section heading in the top border, indented two character positions from the left border.

Figure 13.6 displays four examples of vertically arrayed controls, organized into two pairs. Each pair consists of a control without a border and a control with a border.

**Top Pair (Without and With Borders):**

- Without Border:** A label "Department:" followed by a vertical list of radio buttons: Administration, Finance, Fire, Police, Public Works, and Social Services.
- With Border:** The same label and list of radio buttons enclosed within a rectangular border.

**Bottom Pair (Without and With Borders):**

- Without Border:** A label "Department:" followed by a vertical list of radio buttons: Administration, Finance, Fire, Police, Public Works, and Social Services.
- With Border:** The same label and list of radio buttons enclosed within a rectangular border.

**Figure 13.6:** Examples of vertically arrayed controls without and with borders.

Figure 13.7 displays four examples of horizontally arrayed controls, organized into two pairs. Each pair consists of a control without a border and a control with a border.

**Top Pair (Without and With Borders):**

- Without Border:** A label "Municipality:" followed by four radio buttons: City, Township, County, and State.
- With Border:** The same label and four radio buttons enclosed within a rectangular border.

**Bottom Pair (Without and With Borders):**

- Without Border:** A label "Department:" followed by two rows of radio buttons. The first row contains Administration, Fire, and Public Works. The second row contains Finance, Police, and Social Services.
- With Border:** The same label and two rows of radio buttons enclosed within a rectangular border.

**Figure 13.7:** Examples of horizontally arrayed controls without and with borders.

Line borders may be used to surround groupings of related controls. Section headings should be located at the upper left within the border itself. Display section headings in capital letters to differentiate them easily from individual control captions. The spacing guidelines are to prevent cramping the text within the border. Examples of section borders are illustrated in Figure 13.8.

If both control borders and section borders are included on the same screen, make the section border slightly thicker, as illustrated in Figure 13.9.

Be conservative in the use of borders because too many can lead to screen clutter. Do not place individual field borders around the individual controls previously listed, and illustrated in Figure 13.10. The nature of their design provides them with a border. Also, because of the potential for clutter, do not place a border around groups of push-buttons.

The figure shows a form layout with four distinct sections, each enclosed in a rectangular border. The sections are arranged in a 2x2 grid. The top-left section is titled 'JURISDICTION' and contains two groups of radio buttons. The top-right section is titled 'LOCATION' and contains three text input fields. The bottom-left section is unlabeled but contains two groups of radio buttons. The bottom-right section is titled 'PERSONNEL' and contains three text input fields.

JURISDICTION	LOCATION
Municipality: <input type="radio"/> City <input type="radio"/> Township <input type="radio"/> County <input type="radio"/> State	Building: <input type="text"/> Floor: <input type="text"/> Telephone: <input type="text"/> <input type="text"/> <input type="text"/>
Department: <input type="radio"/> Administration <input type="radio"/> Finance <input type="radio"/> Fire <input type="radio"/> Police <input type="radio"/> Public Works <input type="radio"/> Social Services	PERSONNEL Manager: <input type="text"/> Employees: <input type="text"/> Payroll: <input type="text"/>

**Figure 13.8:** Grouping of sections using borders.

**DOCUMENT**

Justification: ☐ None  
☐ Left  
☐ Center  
☐ Right

Contents: ☐ Preface  
☐ Illustrations  
☐ Index  
☐ Bibliography


Title:

Number of Chapters:

Number of Pages:



**Figure 13.9:** Differentiable control and section borders.

Text Box:

Combo Box:  

Attached Combo:

List Box:

Spin Box:   

**Figure 13.10:** Kinds of borders to avoid using.

## Dependent Controls

- Position a conditional control, or controls
  - To the right of the control to which it relates.

Number of Children:  > Names:

**Figure 13.11**

- Alternately, position it below the control to which it relates.

Number of Children:   
 > Names:

**Figure 13.12**

- Either
  - Display these conditional controls in a subdued or grayed out manner.
    - When a control is relevant, return it to a normal intensity.
  - Do not display a conditional control until the information to which it relates is set.
- Inscribe a filled-in arrow between the selected control and its dependent controls to visually relate them to each other.

In some circumstances, a control may be active only under certain conditions. Only when a particular response is made is this additional information needed. For example, a question such as “Do you have any children?” might necessitate knowing their names. If this question is answered affirmatively, a field requesting their names can be displayed at that point on the screen. If a “No” response is given, the tab will cause the cursor to move to the next field, and the children’s names field will not appear.

Locate a dependent or conditional control to the right of or below the field or choice that necessitates it. Exact positioning will depend upon the flow of eye movement through the screen and screen space constraints. The displayed arrow serves to tie the dependent control to the triggering control. The control field may either be shown in a grayed out or subdued manner, or not displayed at all until it is needed.

Displaying a control as grayed out or subdued allows the user to be aware of its existence but reduces the visual competition between it and other needed information on the screen. Not displaying dependent fields until they are triggered reduces screen clutter. Hiding their existence, however, does not give the screen user a full picture of all the possibly needed data and the relationships that may exist. Hiding them, then, may result in a slight learning price, depending upon the complexity of the needed data. The recommendation is to display them grayed out.

One caution: Place dependent controls within a window only when their frequency of use is moderate or high. Infrequently used dependent information is best located in

a dialog box. The extra step needed to occasionally obtain the dialog box is more than compensated for by the additional screen space made available and the reduction in screen clutter.

### ***Aligning Screen Elements***

---

- Minimize alignment points on a window.
    - Vertically.
    - Horizontally.
- 

Fewer screen alignment points reduce a screen's complexity and make it more visually appealing. Aligning elements will also make eye and pointer movement through the screen much more obvious and reduce the distance both must travel. Screen organization will also be more consistent and predictable. Alignment is achieved by creating vertical columns of screen fields and controls and also horizontally aligning the tops of screen elements.

**MAXIM** Designers work hard so users don't have to.

Fields or controls vertically columnized may be oriented in two directions, vertically or horizontally. Vertical orientation, a top-to-bottom flow through controls and control components, is the recommended structure.

### **Vertical Orientation and Vertical Alignment**

---

- Radio buttons/check boxes:
  - Left-align choice descriptions, selection indicators, and borders.
  - Captions:
    - Those inscribed within borders must be left-aligned.

**Contents**

- ☐ Preface
- ☒ Illustrations
- ☒ Index
- ☐ Bibliography

**Justification**

- ☐ None
- ☒ Left
- ☐ Center
- ☐ Right

**Figure 13.13**

- Those located at the left may be left- or right-aligned.



- Text boxes:
  - Left-align the boxes. If the screen will be used for inquiry or display purposes, numeric fields should be right-aligned.
  - Captions may be left- or right-aligned.

Title:

Number of Chapters:

Number of Pages:

**Figure 13.14**

- List boxes:
  - Left-align fixed list boxes.
  - Captions:
    - Those located above the boxes must be left-aligned.

Location:

Bristol	↑
Buckhead	↓
Canton	↑
Edison Park	↓

Model:

Chevrolet	↑
Edsel	↓
Ford	↑
GMC	↓
Honda	↑

**Figure 13.15**

- Those located at the left may be left- or right-aligned.
- Drop-down/pop-up boxes, spin boxes, combo boxes:
  - Left-align control boxes.
  - Field captions may be left- or right-aligned.

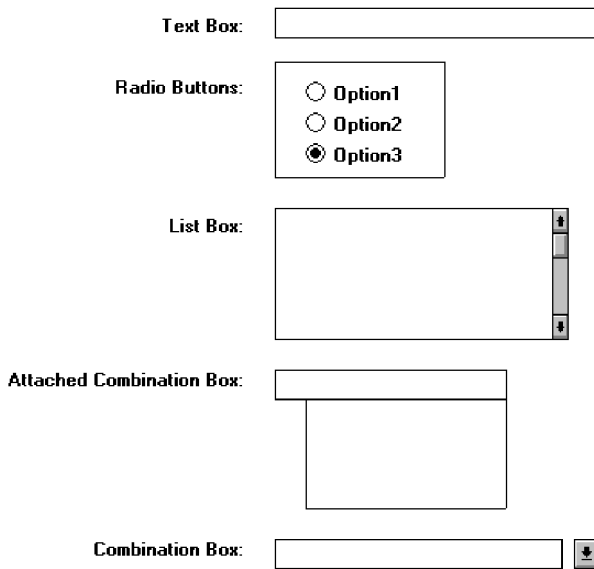
Spin Box:

Attached Combination Box:

Drop-down/pop-up Box:

**Figure 13.16**










- Mixed controls
  - Left-align vertically arrayed
    - Text boxes.
    - Radio buttons.
    - Check boxes.
    - Drop-down/pop-up list boxes.
    - Spin boxes.
    - Combination control boxes.
    - List boxes.
  - Captions may be left- or right-aligned.



**Figure 13.17**

Elements and information should be organized vertically (top to bottom) as well. Two, and sometimes three, columns of controls and fields may occasionally be created. When multiple columns are presented and no section borders are used, column separation and downward flow may be emphasized through line borders, as illustrated in Figure 13.18.

In some cases, window space constraints may dictate a horizontal orientation of controls, most noticeably radio buttons and check boxes. Again the pattern created must be consistent, predictable, and distinct.

APPLICANT	VEHICLE
Name: <input type="text"/>	Make: <input type="text"/> 
Occupation: <input type="text"/>	Model: <input type="text"/> 
Birth Date: <input type="text"/> <input type="text"/> <input type="text"/>	ID Number: <input type="text"/>
	Horsepower: <input type="text"/> 
LICENSE	Annual Miles: <input type="text"/> 
Number: <input type="text"/>	Use: <input type="text"/> 
State: <input type="text"/> 	Miles to Work: <input type="text"/> 
Years: <input type="text"/> 	Symbol: <input type="text"/>
Restriction: <input type="text"/> 	

**Figure 13.18:** Multicolumn controls with a separation border.

### Horizontal Orientation and Vertical Alignment

- Radio buttons/check boxes selection controls:
  - Align leftmost radio buttons and/or check boxes.
  - Field captions may be left- or right-aligned.

Justification: ☐ None ☐ Left ☐ Even ☐ Center

Contents: ☐ Preface ☐ Illustrations ☐ Index ☐ Bibliography

**Figure 13.19**

- Text boxes:
  - Left-align text boxes into columns.
  - Captions may be left- or right-aligned.

Author: <input type="text"/>	Organization: <input type="text"/>
Location: <input type="text"/>	Building: <input type="text"/>

**Figure 13.20**

- Numeric data should be right-aligned.

Length: <input type="text"/>	Width: <input type="text"/>
Thickness: <input type="text"/>	Weight: <input type="text"/>

**Figure 13.21**

- Mixed text boxes and selection controls:
  - Align leftmost radio buttons and/or check boxes.
  - Align the leftmost text box under the leftmost choice description button or box.
  - Captions may be left- or right-aligned.

Justification: ☐ None ☒ Left ☐ Even ☐ Center

Contents: ☒ Preface ☐ Illustrations ☒ Index ☐ Bibliography

Author:  Organization:

Location:  Building:

**Figure 13.22**

---

For horizontally oriented controls, indistinctiveness can be caused if the item descriptions are positioned as close to a following button or box as they are to the button or box they relate to, as illustrated in Figure 13.23. While the objective is to create as few vertical alignment points as possible, this is usually not practical. For check boxes and radio buttons, most often the result will be inconsistently spaced item descriptions, as illustrated in Figure 13.24. Vertical alignment of items in several adjacent controls can also create a false vertical orientation perception, which is also illustrated in Figure 13.24. Final positioning will be a compromise between alignment and providing clear item distinctiveness, as illustrated in Figure 13.25. With a vertical orientation of radio buttons and check boxes, all these problems are avoided. With a horizontal orientation, borders aid discrimination and separation, as illustrated in Figure 13.26. Although the examples in the previous guidelines illustrate text boxes structured left to right, every attempt should be made to maintain a top-to-bottom orientation of single entry and selection fields. The fields in the example will be more effectively structured as illustrated in Figure 13.27.

Municipality:	<input type="radio"/>	City	<input type="radio"/>	Township	<input type="radio"/>	County	<input type="radio"/>	State
Department:	<input type="radio"/>	Administration	<input type="radio"/>	Finance	<input type="radio"/>	Public Services	<input type="radio"/>	Social Services
Job Title:	<input type="radio"/>	Director	<input type="radio"/>	Manager	<input type="radio"/>	Professional	<input type="radio"/>	Staff

**Figure 13.23:** Horizontally arrayed radio buttons with poor item differentiation.

Municipality:	<input type="radio"/> City	<input type="radio"/> Township	<input type="radio"/> County	<input type="radio"/> State
Department:	<input type="radio"/> Administration	<input type="radio"/> Finance	<input type="radio"/> Public Services	<input type="radio"/> Social Services
Job Title:	<input type="radio"/> Director	<input type="radio"/> Manager	<input type="radio"/> Professional	<input type="radio"/> Staff

**Figure 13.24:** Horizontally arrayed control items with inconsistent spacing and a false vertical orientation.

Municipality:	<input type="radio"/> City	<input type="radio"/> Township	<input type="radio"/> County	<input type="radio"/> State
Department:	<input type="radio"/> Administration	<input type="radio"/> Finance	<input type="radio"/> Public Services	<input type="radio"/> Social Services
Job Title:	<input type="radio"/> Director	<input type="radio"/> Manager	<input type="radio"/> Professional	<input type="radio"/> Staff

**Figure 13.25:** Horizontally arrayed control items possessing alignment and distinctiveness.

Municipality:	<input type="radio"/> City	<input type="radio"/> Township	<input type="radio"/> County	<input type="radio"/> State
Department:	<input type="radio"/> Administration	<input checked="" type="radio"/> Finance	<input type="radio"/> Public Services	<input type="radio"/> Social Services
Job Title:	<input type="radio"/> Director	<input type="radio"/> Manager	<input type="radio"/> Professional	<input type="radio"/> Staff

**Figure 13.26:** Horizontally arrayed control items with borders to improve readability.

Municipality: ☐ City ☐ Township ☐ County ☐ State

Department: ☐ Administration ☐ Finance ☐ Public Services ☐ Social Services

Job Title: ☐ Director ☐ Manager ☐ Professional ☐ Staff

Name:

Building:

Floor:

Mail Stop:

**Figure 13.27:** Vertical orientation of text boxes.

### Horizontal Alignment

- Text boxes:
  - Align by their tops horizontally adjacent text boxes.
- Radio buttons/check boxes:
  - Align by their tops horizontally adjacent radio button and/or check box controls.
- Fixed list boxes:
  - Align by their tops horizontally adjacent fixed list boxes.
- Drop-down/pop-up list box, spin box, combo boxes:
  - Align by their tops horizontally adjacent entry/selection fields.

Text Box:

Radio Buttons: ☐ Option1 ☐ Option2 ☒ Option3

List Box:

Combo Box:

Text Box:

Check Boxes: ☐ Check1 ☐ Check2 ☐ Check3 ☐ Check4

Text Box:

Text Box:

Combo Box:

Combo Box:

**Figure 13.28**

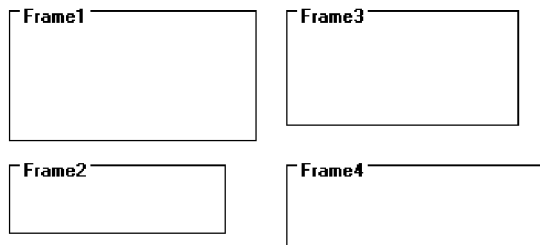
- Mixed text boxes and selection controls:
    - Align by their tops:
      - Text boxes.
      - Radio buttons.
        - If a control border exists, align by top border.
      - Check boxes.
        - If a control border exists, align by top border.
      - Drop-down/pop-up list boxes.
      - Spin boxes.
      - Combination boxes.
      - List boxes.
- 

Arrangement of controls horizontally always consists of aligning them by their tops. Because controls may be of different heights, screen efficiency occasionally dictates that a control must be positioned in an area where it does not align horizontally with another control. When this occurs, attempt to align it horizontally with the bottom of an adjacent control, as illustrated by the list box and adjacent combo box in the previous example. Do not cramp a control, however, to achieve bottom alignment.

### Section Alignment

---

- Align by their left side vertically arrayed groupings containing section borders.
- Align by their top horizontally arrayed groupings containing section borders.



**Figure 13.29**

---

Groupings with borders should also be aligned vertically by their left border and horizontally by their top border. Controls within a grouping will, of course, be aligned following the alignment principles previously discussed.

### Balancing Elements

---

- General:
  - Create balance by
    - Equally distributing controls, spatially, within a window.
    - Aligning borders whenever possible.

- Individual control borders:
  - If more than one control with a border is incorporated within a column on a screen
    - Align the controls following the guidelines for multiple-control alignment.
    - Align the left and right borders of all groups.
    - Establish the left and right border positions according to the spacing required for the widest element within the groups.

**Contents**

- ☐ Preface
- ☐ Illustrations
- ☐ Index
- ☒ Bibliography

**Justification**

- ☐ None
- ☐ Left
- ☐ Center
- ☒ Right

**Figure 13.30**

- With multiple groupings and multiple columns, create a balanced screen by
  - Maintaining equal column widths as much as practical.
  - Maintaining equal column heights as much as practical.

<div style="border: 1px solid black; padding: 5px; margin-bottom: 10px;"> <b>Frame1</b> </div> <div style="border: 1px solid black; padding: 5px;"> <b>Frame2</b> </div>	<div style="border: 1px solid black; padding: 5px; margin-bottom: 10px;"> <b>Frame3</b> </div> <div style="border: 1px solid black; padding: 5px; margin-bottom: 10px;"> <b>Frame4</b> </div> <div style="border: 1px solid black; padding: 5px;"> <b>Frame5</b> </div>
--	---

**Figure 13.31**

- Section borders:
  - If more than one section with borders is incorporated within a column on a screen,
    - Align the left and right borders of all groups.
    - Establish the left and right border positions according to the spacing required by the widest element within the groups.



**DOCUMENT**

Justification: ☒ None  
☐ Left  
☐ Center  
☐ Right

Contents: ☒ Preface  
☒ Illustrations  
☒ Index  
☒ Bibliography

**AUTHOR**

Name:

Telephone:

**Figure 13.32**

- With multiple groupings and multiple columns, create a balanced screen by
- Maintaining equal column widths as much as practical.
  - Maintaining equal column heights as much as practical.

**DOCUMENT**

**PUBLISHER**

**DEPARTMENT**

**AUTHOR**

**LOCATION**

**Figure 13.33**

Screen balance should be attained as much as possible. Do not sacrifice screen functionality to achieve balance, however. Never rearrange controls to simply make the screen look nice. A meaningful order of elements is most important. The look will be the best that can be achieved within the limits imposed by functionality.

One additional point about alignment: While these guidelines suggest aligning section, radio button, and check box borders on the right side as well as the left, a glance back at Figure 13.9 will reveal an instance where this was not done. The Justification and Contents borders were not right-aligned because the text boxes within that grouping created a ragged right edge. Aligning just these two controls would have served no purpose. So, all alignment and balancing must occur within the context of the *whole* screen.

## Control Navigation

---

- Tab/arrow keys:
    - Use the tab key to move between operable window controls, in the logical order of the controls.
      - Do not tab to field captions/labels.
    - Radio buttons:
      - Use arrow keys to move through radio buttons within a single control.
    - Check boxes:
      - Use the Tab key to move between, when they are independent controls.
      - Within a border or group box, use arrow keys to move between the check boxes since they appear as a logical group.
    - List boxes:
      - Use arrow keys to navigate within list box choices.
  - Command buttons:
    - For exiting or expanding/feature dialog command buttons,
      - Tab to them at the end of the screen control tabbing sequence.
    - For a command button with a contingent relationship to a control in the window body,
      - Tab to it at the logical point in the tabbing sequence within the window.
  - Keyboard equivalents:
    - Use keyboard equivalents (mnemonics) for direct access to each control, whenever possible.
      - Mnemonic designations must be unique within a window.
- 

Use the tab key to move between operable window controls, in the logical order that the controls are organized. Do not tab to control captions but the control itself. For a grouping of radio buttons, use the arrow keys to move through an array of buttons. For check boxes, use the tab key to move between them when they are arrayed as independent controls. When check boxes are located within a border or group box, use the arrow keys to move between the boxes since they appear as a logical group. Always use arrow keys to navigate within a listing of choices.

Tab to exiting or expanding/feature dialog buttons at the end of the screen control tabbing sequence. If a button has a contingent relationship to a control in the window body, tab to it at the logical point in the tabbing sequence within the window.

Use keyboard equivalents (mnemonics) for direct access to each window control, whenever possible. Mnemonic designations must be unique within a window. The command buttons OK and Cancel are not typically assigned mnemonics, the Enter and Esc keys being used instead.

## Window Guidelines

---

- Organization:
    - Organize windows to support user tasks.
    - Present related information in a single window whenever possible.
    - Support the most common tasks in the most efficient sequence of steps.
    - Use
      - Primary windows to
        - Begin an interaction and provide top-level context for dependent windows.
        - Perform a major interaction.
      - Secondary windows to
        - Extend the interaction.
        - Obtain or display supplemental information related to the primary window.
      - Dialog boxes for
        - Infrequently used or needed information.
        - “Nice-to-know” information.
  - Number:
    - Minimize the number of windows needed to accomplish an objective.
  - Size:
    - Provide large enough windows to
      - Present all relevant and expected information for the task.
      - Prevent hiding important information.
      - Prevent crowding or visual confusion.
      - Minimize the need for scrolling.
      - Occupy less than the full size of the entire screen.
    - If a window is too large, determine
      - Is all the information needed?
      - Is all the information related?
- 

These guidelines, and many others, are discussed in Step 5. In summary, always organize windows to support user tasks. Support the most common tasks in the most efficient sequence of steps. Minimize the number of windows needed to accomplish an objective. In general, present all related information in a single window whenever it is possible to do so. A window should be large enough to accommodate the amount of data a person would typically expect to see. The needed information can only be determined through thorough task analysis.

The necessity for window scrolling should also be avoided whenever possible. If all the relevant controls or data cannot be placed within the confines of a single window, place that which is less frequently needed on another window. Do not make the default size of a window the full screen. The option to maximize a window always exists.

Finally, use primary windows, secondary windows, and dialog boxes consistently and in the manner they are intended to be used.

## Web Page Guidelines

The following specific Web page layout and navigation principles, previously discussed in Steps 3 and 4, must also be considered in page design. Important principles are summarized here.

### *Page Layout*

---

- General:
  - Provide a layout that is
    - Efficient.
    - Logical.
    - Consistent.
    - Self-explanatory.
    - Scannable.
  - Strike a proper balance between
    - Textual information.
    - Graphics.
    - Links.
- Layout Grid:
  - Create and use a layout grid.
- Elements:
  - Restrict the number of elements per page.
- Size:
  - Minimize page length.
    - Generally, use shorter pages for
      - Homepages.
      - Navigation pages.
      - Pages needing to be quickly browsed or read online and information quickly found.
      - Situations where a page will be loading over slow modems and all pages are not needed.
    - Generally, use longer pages for
      - Content pages where uninterrupted reading is desirable.
      - Content pages in which an entire concept must be understood without interruption.
      - To match the structure of a paper counterpart.
      - To make pages more convenient to download.
      - To make pages more convenient to print.
- To simplify page maintenance,
  - Generally, restrict to two or three screens of information.
  - Place critical or important information at the very top so it is always viewable when the page is opened.
  - Locate within the top 4 inches of page.
  - Anticipate page breaks.
- Avoid horizontal scrolling.
- Allow complete page printing.

- **Organization:**
    - Place critical or important information at the very top so it is always viewable when the page is opened.
      - Locate it within the top 4 inches of page.
    - Limit large images above the fold.
    - Position remaining elements according to importance.
    - Reduce graphic complexity and textual density toward the page bottom.
  - **Formatting:**
    - Provide sufficient but moderate amount of white space.
      - A minimum of 30 percent.
    - Keep the length of textual lines short.
      - If fast reading is required, use line lengths of about 75 to 100 characters.
      - If user preference is important, a length of about 50 to 60 characters is acceptable.
      - Very narrow columns should be avoided.
    - Keep text and related graphics close to each other.
    - Provide adequate horizontal spacing.
    - Use horizontal rules sparingly.
  - **Multiple Audiences:**
    - If distinctively different audiences exist for presented information, provide information formatted for each audience.
  - **Platforms:**
    - Design for different platforms and screens.
    - Specify an image-safe area.
  - **Frames:**
    - Use frames with caution.
      - Consider them for global elements.
  - **Fixed versus fluid layout.**
    - Use a fluid layout.
  - **Change:**
    - Change the site and page organization and structure only when significant benefits exist.
- 

Well-organized, thoughtfully-laid-out, and consistent Web pages will permit people to easily learn a site's structure, allow users to effectively and quickly scan its pages, and foster interest in its content. Layout does influence satisfaction with the page, found Chaparro et al. (2005). A well-designed page will possess the following qualities.

### **General**

**Efficient.** A page will be more powerful if more is said with less. Every element of the design must support the goal of the message being presented. Eliminate all superfluous elements. Avoid clutter that prevents people from finding what they want. Exercise extreme care in using decorative elements.

**Logical and consistent.** Locate important page elements in consistent locations on all pages. A logical and consistent layout of the site and its pages will enable people to quickly find what they are looking for. As said earlier in this book, people

develop expectations for how to find different types of information and how to perform particular tasks. A layout that matches a person's expectations speeds learning and enables prediction of where to find things and how to do things. Illogical and inconsistent structures lead only to user frustration.

**Self-explanatory.** Each page should be self-explanatory, giving a clear indication of what Web site it belongs to and where it fits within the structure of the Web site's information space. Present the proper information, and arrange it, so users always know where they are. Remember, most pages can be accessed from outside the site.

**Scannable.** Allow people to easily scan a page and select relevant and useful information. To foster scannability, include headings that accurately reflect the content of text. Also create short paragraphs and use simple bulleted lists (complex bullets, such as diamonds and fingers, may create unnecessary visual noise). Avoid the use of too many links and use plenty of white space.

**Proper balance.** Always strike a proper balance between textual information, graphics and navigation links. A page's elements must balance, interact with, and support each other. As mentioned earlier, studies show people prefer textual content, so do not clutter up the page with too many graphics, particularly at the top. A large top graphic may push important textual content and navigation links off the bottom of the page where they cannot be seen. Always make sure text is visible first so people can start reading right away. Conversely, densely packed text will be difficult to read. Long and detailed textual information can usually be relegated to secondary pages.

**MYTH** This way of doing it must be right because (fill in the blank) does it that way.

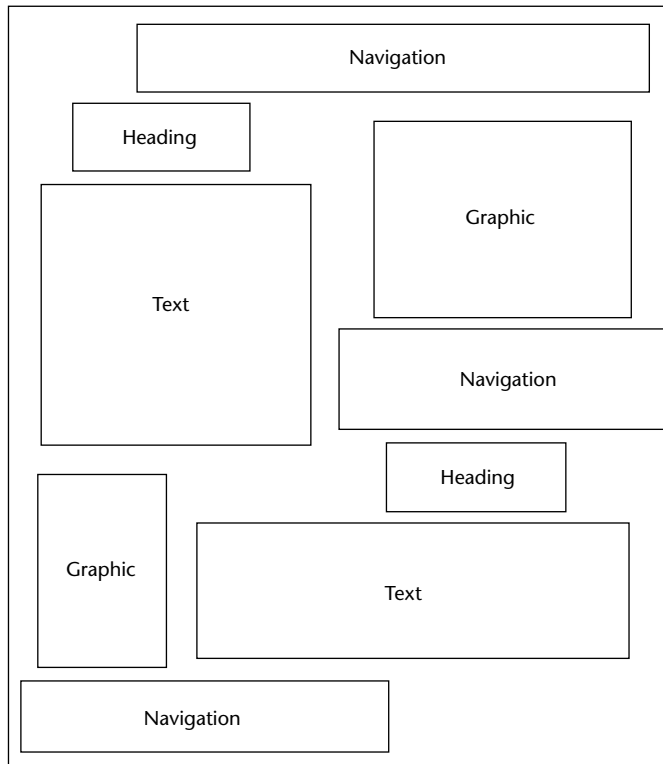
### **Layout Grid**

To provide both structure and consistency to a Web site, establish a standard layout grid, or grids, for each of the site's pages. Multiple, but similar, grids may be necessary for site pages with different purposes or varying complexity. These design grids will be composed of rectangular sections into which the page's navigation components (headings, text, and graphics) will be placed. To develop the grids,

- Gather representative samples of the contents of site pages. Include all types of pages: navigation pages, content pages, simple pages, and complex pages.
- Experiment with various arrangements for all kinds of pages, painting or sketching patterns of organization on sample pages.
- Follow all layout guidelines (alignment, balance, and so on) and evolving page organizational standards in the sketching process.
- Establish a design grid (or grids) for the identified page types by maintaining as much consistency between page types as possible.
- Plug in content (navigational components, text, and graphics) for each page.

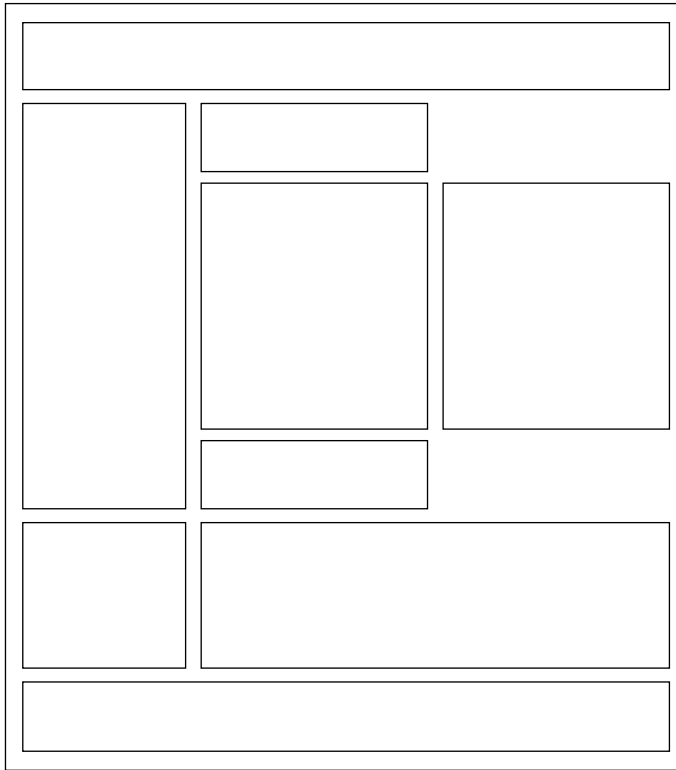
Ideally, one standard design grid may satisfy the needs of many Web sites. Larger sites with more content may require multiple grids. Keep in mind that fewer grids are better, and if you use multiple grids, maintain as much consistency as possible between the grids' layouts.

The use of the grids on all pages will provide a unifying character within and between pages, a standard look and feel. Regular and repeating organizational patterns help readers to understand site's organization and aid them in maintaining a sense of place. Grids allow pages to be laid out without the designer having to stop and rethink the basic design approach for every new page. Pressing needs of the moment will also have much less of an influence on design. Examples of page layout grids, both poor and good, are illustrated in Figure 13.34. The poor grid is haphazard in design, and there is little connection of the elements. The good grid reflects much greater alignment and better visual connection of elements. It is structured and more consistent. As described in Step 3, the good grid has a visual complexity measure that is 47 percent lower than the poor grid.



Poor grid (continues)

**Figure 13.34:** Page layout grids.



Good grid

**Figure 13.34:** Continued.

### Elements

To keep pages simple, restrict the number of distinct functional elements on a page. Elements include such things as the title, the navigation bar, the textual link listing to major site areas, textual content, graphics, and the page footer. The more elements on a page, the greater the competition between the elements for a person's attention. Too many elements will eventually overwhelm the user's information-processing system.

### Size

**Page length.** The many issues associated with page length are thoroughly described in Step 3 (such as vertical scrolling, user memory requirements, user tasks, and monitor sizes), so optimum page length must balance all these factors. In general, the best approach seems to be to minimize a page's length, restricting it to two or three screens of information.

**Anticipate page breaks.** In laying out pages exceeding one screen in length, always consider where the browser window will cut off a page when it loads. If the break occurs at a logical point on the page, and the page appears to fit totally on the screen, people may assume that it ends at that point and look no further down. Where potential screen cut-offs can occur, present a design element (textual or



graphical) that, by its appearance, is obviously not completed or finished. This signals to the user that there is more below and indicates a need to scroll down.

**Horizontal scrolling.** While people can accept some vertical scrolling, horizontal scrolling is cumbersome and disliked. Lay out a page so that horizontal scrolling is never necessary. Also ensure that a page can be fully printed on a standard  $8\frac{1}{2} \times 11$  inch portrait orientation.

## **Organization**

**Important elements at the top.** Critical or important information should be placed where it will be immediately visible when the page is displayed. As described earlier, in Web page design this is referred to as “above the fold,” a term borrowed from newspaper page layout. Above the fold is about the top 4 inches of a page and is the area of the page most users are assured of seeing. Information “below the fold” will usually require scrolling to access and may not always be seen. Positioning large images above the fold may cause readers to assume there is no information further down the page.

Generally, the top of the page should be more dense and packed with more useful and interesting information than the area lower down. Include in this area the page title, links to other important content and pages, and summary information so the user can determine whether continued reading is desirable. Less important information should be positioned lower down on the page.

**Positioning.** The effect of presented information drops very quickly as one moves lower down the page. Visually prioritize the page information space by establishing an information hierarchy, using contrast, size, location pattern, and other concepts discussed in Step 3.

**Reduced complexity.** Reduce graphic complexity and textual density as one moves lower down a page or to interior site pages. Produce diminishing or thinning vertical gradients of complexity, providing quieter and less-distracting design elements. This will enable people to focus on the content, which must have some interest because they have navigated that far. To be distracted by irrelevant information or useless animation at that point can be very irritating.

## **Formatting**

**White space.** To enhance readability and organization, and make the page more inviting, allow a sufficient amount of white space on each page. According to many experts, white space should consume at least 30 percent of a page. (Remember, white space is a design element, not something to always be consumed.) Of the remaining 70 percent of space, restrict text to no more than another 30 percent. The remaining 40 percent may be used for graphics. If the entire 40 percent is not used for graphics, more text may be included. White space should always be used to separate paragraphs. A satisfactory balance between text and white space must be achieved to avoid excessive scrolling.

**Textual line length.** If fast reading is required, use line lengths of about 75 to 100 characters. If user preference is important, a length of about 50 to 60 characters is acceptable. Avoid very narrow columns.

**Text and related graphics.** Keep text and any related graphics close to each other. The viewer will assume a connection between elements located in close proximity. Also, keep chunks of related text in closer proximity than unrelated text, to ensure a connection.

**Horizontal spacing.** Provide sufficient horizontal spacing so that groupings of information are obvious. A chunk of text, its heading, and any related graphic should be readily discernible as a grouping.

**Horizontal rules.** Use horizontal rules sparingly on pages. These rules can break up page flow and signal a page's end when it is not intended. Use horizontal rules only if the objective is to break the flow. Uses might be, for example, to separate standard header and footer information from basic page content.

### **Multiple Audiences**

If distinctively different audiences exist for presented information, provide pages formatted for each audience. Information can be presented at different levels of detail, or at different reading levels, for example. Allow users from each audience to have access to all versions, however. Initially, do not segment the Web site on the homepage itself. Provide links to multiple versions from the homepage or on a later page.

### **Platforms**

Anticipate the typical screen and resolution and specify an image-safe area of the screen that will always be visible to most viewers. The most important information can be located here. This is not an ideal solution because users with very small screens will still have to scroll and those with large screens will be viewing the page through a window that is smaller than the actual screen size.

### **Frames**

Use frames with caution. As discussed in Step 4, frames can be confusing for many users. They may cause problems with scrolling, bookmarking, and printing. If a person arrives at a framed page from a search facility, the page is seen without the accompanying frames. The site name and a link to the homepage must always be provided on the content page so the user does not reach a dead-end. They also reduce the content window size.

Frames, however, are useful for presenting global page elements such as navigation links, or information, that must remain visible on the screen while additional information is displayed. Frames containing links are sometimes called simultaneous menus because a selection in one frame causes another to change its contents. Side-by-side frames appear to work best, the links appearing on the left side and the resulting information on the right side.

### **Fixed versus Fluid Page Layouts**

Layouts of a Web page may be either *fluid* or *fixed* (sometimes referred to as *frozen*). The fluid layout is the most traditional, the contents of a Web page filling the entire window, expanding and contracting with the size of the window. The fluid layout is simplest to implement. Common fixed layouts are *centered* or *left-justified*. The centered layout is

centered within the boundaries of a display, requiring users to only focus on a narrower layout in the middle of the display no matter what screen resolution is being used. The left-justified layout is positioned against the left border of the display and is usually slightly wider than the centered layout. Bernard and Larsen (2001c) compared these three layout styles and found no differences in search time, accuracy in finding information, or in search efficiency. However, study participants felt that the fluid style was best suited to reading and finding information. It was also the preferred layout style.

### **Change**

In an environment as volatile and changing as the Web, the urge to try the newest and latest concepts in design and layout always exists. Use restraint. A Web site's fashion should be evident in its content, not its design. Fashion changes all the time, and constant changes in design will continually antagonize users who have invested time and effort in understanding a site's structure and organization. Change the organization and structure only when significant benefits for the user and the Web site owner are perceived to exist.

### ***Navigation Elements***

---

- Differentiate and group navigation elements.
    - Provide a global navigation bar at the top of each page.
    - Provide a local category or topical links navigation bar on the left side of a page.
      - For long lists, consider placing within a frame.
    - Optionally, provide a secondary navigation column on the right side of the page.
    - Provide explicit or embedded textual links within the contents area.
      - Consider duplicating embedded links in the left side of the navigation bar.
    - Place minor illustrative, parenthetical, or footnote links at the end of the page.
    - For long pages provide
      - "List of Content" Links.
      - Important global or local links in a navigation bar repeated at the page bottom.
  - Create a common and consistent theme.
  - Never create pages without navigational options.
- 

A Web site contains at least three levels of navigation elements: global or site-wide; local and specific; and minor or footnote. Clearly differentiate these navigation elements from one another and locate them consistently from page to page. Commonly selected items should be positioned close to the edge of a screen to expedite their selection.

**Global.** Global or site-wide navigation elements provide access to the site's total scope or categories of available information. An evolving standard in design is to locate the global navigation elements horizontally at a page's top.

**Category or topical.** Local, specific and contextual navigation elements within the category or topical area being presented are typically displayed in a column array down the left page side. For long lists consider placing the links within a frame navigation panel.

**Explicit or embedded links.** Phrases or embedded links will be provided within the contents area of a Web page. An embedded link is one found in the middle of prose or continuous text. Embedded links are frequently used to lead to supporting information or provide definition of terms. They are designated by an underline and a unique color. Because users preferred redundant links, consider duplicating embedded links in the left side navigation bar.

**Minor.** Minor illustrative, parenthetical, or footnote links can be arrayed horizontally at the page bottom.

**List of content.** For long pages with sections that are not visible without scrolling include a set of links to each page section at the top of the page. These “anchor” or “within page” links provide a reminder of the page’s contents, a page outline that can be easily reviewed, and a quick way to navigate to desired sections. These links also assist people in getting to a specific section if they arrive from a different page.

**Important links.** For long scrolling pages, repeat important global or local links at the page bottom. When finishing a page, the user, then, will not have to scroll upward to locate important navigation links.

**Common theme.** A common and consistent Web site navigation theme will enable people to more easily understand and learn its structure. Incorporate different styles for these different navigation elements to aid people in understanding the differences in their meaning and function.

**Always present options.** All pages must have navigation options. Never create pages without navigation options.

## ***Homepage***

---

- Limit to one screen.
  - Clearly identify the Web site’s content and purpose.
  - Elements to include:
    - Masthead, name or logo of Web site owner and tagline.
    - Web site name.
    - Brief description of Web site.
    - Summary of the key informational content.
    - Navigation links to most (if not all) of the site or major sections.
    - Site overview or map.
    - Summary of the latest news or promotions.
    - Search facility.
- 

A site’s homepage is a concrete and visual anchor point; a safe haven to return to when one is confused or decides to do something else. The homepage also gives people a first impression of a site, and first impressions can create a positive or negative feeling, or a feeling somewhere in between. A negative first impression can lead to rejection, whereas a positive first impression can create an urge for further exploration. In many ways, the homepage is a site’s most important page.

**One screen.** Keep the homepage to one screen. All important elements should be viewable without scrolling so they will not be missed. Elements of less importance and those not vital to site use (corporate or organizational identification information or e-mail links, for example) may be included below the visible area, however.

**Masthead, logo, and tagline.** Position a masthead or banner at the top of the homepage to identify the organization that hosts the Web site. A masthead usually contains the name of the organization (and site, if different) and a logo. Position the organization's logo in a consistent location on all pages, most commonly the top-left corner. This provides a confirmation for viewers about where they are or where they may have gone. To identify, if necessary, the site's purpose, locate a one-sentence tagline (phrase or short sentence) below the masthead.

**Content and purpose.** The homepage should clearly identify the Web site's content and purpose. Assume that potential users know nothing about the site and what is presented within it. Describe who the site is intended for or of interest to. Describe the kinds of information available. The homepage must convince users to stay by telling them what they will find within the site.

**Elements.** Other elements to include on the homepage include a site overview or map, navigational links to most (if not all) of the site or its major sections, a small amount of useful content, and a search facility. The homepage may also be used to promote site areas that should be seen, and new or changed content.

## Page Elements

Several items might be included within any Web site's pages. Table 13.1 provides a summary of possible components.

**Table 13.1:** Possible Web Page Components

COMPONENT	PURPOSE	PAGE FREQUENCY	LOCATION
Page title	To clearly identify and describe page.	Every page	Top
Navigation bar	To allow global site navigation.	Every page	Top
Table of contents	To list page contents.	If longer than 2 to 3 screens	Top
Site identifier (trademark, logo, or organizational identification)	To identify page's owner. A strong navigational and orientation cue.	Every page	Top
Search facility	To provide a means for locating content of interest.	Every page	Top

(continued)

**Table 13.1** (*continued*)

COMPONENT	PURPOSE	PAGE FREQUENCY	LOCATION
Page's author or contact person	To identify page's author, or to indicate page's contact person.	Every page	Footer
Contact e-mail address	To solicit queries or comments.	Every page	Footer
Comment facility	To solicit queries or comments.	As necessary	Footer
Other contact details (telephone number, mailing address, and so forth)	To identify other methods for soliciting queries or comments.	As necessary	Footer
Copyright information	To identify page's legal ownership. To caution against unauthorized use.	Every page	Footer
Date of creation or update	To indicate currency of information.	Every page	Footer
<b>Links to:</b>			
Skip to main content	An accessibility consideration.	Every page	Top
Other major sections of site	To provide easy access to all major site content.	Every page	Top, footer also for long pages
Homepage	To return to homepage.	Every page	Top-left, footer also for long pages.
Index page	To return to index.	Every page	Top, footer also for long pages
Site map or directory	To return to site map or directory.	Every page	Top, footer also for long pages
Next page	To go to next page in a sequence.	Every page (if applicable)	Bottom
Previous page	To return to previous page.	Every page (if applicable)	Bottom

Shaihk and Lenz (2006) investigated, based upon user's experiences, their expectations concerning locations of several Web page elements. They found that the most common expectation for the "Back-to-home" link was the upper-left corner. The "Site Search" facility most common expectation was in the far upper-right followed by the upper-left location just below the logo. Totally, a combined general upper-left location including the logo had a higher expectancy than upper-right (24% versus 17%).

## Screen Examples

Following are more examples of good and poor design. Included are redesigns of the screens critiqued in Step 3.

### Example 1

Here is a poorly designed screen and its redesigned version.

#### Screen 1.1

This is a poorly designed screen. The captions are not discernible from choice descriptions, and the initial choice descriptions are not left-aligned. The radio buttons and check boxes are not strongly associated with their respective descriptions, and they also *follow* their respective descriptions, certainly causing selection confusion. The horizontal orientation of choices is not efficient for human scanning. No perception of groupings exists. The ordering scheme of Family, Style, and Color is questionable. Alphabetic ordering would seem to be more appropriate.

Screen 1.1

### Screen 1.2

This is a much better screen. The title is capitalized to set it off from the remainder of the screen. The radio buttons and check boxes are arrayed vertically to facilitate scanning and comparison of alternatives. All controls are enclosed in borders to focus the viewer's attention to them. While the overall organization does not assist the viewer in establishing a scanning direction (horizontal or vertical), the kind of information presented does not make this critical. The screen can be effectively used from left to right or from top to bottom. Family, Style, and Color are alphabetized. Using the complexity measure discussed in Step 3, this redesigned screen is 42 percent less complex than the original.

**TEXT PROPERTIES**

**Family**

- ☐ Courier
- ☐ Helvetica
- ☐ Sans Serif
- ☐ Times

**Pitch**

- ☐ 10 CPI
- ☐ 12 CPI
- ☐ 15 CPI
- ☐ Proportional

**Border**

- ☐
- ☐
- ☐
- ☐

**Size**

- ☐ Small
- ☐ Medium
- ☐ Large

**Style**

- ☐ Bold
- ☐ Italic
- ☐ Underline

**Color**

- ☐ Black
- ☐ Blue
- ☐ Green
- ☐ Red

OK Apply Cancel Help

**Screen 1.2**

### Example 2

This is a representation of an actual screen from Microsoft Windows. Two alternate designs are presented.



### Screen 2.1

This is a screen with several faults. On a positive note, the captions on the left side are nicely aligned, as are the top four text boxes. The box alignment, however, breaks down in the middle of the screen. Also, what appear to be captions (because they possess an ending colon) are really headings, communicating a false message to the viewer (Memory Requirements, EMS Memory, and XMS Memory). The word “memory” repeated four times in succession seems redundant, indicating the potential for a heading. One radio button field (Video Memory) is arrayed horizontally, the others (Display Usage and Execution) are arrayed vertically. The control “Close Window on Exit” seems lost.

Screen 2.1

### Screen 2.2

This is a much-improved alternative. Groupings of elements are provided. Section borders, with titles, are included in the upper part of the screen to strengthen the perception of groupings. Control borders in the lower part of the screen serve the same purpose. Proper alignment of data fields is achieved in the top two sections of the screen. The redundant word “memory” is incorporated as a section heading. Section headings are displayed capitalized to distinguish them from control captions. Subsection headings are created in the Memory section where the heading-caption confusion previously existed. Subsection headings are set off by capitalization and arrows.

The radio buttons/check boxes at the bottom of the screen are arrayed horizontally to provide screen balance. The “Close Window on Exit” control field is given an (admittedly redundant) caption to allow a control border consistent with its neighbors and to create screen balance. The Video (Memory) control remains, as a trade-off, arrayed horizontally. It would have been desirable to organize its choices vertically, but the best overall fit within the screen is achieved by horizontal orientation. This redesigned version of the screen is actually 4 percent *more* complex than the original. The addition of headings and subheadings added to its complexity measure. In spite of this, it is a better screen. Additional information added to a screen to aid understanding can sometimes increase its complexity. So, use the complexity measure as a guide, not as an absolute and final measure of a screen’s effectiveness.

The screenshot shows a window titled "PIF EDITOR". It contains several sections for configuring a program's properties:

- APPLICATION**: Includes fields for "Program Filename:", "Window Title:", "Optional Parameters:", and "Start-up Directory:", each with a corresponding text input box.
- MEMORY**: Includes fields for "REAL > Required:", "Desired:", "EMS > Required:", "Limit:", and "XMS > Required:", "Limit:", each with a text input box followed by "KB".
- VIDEO > Type:**: Includes three radio buttons: "Text", "Low Graphics", and "High Graphics" (which is selected).
- Display Usage**: Includes two radio buttons: "Full Screen" and "Windowed" (which is selected).
- Execution**: Includes two radio buttons: "Background" (which is selected) and "Exclusive".
- Window**: Includes a checkbox labeled "Close on Exit".

**Screen 2.2**

### Screen 2.3

Here is another redesigned version of this screen. The Memory section has been restructured to maintain a top-to-bottom flow. The trade-off is that two columns are now required to present the information. This version is 8 percent more complex than the original, again because of the added information. Which version do you prefer, 2.2 or 2.3?

**PIF EDITOR**

**APPLICATION**

Program Filename:

Window Title:

Optional Parameters:

Start-up Directory:

**MEMORY**

**Real**

Required:  KB

Desired:  KB

**EMS**

Required:  KB

Limit:  KB

**XMS**

Required:  KB

Limit:  KB

**Video**

☐ Text

☐ Low Graphics

☐ High Graphics

**Display Usage**

☐ Full Screen

☐ Windowed

**Execution**

☐ Background

☐ Exclusive

**Window**

☐ Close on Exit

**Screen 2.3**

More examples and an exercise for Step 13 can be found on this book's companion Web site, [www.wiley.com/college/galitz](http://www.wiley.com/college/galitz).



## Test, Test, and Retest

The design of graphical systems and Web pages, and their screens, is a complicated process. As has been shown, in both a host of factors must be considered. In graphical systems, among the many design elements are the types of windows used, the way the windows are organized, what controls are selected to collect and present information, and the way the controls are organized within one window and between several windows. Web page design factors include the proper integration of text, graphics, navigation links, and controls, page size, writing for simplicity and clarity, the characteristics of browsers and monitors, and accessibility requirements. In both design processes numerous design trade-offs will be made. Also, some design decisions may be based on skimpy data and reflect the most educated guess possible at the moment. Finally, the implications for some design decisions may not be fully appreciated until the results can be seen.

To wait until after a system has been implemented to uncover and correct any system usability deficiencies can be aggravating, costly, and time-consuming for both users and developers. Indeed, after implementation many problems may never be corrected because of time constraints and costs. To minimize these kinds of problems and ensure usability, interfaces must be continually tested and refined before they are implemented.

What follows is an overview of the usability testing process and the role it plays in design. Its purpose is to provide an awareness of the testing procedures and methods, and to summarize some basic testing guidelines. Sometimes tests are referred to as evaluations. These terms are generally interchangeable. In this discussion the term testing will be used. The testing steps to be reviewed are:

- Identifying the purpose and scope of testing.
- Understanding the importance of testing.
- Developing a prototype.
- Developing the right kind of test plan.
- Designing a test to yield relevant data.
- Soliciting, selecting, and scheduling users to participate.
- Providing the proper test facility.
- Conducting tests and collecting data.
- Analyzing the data and generating design recommendations.
- Modifying the prototype as necessary.
- Testing the system again.
- Evaluating the working system.

## Usability

---

The concept of usability, a common theme running through this text, was introduced and defined in the introduction to Part 2. By way of a summary, the following dimensions of usability were described by Quesenbery (2003):

**Effective.** The completeness and accuracy with which users achieve their goals.

**Efficient.** The speed and accuracy with which users can complete their tasks.

**Engaging.** The degree to which the tone and style of the interface makes the product pleasing or satisfying to use.

**Error tolerant.** How well the design prevents errors and helps with recovery from those that do occur.

**Easy to learn.** How well the product supports both initial orientation and an increase in the understanding of its capabilities.

## The Purpose of Usability Testing

Usability testing serves a twofold purpose. First, it establishes a communication bridge between developers and users. Through testing, the developer learns about the user's goals, perceptions, questions, and problems. Through testing, the user is exposed to the capabilities of the system early on, before design is solidified.

Second, testing is used to evaluate a product. It validates design decisions. It also can identify potential problems in design at a point in the development process where they can be more easily addressed. Testing also enables comparison of alternate versions of a design element, when a clear direction is not immediately evident. How well the interface and screens meet user needs and expectations can also be assessed.

Thorough testing also has one other benefit for the developer. It can prevent the massive embarrassment that often results from letting things "slip through the cracks."

## The Importance of Usability Testing

A thorough usability testing process is important for many reasons, including the following:

**Developers and users possess different models.** As discussed earlier, developers and users have different expectations and levels of knowledge. Specialized knowledge possessed by the developers enables them to deal with complex or ambiguous situations on the basis of context cues not visible to the users. Developers also frequently use terminology that does not always match that of the users.

**Developer's intuitions are not always correct.** The intuition of designers, or anyone for that matter, no matter how good or bad they may be at what they do, is error prone. This is illustrated by the previously reported Tullis and Kodimer (1992) study evaluating several screen-based controls. They found that programmers' predictions of control usage speed correlated only .07 with actual measured speeds. They also found that programmers' predictions of user control preferences correlated only .31 with actuality. Intuition is too shallow a foundation on which to base design decisions.

**There is no average user.** We all differ — in looks, feelings, motor abilities, intellectual abilities, learning abilities and speeds, device-based control preferences, and so forth. In a keyboard data entry task, for example, the best operators will probably be twice as fast as the poorest and make 10 times fewer errors. The design must permit people with widely varying characteristics to satisfactorily and comfortably learn and perform the task or job.

**It's impossible to predict usability from appearance.** Just as it is impossible to judge a person's personality from his or her looks, it's impossible to predict a system's usability from its appearance.

**Design standards and guidelines are not sufficient.** Design standards and guidelines are an important component of good design, laying the foundation for consistency. But design standards and guidelines often fall victim to trade-offs. They also cannot address all the countless design element interactions that occur within a completed system.

**Informal feedback is inadequate.** Informal feedback is a hit-and-miss proposition. Parts of the system may be completely overlooked; significant problems in other parts may never be documented.

**Products' built-in pieces almost always have system-level inconsistencies.** This is a normal and expected result when different developers work on different aspects of a system. We might also say that developers differ — there is no average developer.

**Problems found late are more difficult and expensive to fix.** Unless they're really severe, they may never be fixed.

**Problems fixed during development mean reduced support costs later.** Support costs are directly proportional to the usability problems that remain after development. The more problems, the higher the support costs.

**Advantages over a competitive product can be achieved.** Many products can do something. The most successful products are those that permit doing something easily.

**MAXIM** Test early, test often.

## Scope of Testing

Testing should begin in the earliest stages of product development. It is an iterative process performed throughout a product's design cycle. Paper or software prototypes are created and tested, and then design changes are made based on the test results. The redesigned product is then tested again, and the process repeated as necessary until all performance goals are met. At this point the iterative process ends. Testing should include as many of the user's tasks, and as many of the product's components, as reasonably possible.

Rubin (1994) classifies tests into four types: exploratory, assessment, comparison, and validation. Each test type has a different objective and may be applied at different times in the system life cycle.

### ***Exploratory Evaluations***

Exploratory evaluations, often informal, are conducted early in the system development process. Their purpose is to

- Explore the user-interface design features of a prototype.
- Gather feedback on preliminary designs.
- Verify the assumptions about users derived during requirements determination.

The data obtained in an exploratory evaluation is mostly qualitative in nature, and is primarily based on discussions with users. Prototypes typically used include *hand sketches and scenarios*, and *interactive paper prototypes*.

### ***Assessment Evaluations***

Assessment evaluations are carried out early or midway in the development process after a conceptual model has been created that incorporates information gathered during the exploratory evaluation. Their purpose is to

- Establish how well user tasks are supported.
- Determine what usability problems may exist.

The evaluations are conducted using user task descriptions, and measures of the level of usability of the system can be obtained. The outcome of this evaluation may result in a refinement of the system's requirements.



## Comparison Evaluations

A comparison evaluation may be performed at any stage in the development process. When two or more design alternatives exist, either of which may appear possible, a *classic experiment* may be developed to compare them directly. Two or more prototypes are constructed, identical in all aspects except for the design issue (type of control, wording of an instruction, and so forth). Speed and accuracy measures are collected and user preferences solicited.

## Validation Evaluation

Validation evaluations are conducted toward the end of the development cycle or once the system is in use. Their purpose is to ascertain that the system meets a predetermined usability objective. They may also be conducted to determine how well all of the components of a system work together. The result of this evaluation is a determination about whether the components of the interface do or do not meet the required levels of performance.

Always involve all members of the design team in the testing to ensure a common reference point for all. Involving all members also permits multiple insights into the test results from the different perspectives of team members.

## Prototypes

---

A prototype, a simple, incomplete model or mock-up of a design, is primarily a vehicle for exploration, communication, and evaluation. Its purpose is to obtain user input in design, and to provide feedback to designers. Its major function is the communicative role it plays, not accuracy or thoroughness. A prototype enables a design to be better visualized and provides insights into how the software will look and work. It also aids in defining tasks, their flow, the interface itself, and its screens.

A prototype is a simulation of an actual system that can be quickly created. A prototype may be a rough approximation, such as a simple hand-drawn sketch, or it may be interactive, allowing the user to key or select data using controls, navigate through menus, retrieve displays of data, and perform basic system functions. A prototype need not be perfectly realistic, but it must be reasonably accurate and legible. A prototype also need not be functionally complete, possessing actual files or processing data. Today many software support tools for prototyping are available that permit the prototype to be integrated directly into the application code.

A prototype may have great breadth, including as many features as possible to present concepts and overall organization, or it might have more depth, including more detail on a given feature or task to focus on individual design aspects. By nature, a prototype cannot be used to exercise all of a system's functions, just those that are notable in one manner or another.

Particularly useful early in design, a prototype should be capable of being rapidly changed as testing is performed. A prototype is characterized by its fidelity, the exactness and thoroughness of its replication of a system's screens and user interaction. Prototypes range in fidelity from low to high, from rough hand-drawn sketches to fully functioning software (Microsoft, 1995; Weinschenk, 1995; Winograd, 1995). Low-fidelity prototypes are usually paper-based and include sketches and scenarios, and interactive paper prototypes. They can be prepared by hand or by using a drawing package like Microsoft Paint or PowerPoint. High-fidelity prototypes are software based, yielding a functional version of the system that can be interacted with. Various kinds of prototypes, in general order of increased fidelity, are as follows.

## Hand Sketches and Scenarios

---

- **Description:**
    - Screen sketches created by hand or a drawing package.
    - Focus is on the design, not the interface mechanics.
    - A low-fidelity prototype.
  - **Advantages:**
    - Can be used very early in the development process.
    - Suited for use by entire design team.
    - No large investment of time and cost.
    - No programming skill needed.
    - Easily portable.
    - Fast to modify and iterate.
    - A rough approximation often yields more substantive critical comments.
    - Easier to comprehend than functional specifications.
    - Can be used to define requirements.
  - **Disadvantages:**
    - Only a rough approximation.
    - Limited in providing an understanding of navigation and flow.
    - A demonstration, not an exercise.
    - Driven by a facilitator, not the user.
    - Limited usefulness for a usability test.
    - A poor detailed specification for writing the code.
    - Usually restricted to most common tasks.
- 

**Description.** The first, and simplest, prototype is a low-fidelity rough hand-drawn sketch, or mock-up, of the screens. These can start early in the design process and before any attempt is made to create a prototype using an available toolkit or interface builder. With sketches, the focus is on the design of individual screens, not the interface mechanics. The hand sketch should be an entity that has enough of a general look to suggest the functionality, interaction, and layout of screens. The goal is a rough vision, not a polished work of art. This sketch will be useful in defining and refining task organization, conceptual ideas, and the general layout of screens.

**MYTH** The design is finished.

**Advantages.** Hand-drawn sketches of screens can be easily developed and used very early in the development process. Many usability problems can then be identified and corrected quickly. Sketches are also suitable for use by the entire development team, giving everyone a sense of what the real design issues are. Sketches require no large investment of time and money, and they are portable, placing few restrictions on where the testing may occur. Sketches can also be quickly modified and iterated, as many times as necessary. Because there has been no emotional investment in code and the status quo, there is no necessity for team members to defend something already created from hard work. Screen sketches are rough approximations, and rough approximations often yield more substantive suggestions or critical comments than actual screen-drawn versions. Their draft or unpolished look greatly softens the attitude that everything is cast in concrete. Sketches can also be used to define a system's requirements.

**Disadvantages.** Because hand-drawn sketches are rough approximations, they can only suggest the final layout of the interface. They are limited in helping understand system navigation and flow, and are a demonstration device driven by a facilitator, with the user assuming a more passive role. They are usually restricted to the most common user tasks. As a result, they possess limited usefulness for usability testing.

***Sketch Creation Process\****

- 
- Sketch (storyboard) the screens while determining
    - The source of the screen's information.
    - The content and structure of individual screens.
    - The overall order of screens and windows.
  - Use an erasable medium.
  - Sketch the screens needed to complete each workflow task.
  - Try out selected metaphors and change them as necessary.
  - Sketch common/critical/frequent scenarios first.
    - Follow them from beginning to end.
    - Go back and build in exceptions.
  - Don't get too detailed; exact control positioning is not important, just overall order and flow.
  - Sketch storyboard as a team, including at least one user.
  - Develop online prototypes only when everyone agrees that a complete set of screens has been satisfactorily sketched.
- 

\* Based on Weinschenk (1995).

Storyboards are sequences of sketches or screen layouts that reflect an interaction. Sketch the screens while determining the source of the screen's information, the content and structure of the individual screens, and the overall flow of the screens. Use an erasable medium so that as ideas are explored and changed, the modifications will be easy to make. Sketch the screens needed to complete each task in the workflow. First, sketch the most common, critical, or frequent activities, following them from beginning to end. Then, go back and build in the exceptions. Try out all selected metaphors and modify them as necessary. Make sure the major user objects are very obvious. Avoid getting too detailed. Most important is the overall screen flow, order, and structure. Approximate the positioning of controls simply to verify fit. Exact positioning will come later. Sketch the screens as a team, including at least one user. To avoid solidifying the product too soon, develop online prototypes only when everyone agrees a complete set of screens have been satisfactorily sketched.

## Interactive Paper Prototypes

---

- **Description:**
    - Interface components (menus, windows, and screens) constructed of common paper technologies (Post-it notes, transparencies, and so on).
    - The components are manually manipulated to reflect the dynamics of the software.
    - A low-fidelity prototype.
  - **Advantages:**
    - More illustrative of program dynamics than sketches.
    - Can be used to demonstrate the interaction.
    - Otherwise, generally the same as for hand-drawn sketches and scenarios.
  - **Disadvantages:**
    - Only a rough approximation.
    - A demonstration, not an exercise.
    - Driven by a facilitator, not the user.
    - Limited usefulness for usability testing.
- 

**Description.** Another simple low-fidelity prototype involves use of common office supplies such as Post-It notes, transparencies, markers, and scissors. Menu bars, pull-down menus, pop-up windows, screen bodies, and so on reflecting system tasks are created using these media. Then, the components are manually manipulated to illustrate the dynamics of the software. The purpose of this kind of prototype is to provide a sense of the dynamics of a program without actually having to build a functional version of it. The objective, again, is to create a rough vision, not a polished piece of art, to elicit substantive suggestions and critical comments. The goal is not to show how the system or Web site will look, but to assess if people can easily use it.

**Advantages.** Interactive paper prototypes are more illustrative of program dynamics than simple screen sketches. System components can be manipulated to demonstrate the interactive nature of the system. The other paper prototype

advantages are generally the same as those described for hand-drawn sketches and scenarios.

**Disadvantages.** The disadvantages are similar to those for hand-drawn sketches. They are only rough approximations, are demonstrations and not exercises, are driven by a facilitator not the user, and possess limited usefulness for usability testing.

## Programmed Facades

---

- **Description:**
    - Examples of finished dialogs and screens for some important aspects of the system.
    - Created by prototyping tools.
    - Medium-fidelity to high-fidelity prototypes.
  - **Advantages:**
    - Provide a good detailed specification for writing code.
    - A vehicle for data collection.
  - **Disadvantages:**
    - May solidify the design too soon.
    - May create the false expectation that the “real thing” is only a short time away.
    - More expensive to develop.
    - More time-consuming to create.
    - Not effective for requirements gathering.
    - Not all of the functions demonstrated may be used because of cost, schedule limitations, or lack of user interest.
    - Not practical for investigating more than two or three approaches.
- 

**Description.** To provide a realistic surface view of a real program and to illustrate some of the program’s functioning, a programmed facade can be created. Using prototyping tools such as Microsoft PowerPoint and Visual Basic, examples of finished dialogs and screens for some important aspects of the system are constructed and viewed. A facade is very shallow, duplicating only a small portion of what is ultimately intended in both depth and breadth. Because of this shallow nature, it is best described as a medium-fidelity to high-fidelity prototype.

**Advantages.** While much is missing underneath, what is visible can provide a good impression of finished design. Programmed facades also provide a good detailed specification for writing code, and can be vehicles for the actual collection of data.

**Disadvantages.** First, a highly polished product can foster a sense of finality because of its appearance. Significant reorganization or restructuring suggestions may be inhibited, with the focus simply being on screen cosmetics. Second, the false expectation that the real thing is only a short time away may be easily created, even though much work still remains to be done. Programmed facades are also much more expensive to develop than paper-based prototypes, and they

take much longer to create. Also, not all of the functions demonstrated may eventually be used because of cost, schedule limitations, or lack of user interest, and they are not practical for investigating more than two or three alternate design approaches.

## Prototype-Oriented Languages

---

- **Description:**
    - An example of finished dialogs and screens for some important aspects of the system.
    - Created through programming languages that support the actual programming process.
    - A high-fidelity prototype.
  - **Advantages:**
    - May include the final code.
    - Otherwise, generally the same as those of programmed facades.
  - **Disadvantages:**
    - Generally the same as for programmed facades.
- 

**Description.** To present an example of completed dialogs and screens for some parts of the system, prototypes can be constructed using programming languages that support the actual programming process. Examples include Power Builder, Visual Basic, and so on. Using these tools, a high-fidelity, real program can be created to illustrate some of the program's functioning and the mechanics of the interface. One consideration to be decided up front: Will the prototype software be a "throwaway," or the actual system software? This will have implications concerning the amount of programming effort expended on the prototype.

**Advantages.** The greatest advantage of this kind of prototype is that it may include the actual code needed for the system. Otherwise, advantages are generally the same as those of programmed facades.

**Disadvantages.** Like a programmed facade, a danger is that the highly polished product can foster a sense of finality because of its appearance, inhibiting reorganization or restructuring suggestions, the focus simply being on screen cosmetics. Other disadvantages are also similar to those of programmed facades.

## Comparisons of Prototypes

The fidelity of the prototypes previously described varies from low to high. Does fidelity affect a prototype's usefulness as a testing tool? Most usability experts agree that low-fidelity prototypes are very useful in the early stages of design. Their usefulness in later stages of design has been questioned because (1) they are limited in functionality, usually not including all of a system's features, (2) they usually have little interaction, not permitting use of a mouse and keyboard, and (3) they have limited

presentation features, usually not including screen colors or high-level graphics. Several recent studies have addressed the fidelity issue.

The first study, reported by Virzi et al. (1996), compared a computer-based high-fidelity prototype with a paper-based low-fidelity prototype in performing the exact same activities. They found no statistically significant difference in the number of usability issues detected by each method. They concluded that the two approaches were equally sensitive in identifying the same problems.

Next, Catani and Biers (1998) examined prototypes created at three fidelity levels: low (paper), medium (screen shots), and high (a prototype-oriented language — Visual Basic). There were no significant differences in the number and severity of problems identified with each kind of prototype. There was also a high degree of commonality in the specific problems uncovered.

The third study, reported by Uceta et al. (1998), compared a paper-based prototype with a computer-based prototype. Both interfaces were identical except for the medium of presentation. Most of the usability problems were found using both approaches, the results being statistically comparable. Identifying problems using the paper prototype, however, took about 30 percent longer than using the computer-based prototypes.

Two more recent studies (Walker et al., 2002; Sefelin et al., 2003) also did not find any significant difference between the two contrasting fidelity methods. The latter study, however, found that 92 percent of the participants preferred working with the computer-based prototypes.

The results of these studies indicate that prototype fidelity seems to have no impact on the identification of usability problems. Other prototyping issues (prototype creation time and cost, testing time, and so on) remain important issues in usability testing, however. It seems reasonable that any system development effort should use combinations of these prototyping techniques throughout the entire design cycle to visualize the design, solicit users' input, and obtain needed feedback for the developers. Successively increasing the complexity of the prototypes as development moves toward the final solution, allows users to continually get a better idea of how the system will actually look and work. This will give them greater and greater insights into how the system may be improved. The reader desiring more information on paper prototyping is referred to Snyder (2003).

## **Kinds of Tests**

---

A test is a tool that is used to measure something. The “something” may be

- Conformance with a requirement.
- Conformance with guidelines for good design.
- Identification of design problems.
- Ease of system learning.
- Retention of learning over time.
- Speed of task completion.

- Speed of need fulfillment.
- Error rates.
- Subjective user satisfaction.

A test is usually formal; it is created and applied intentionally and with a purpose. It is usually based upon some kind of criteria, an understanding of what a good result would be. A test is usually comprised of one or more of the following elements: observe, compare, listen, and measure (Whitelock, 1998; Stone et al., 2005).

**Observe.** Most tests include some form of observation. It may be watching the user perform tasks, or an evaluator or inspector reviewing or evaluating the aspects of a system.

**Compare.** The interface is compared with standards, guidelines and/or, usability requirements. It may also be compared to the evaluator's knowledge and experience concerning the qualities of a good interface.

**Listen.** The user often is asked to provide comments or opinions concerning the usability of the interface. Evaluators often make similar statements. Listening may be informal, such as being a response to a question, or may be more formal, collected from listening to an audio or video recording. Responses may also be gathered indirectly by reading responses to a questionnaire.

**Measure.** Measures may be taken to ascertain how well the interface complies with usability requirements. Quantitative values such as "time to complete a task" or "number of errors" are collected.

Several testing techniques, at varying levels of sophistication and cost, are available to exercise the system. These techniques include guideline and standard reviews, heuristic evaluations, cognitive walk-throughs, think-aloud evaluations, usability tests, classic experiments, and focus groups. Table 14.1, derived from Stone et al. (2005), relates these four elements to the testing tools (and in some cases components of these tools) that are now going to be reviewed.

**Table 14.1:** Testing Elements and Testing Tools

TEST ELEMENT	TESTING TOOL
Observe	Direct observation of participants by being next to and watching them.
	Indirect observation of participants by
	- Watching video recordings.
	- Watching the participant through a one-way mirror in a test lab.
	- Using eye-tracking equipment to show precisely where the user is looking.
	- Using software to replicate the user's screen on another monitor so screen changes can be seen as the user makes them.



**Table 14.1** (*continued*)

TEST ELEMENT	TESTING TOOL
Compare	With the user's personal concept of what constitutes a good interface.
	With heuristics.
	With design principles.
	With design standards, guidelines or style guides.
	With usability standards.
Listen	To the user thinking aloud.
	To responses to cognitive walk-through questions.
	To post-session interviews or debriefings.
	To participant responses describing or recalling what they did.
	To the user's opinions.
Measure	To the responses on questionnaires.
	Times to perform tasks.
	Tasks successfully completed.
	Errors made.
	User satisfaction.

Reprinted from User Interface Design and Evaluation, Stone et al., Table 27.2, p. 542 (2005), with permission from Elsevier.

## Guidelines and Standards Review

- Description:
  - A review of the interface in terms of
    - An organization's standards, design guidelines, or style guide.
    - A government standard.
- Advantages:
  - Can be performed by developers.
  - Low cost.
  - Can identify general and recurring problems.
  - Particularly useful for identifying screen design and layout problems.
- Disadvantages:
  - May miss severe conceptual, navigation, and operational problems.

**Description.** A *guideline* or *standards review* is an inspection of an interface's navigation, screen design and layout, and accessibility in the context of an organization's standards and design guidelines or a governmental standard. A checklist summarizing a system's standard or guideline document is prepared and is used as the basis for the comparison. Failure to comply with a guideline or standard indicates that a design modification may be necessary.

**Advantages.** Software developers can perform this kind of test. Its advantages include its low cost and its ability to identify recurring and general problems. It is particularly useful in identifying screen design and layout problems.

**Disadvantage.** Because this review tends to be static in nature, it may miss severe conceptual, navigation, and operational problems.

## Heuristic Evaluation

---

- **Description:**
    - A detailed evaluation of a system by interface design specialists to identify problems.
  - **Advantages:**
    - Easy to do.
    - Relatively low cost.
    - Does not waste user's time.
    - Can identify many problems.
  - **Disadvantages:**
    - Evaluators must possess interface design expertise.
    - Evaluators may not possess an adequate understanding of the tasks and user communities.
    - Difficult to identify systemwide structural problems.
    - Difficult to uncover missing exits and interface elements.
    - Difficult to identify the most important problems among all problems uncovered.
    - Does not provide any systematic way to generate solutions to the problems uncovered.
  - **Guidelines:**
    - Use three to five expert evaluators.
    - Choose knowledgeable people
      - Familiar with the project situation.
      - Possessing a long-term relationship with the organization.
- 

**Description.** In a *heuristic evaluation*, interface specialists study a system in depth and look for properties they know, from experience, will lead to problems. The interface is judged for its compliance with recognized usability principles, the heuristics. These evaluations are also referred to as inspections and expert reviews.

**Advantages.** A heuristic evaluation is relatively easy to do and its cost is fairly low (see "Heuristic Evaluation Process" in the next section). It does not waste the

user's valuable time. It also can identify many usability problems (see the research studies that follow).

**Disadvantages.** Ideally, the evaluators should possess user interface expertise; this greatly restricts the population from which evaluators are chosen. Because of the small size of this available group, evaluators may not possess an adequate understanding of the system tasks and the user communities. (Even expert reviewers have great difficulty knowing how typical users, especially first-time users, will really behave.)

With this method of evaluation, it is difficult to identify deeper design problems, including systemwide structural problems, missing exits, and missing interface elements or features. It is also difficult to identify the most important problems among those documented. Finally, this method does not provide any systematic way to generate solutions to the problems uncovered.

**Guidelines.** Based upon a study, Nielsen (1992) suggests that the optimum expert group size to satisfactorily perform a heuristic evaluation is three to five people. He found that different evaluators tend to find different problems, and one person will never be able to find the number of usability problems that several people will. Nielsen also found that including more than five evaluators does not gain enough additional usability information for the extra cost involved.

Ideally, evaluators used should be familiar with the project situation, possess a long-term relationship with the developing organization, and possess expertise in human-computer interface design. This suggestion is often difficult to fully achieve, however. Because of this difficulty, Stone et al. (2005) suggests that the evaluation team can also be comprised of people with the following experience:

- Usability experts experienced in conducting evaluations.
- Domain experts, people with knowledge of the domain (including users).
- Designers with extensive design experience.
- Developers without any formal usability training, but who are keen to explore the usability defects that users might experience.
- Nonexperts, people who are neither usability nor domain experts, but who understand what is to be done and are willing to help. They may be experts in their own particular domains.

One study, however, has found that more-experienced usability practitioners tend to discover more problems in an expert review than less-experienced practitioners (Huart et al., 2004).

### ***Heuristic Evaluation Process***

- 
- Preparing the session:
    - Select evaluators.
    - Prepare or assemble
      - A project overview.
      - A checklist of heuristics.

- Provide briefing to evaluators to
    - Review the purpose of the evaluation session.
    - Preview the evaluation process.
    - Present the project overview and heuristics.
    - Answer any evaluator questions.
    - Provide any special evaluator training that may be necessary.
  - Conducting the session:
    - Have each evaluator review the system alone.
    - The evaluator should
      - Establish own process or method of reviewing the system.
      - Provide usage scenarios, if necessary.
      - Compare his or her findings with the list of usability principles.
      - Identify any other relevant problems or issues.
      - Make at least two passes through the system.
    - Detected problems should be related to the specific heuristics they violate.
    - Comments are recorded either
      - By the evaluator.
      - By an observer.
    - The observer may answer questions and provide hints.
    - Restrict the length of the session to no more than 2 hours.
  - After the session:
    - Hold a debriefing session including observers and design team members where
      - Each evaluator presents problems detected and the heuristic it violated.
      - A composite problem listing is assembled.
      - Design suggestions for improving the problematic aspects of the system are discussed.
    - After the debriefing session:
      - Generate a composite list of violations as a ratings form.
      - Request evaluators to assign severity ratings to each violation.
      - Analyze results and establish a program to correct violations and deficiencies.
- 

**Preparing the session.** First, as previously described, *select* three to five evaluators. Ideally, the evaluators used should be familiar with the project situation and possess a long-term relationship with the developing organization. Then, *prepare or assemble* a project overview and a checklist of heuristics. A useful checklist may be available from one or more of the evaluators. Examples of checklists are found in Tables 14.3 and 14.4. Finally, provide a *briefing* to the evaluators to review the purpose of the evaluation session, to preview the entire evaluation process that is being undertaken, to present the project overview and heuristics, and to answer any questions the evaluators may have. Any special evaluator training that may be necessary can also be provided at this time. This briefing session will normally consume 45 to 60 minutes.

**Conducting the session.** Each evaluator should inspect the system *alone*, not with or in the presence of other evaluators. This is to ensure independent and unbiased evaluations from each evaluator. Allow the evaluators to establish their *own process or method* of reviewing the system. Ideally, let the evaluator use the system

without procedural assistance, browsing as is felt necessary. If, however, the evaluators are not familiar with the system's content and purpose, they may be provided with scenarios listing the steps a user would take to perform a sample set of realistic user tasks.

During the session, the evaluators will compare their findings with the list of usability principles. *Detected problems* should be related to the specific heuristics they violate. Multiple heuristics can be linked to any one identified problem. Other relevant problems or issues can also be noted. Two or more passes should be made through the system.

Evaluator *comments* can be recorded either by the evaluator or by an observer. Evaluator-written comments require additional effort by the evaluator during the review process and can break the continuity of the evaluation process. Having an observer record the evaluator's verbal comments adds to the overhead of the session, but reduces the evaluator's workload and allows greater focusing on the review process. If the same observer is used for all evaluation sessions, session results should be available faster because the observer only needs to understand and organize one set of personal notes, not multiple reports or reviews written by all session participants. The observer may answer questions and provide hints to the evaluator, as necessary. Evaluators not familiar with the system's content will occasionally need advice. Also, precious time will not be wasted by the evaluator's struggling with the interface's mechanics. Observer comments should be restricted to situations where the evaluator is clearly confused or in trouble.

**MAXIM** Not even the most brilliantly conceived and ingenious computer system can do all that it was designed to do — or even a small part of what it was designed to do — unless the brilliance of its operation and purpose is matched by the cunning simplicity of its user interface (Hicks and Essinger, 1991).

Finally, to minimize evaluator fatigue, restrict the *length of a session* to about 2 hours. For large or complicated interfaces that require more time to evaluate, it is better to split the session into several sessions, each concentrating on a part of the interface.

**After the session.** When all evaluator and/or observer notes have been compiled, hold a *debriefing session*, no more than 2 hours in length. Include all observers and design team members. Have each evaluator present the problems detected and the heuristic each violated. Assemble a composite list of usability problems (if one has not yet been compiled). Solicit and discuss design suggestions for improving the problematic aspects of the system.

*After the debriefing session*, form the composite list of violations into a ratings form. Request the evaluators to assign severity ratings to each violation on a scale ranging from "usability catastrophe" to "not a usability problem," as shown in Table 14.2. It is difficult to obtain these estimates during the evaluation process, where the focus is on finding problems. Then, analyze the results and establish a program to correct the necessary violations and deficiencies. The ratings will identify the most serious problems, the first in need of attention.

**Heuristic Evaluation Effectiveness**

One of the earliest papers addressing the effectiveness of heuristic evaluations was by Nielsen (1992). He reported that the probability of finding a *major* usability problem averaged 42 percent for single evaluators in six case studies. The corresponding probability for uncovering a *minor* problem was only 32 percent. He also found that the actual number of located minor problems exceeded the number of major problems uncovered by about a 3:1 ratio (152 minor problems versus 59 major problems). Minor design problems are normally more prevalent in design (for example, inconsistent use of typefaces) and, consequently, show up in greater numbers in this kind of evaluation. Minor problems, such as inconsistencies, are more susceptible to identification by inspection, and may not be noticed in testing with actual users, who are focusing on a task or procedure. Nielsen suggests that severity ratings are especially useful in prioritizing minor problems.

**Table 14.2:** Severity Ratings in Heuristic Evaluation

0 = I don't agree that this is a usability problem at all.
1 = A cosmetic problem only. Need not be fixed unless extra time is available.
2 = A minor usability problem. Fixing should be given a low priority.
3 = A major usability problem. Important to fix and should be given a high priority.
4 = A usability catastrophe. Imperative to fix before the product can be released.

From useit.com

**Table 14.3:** Research-Based Set of Heuristics

1. Automate unwanted workload. <ul style="list-style-type: none"><li>• Free cognitive resources for high-level tasks.</li><li>• Eliminate mental calculations, estimations, comparisons, and unnecessary thinking.</li></ul>
2. Reduce uncertainty. <ul style="list-style-type: none"><li>• Display data in a manner that is clear and obvious.</li></ul>
3. Fuse data. <ul style="list-style-type: none"><li>• Reduce cognitive load by bringing together lower-level data into a higher-level summation.</li></ul>
4. Present new information with meaningful aids to interpretation. <ul style="list-style-type: none"><li>• Use a familiar framework, making it easier to absorb.</li><li>• Use everyday terms, metaphors, and so on.</li></ul>
5. Use names that are conceptually related to functions. <ul style="list-style-type: none"><li>• Context-dependent.</li><li>• Attempt to improve recall and recognition.</li></ul>
6. Group data in consistently meaningful ways to decrease search time.

**Table 14.3** (*continued*)

---

7. Limit data-driven tasks.
<ul style="list-style-type: none"> <li>• Reduce the time needed to assimilate raw data.</li> <li>• Make appropriate use of color and graphics.</li> </ul>
8. Include in the displays only that information needed by a user at a given time.
<ul style="list-style-type: none"> <li>• Allow users to remain focused on critical data.</li> <li>• Exclude extraneous information that is not relevant to current tasks.</li> </ul>
9. Provide multiple coding of data where appropriate.
10. Practice judicious redundancy.
<ul style="list-style-type: none"> <li>• To resolve the conflict between heuristics 6 and 8.</li> </ul>

---

From Gerhardt-Powals (1996).

For many years, Bailey (2001, 2003d, 2005a) has questioned the effectiveness of heuristic evaluations as now conducted. In a 1992 article (Bailey et al., 1992), he suggested that many of the problems identified by heuristic evaluations were really not problems at all. He bolstered his argument by reporting on three studies looking at their effectiveness (Catani and Biers, 1998; Rooden et al. 1999; Stanton and Stevenage, 1998). In each study, he determined what the heuristic evaluators thought the problems were, and then he compared these determinations with the problems users actually had in performance testing. The results showed that about one-third (36 percent) of identified problems were actually usability problems, and almost one-half (43 percent) of the problems identified by evaluators *did not* turn out to be problems at all. The evaluators also missed about one-fifth (21 percent) of the problems users actually had. He concluded that when a heuristic evaluation is conducted, about one-half of the problems identified will be problems, and one-half will not be problems.

As a result of these and other studies, experts (Koyani, 2004), and Bailey, recommend that heuristic evaluations/expert reviews should be used with caution because they detect far more potential problems than actually exist, as compared to performance testing. They should not be exclusively relied upon in evaluating usability. They are very useful, however, for identifying potential usability issues to evaluate during usability testing and are but one part in the system-testing arsenal. Bailey has also recommended that the heuristics themselves must be greatly improved. (The messenger is all right; the problem is the message.) He recommends establishing more solidly-research-based heuristics. The most used set of heuristics can be traced back to a paper by Molich and Nielsen in 1990, and revised by Nielsen in 1994 (Bailey, 1999b). The research foundation of these papers is somewhat suspect. Bailey suggests a better research-based set of heuristics will lead to improved evaluation results, for example, those proposed by Gerhardt-Powals (1996). This set of heuristics is summarized in Table 14.3

Web heuristics are still an evolving entity and must also be validated through research. The set proposed by Levi and Conrad (1996), and summarized in Table 14.4 seem a good place to start.

In conclusion, heuristic evaluations are useful in identifying many usability problems and should be part of the testing arsenal. Performing this kind of evaluation before beginning actual testing with users will eliminate many design problems, and is but one step along the path toward a very usable system.

**Table 14.4:** Possible Web Page Heuristics

- 
1. Speak the user's language.
    - Use familiar words, phrases, and concepts.
    - Present information in a logical and natural order.
  2. Be consistent.
    - Indicate similar concepts through identical terminology and graphics.
    - Adhere to uniform conventions for layout, formatting, typefaces, labeling, and so on.
  3. Minimize the user's memory load.
    - Take advantage of recognition rather than recall.
    - Do not force users to remember key information across documents.
  4. Build flexible and efficient systems.
    - Accommodate a range of user-sophistication and diverse-user goals.
    - Provide instructions where useful.
    - Lay out screens so that frequently-accessed information is easily found.
  5. Design aesthetic and minimalist systems.
    - Create visually pleasing displays.
    - Eliminate information that is irrelevant or distracting.
  6. Use chunking.
    - Write materials so that documents are short and contain only one topic.
    - Do not force the user to access multiple documents to complete a single thought.
  7. Provide progressive levels of detail.
    - Organize information hierarchically, with more general information appearing before more specific detail.
    - Encourage the user to delve as deeply as needed, but to stop whenever sufficient information has been obtained.
  8. Give navigational feedback.
    - Facilitate jumping between related topics.
    - Allow the user to determine his/her current position in the document structure.
    - Make it easy to return to an initial state.
  9. Don't lie to the user.
    - Eliminate erroneous or misleading links.
    - Do not refer to missing information.
- 

From Levi and Conrad (1996).

---

## Cognitive Walk-Throughs

---

- Description:
  - Reviews of the interface in the context of tasks users perform.
- Advantages:
  - Allow a clear evaluation of the task flow early in the design process.
  - Do not require a functioning prototype.



- Low cost.
  - Can be used to evaluate alternate solutions.
  - Can be performed by developers.
  - More structured than a heuristic evaluation.
  - Useful for assessing exploratory learning.
  - Disadvantages:
    - Tedious to perform.
    - May miss inconsistencies and general and recurring problems.
  - Guidelines:
    - Needed to conduct the walk-through are
      - A general description of proposed system users and what relevant knowledge they possess.
      - A specific description of one or more core or representative tasks to be performed.
      - A list of the correct actions required to complete each of the tasks.
    - Review:
      - Several core or representative tasks across a range of functions.
      - Proposed tasks of particular concern.
    - Developers must be assigned roles of
      - Scribe to record results of the action.
      - Facilitator to keep the evaluation moving.
    - Start with simple tasks.
    - Don't get bogged down demanding solutions.
    - Limit session to 60 to 90 minutes.
- 

**Description.** In a *cognitive walk-through*, developers or users walk through an interface in the context of representative user tasks. Individual task actions are examined and the evaluators try to establish a logical reason why the user would perform each examined action. Actions are compared to the user's goals and knowledge. Discrepancies and problems are noted and analyzed and the tasks modified as necessary. Walk-throughs require that the task definition methodology must have been properly accomplished in the system requirements stage. The user's goals and assumptions must also be clearly defined before the walk-through is performed.

**Advantages.** Walk-throughs permit a clear evaluation of the task flow early in the design process, before empirical user testing is possible. The earlier a design flaw can be detected, the easier it is to fix it. They can also be used to evaluate alternate design solutions. Walk-throughs are of low cost and can be performed by developers. They are also more structured than a heuristic evaluation, being less likely to suffer from subjectivity because of the emphasis on user tasks. Walk-throughs are very useful for assessing exploratory learning, first-time use of a system without formal training.

**Disadvantages.** A cognitive walk-through is tedious to perform, and it may miss inconsistencies and general and recurring problems. Studies have found that cognitive walk-throughs appear to detect far more problems than actually exist,

compared to performance usability testing results (Koyani et al., 2004). In these studies only about 25 percent of predicted problems turned to be actual problems in a usability test. Thirteen percent of problems were missed in the walk-through. Like heuristic evaluations, walk-throughs appear to be most useful in identifying issues to evaluate in a usability test.

**Guidelines.** Needed to conduct the walk-through are a general description of proposed system users and what relevant knowledge they possess, a specific description of one or more core or representative tasks to be performed, and a listing of the correct actions required to complete each of the tasks. The walk-through should review several core or representative tasks across a range of functions, as well as proposed tasks of particular concern to the developers. Start with simple tasks to get the brain juices flowing. Don't get bogged down looking for solutions to problems. Avoid detailed screen designs; they often inhibit assessment of the big picture. Limit a session to 60 to 90 minutes.

## Think-Aloud Evaluations

---

- **Description:**
    - Users perform specific tasks while thinking out loud.
    - Comments are recorded and analyzed.
  - **Advantages:**
    - Utilizes actual representative tasks.
    - Provides insights into the user's reasoning.
    - Helps users focus and concentrate during the session.
  - **Disadvantages:**
    - Some participants may find thinking aloud distracting and unnatural.
    - Can slow the participants thought processes.
    - Can be exhausting for the participant in a long session.
  - **Guidelines:**
    - Develop
      - Several core or representative tasks.
      - Tasks of particular concern.
    - Limit session to 60 to 90 minutes.
- 

**Description.** In a *think-aloud evaluation*, users perform specific tasks while thinking aloud. The objective is to get the user to talk continuously. All comments are recorded so all thoughts are captured and subtle points are not missed when analysis occurs.

**Advantages.** This kind of evaluation utilizes actual representative tasks. Valuable insights into why the user does things are obtained. Because users are continuously explaining what they are doing, concentration and focus on the task is improved.

**Disadvantages.** It may be difficult to get all people to think aloud because some may find the process distracting and unnatural. Thought processes may also

increase task performance time and reduce the number of errors that would occur in a normal working environment. Talking while working may also be more exhausting than simply working.

**Guidelines.** Develop core or representative task scenarios, or scenarios of proposed tasks of particular concern. Limit a session to 60 to 90 minutes.

## Usability Test

---

- **Description:**
    - An interface evaluation under real-world or controlled conditions.
    - Measures of performance are derived for specific tasks.
    - Problems are identified.
  - **Advantages:**
    - Utilizes an actual work environment.
    - Identifies serious or recurring problems.
  - **Disadvantages:**
    - High cost for establishing facility.
    - Requires a test conductor with user interface expertise.
    - Emphasizes first-time system usage.
    - Poorly suited for detecting inconsistency problems.
- 

**Description.** A *usability test* evaluates an interface under real-world or controlled conditions. Specific tasks are performed by users, and measures of performance are taken, and the results compared with the previously defined performance goals. Evaluators also gather data on problems that arise. Errors, confusion, frustrations, and complaints can be noted and discussed with the user. It is also useful to have the user talk aloud about what he or she is doing. Failure to meet these usability design objectives will indicate that redesign is necessary.

**Advantages.** Usability tests incorporate a realistic work environment. Tasks are performed in an actual work setting — either a usability laboratory or another controlled setting. A usability test can identify serious or recurring problems, avoiding low-priority items.

**Disadvantages.** Its most serious problem is the high cost associated with establishing a facility to perform the testing. To effectively perform a usability test also requires a test conductor with user interface expertise. A usability test also emphasizes first-time or early system use, collecting little data concerning use of a system by experienced system users. These tests are also poorly suited to detecting problems with consistency.

## Performance and Process Measures

A typical usability test collects user-performance data such as task completion rates and times, and error rates. *Performance data* focuses on how well users *can do* their

tasks. Straub (2006) describes another possible measure called process. *Process data* is comprised of measures about what users *are doing* during task completion. This can include both objective and subjective measures such as a participant's self report, an observation of participant behavior, and eye movement tracking. Both of these measures were defined and addressed in a study by Kelkar et al. (2005).

To measure the relevant impact of each measure, these researchers conducted a study consisting of three stages for a Web application. A usability test was followed by a redesign stage in which the test's findings were incorporated into a modified design. A second test plus redesign was then performed. Finally, a third usability test was run. The result: Improvement recommendations based only on test performance data yielded an increase in task completion and a reduction in errors, which is not a surprising conclusion. Recommendations based upon both performance *and* process data yielded increased satisfaction with the interface.

The researchers concluded that the process measures gave the researchers greater insight into the user's expectations and task flow. Access to the user's mental model allowed them to identify and minimize gaps between the user's model and the Web site's interaction model. Based upon this research, process measures may be another useful tool for usability testing.

## Classic Experiments

---

- **Description:**
    - An objective comparison of two or more prototypes identical in all aspects except for one design issue.
  - **Advantages:**
    - Objective measures of performance are obtained.
    - Subjective measures of user satisfaction may be obtained.
  - **Disadvantages:**
    - Requires a rigorously controlled experiment to conduct the evaluation.
    - The experiment conductor must have expertise in setting up, running, and analyzing the data collected.
    - Requires creation of multiple prototypes.
  - **Guidelines:**
    - State a clear and testable hypothesis.
    - Specify a small number of independent variables to be manipulated.
    - Carefully choose the measurements.
    - Judiciously select study participants and carefully or randomly assign them to groups.
    - Control for biasing factors.
    - Collect the data in a controlled environment.
    - Apply statistical methods to data analysis.
    - Resolve the problem that led to conducting the experiment.
-

**Description.** When two or more design alternatives exist, either of which may appear possible, a *classic experiment* may be developed to compare them directly. Two or more prototypes are constructed, identical in all aspects except for the design issue (type of control, wording of an instruction, and so forth). Speed and accuracy measures are collected and user preferences solicited.

**Advantages.** Objective measures of performance and subjective measures of user satisfaction are obtained, thereby permitting a better-informed selection of the best alternative.

**Disadvantages.** A rigorously controlled experiment, and experimental environment, is required to conduct the evaluation. The experiment conductor must have expertise in setting up, running, and analyzing the data collected. Multiple prototypes reflecting the design issues must also be created.

**Guidelines.** General steps to be followed in conducting an experiment are the following. First, state a clear and testable hypothesis (screen background color will affect screen readability). Specify a small number of independent variables to be manipulated (five screen background colors). Carefully choose the measurements (screen reading time and reading errors). Judiciously select study participants and carefully or randomly assign them to groups (ascertain reading speeds and equalize mean speeds for groups). Control for biasing factors (ensure that the same monitor is used for all experimental sessions). Collect the data in a controlled environment (within a usability laboratory). Apply statistical methods to data analysis, and resolve the problem that led to conducting the experiment (choose the best screen background color for the system).

## Focus Groups

---

- **Description:**
    - A discussion with users about interface design prototypes or tasks.
  - **Advantages:**
    - Useful for
      - Obtaining initial user thoughts.
      - Trying out ideas.
    - Easy to set up and run.
    - Low cost.
  - **Disadvantages:**
    - Requires experienced moderator.
    - Not useful for establishing
      - How people really work.
      - What kinds of usability problems people have.
  - **Guidelines:**
    - Restrict group size to 8 to 12.
    - Limit to 90 to 120 minutes in length.
    - Record session for later detailed analysis.
-

**Description.** In a *focus group*, a small group of knowledgeable users and a moderator are brought together to discuss an interface design prototype or proposed design tasks. The discussion is loosely structured, but must be focused on a specific topic or topics.

**Advantages.** A focus group is a useful method for obtaining initial thoughts or trying out ideas. They are easy to set up and run and have a fairly low cost.

**Disadvantages.** Focus groups do require an experienced moderator. Also, they are not useful for establishing how people really work and what kinds of usability problems people have.

**Guidelines.** A focus group should be restricted in size to 8 to 12 people. An individual session should consume no more than 90 to 120 minutes. Recording of the session, either by video or audio, will permit later detailed analysis of participant's comments. The recording can also be played for the entire design team again, providing insights into user needs for all developers.

## Choosing a Testing Method

Unfortunately, there is little published detailed advice on which tests to use, when to use them, and which tests work best together. Beer, Anodenko, and Sears (1997) suggest a good pairing is cognitive walk-throughs followed by think-aloud evaluations. Using cognitive walk-throughs early in the development process permits the identification and correction of the most serious problems. Later, when a functioning prototype is available, the remaining problems can be identified using a think-aloud evaluation.

### ***Usability Test versus Heuristic Evaluation***

As previously described, heuristic evaluations are useful in identifying many usability problems and should be part of the testing arsenal. Performing this kind of evaluation before beginning an actual usability test with users will eliminate a number of design problems. Straub (2003c) has summarized the results of an interesting study contrasting a usability test with a heuristic review that confirms this recommendation. This study, done by Fu et al. (2002), addresses three levels of human behavior identified by Rasmussen (1986) as leading to interface usability challenges: skill-based, rule-based, and knowledge-based.

The *skill-based* level of success depends upon the users' ability to recognize and quickly react to the right signals presented by the interface. Noncritical elements such as flashing or animation can grab attention at the wrong time and pull the user's focus from the task at hand. The result may be errors or degraded speed.

The *rule-based* level of success depends upon the users' ability to perceive and respond to the signals that are associated with the ongoing task. People stumble when they fail to complete a step or steps because the system's state or next-step information was not noticeable or clearly presented.

The *knowledge-based* level of success depends upon the users' ability to develop the right mental model for the system. User actions based upon an incorrect or incomplete mental model may cause the wrong action, resulting in an error.

In the Fu et al. study, 39 distinct usability problems were identified. The heuristic evaluation identified 87 percent and the usability test identified 54 percent. There was a 41 percent overlap in the problems identified. When the problems/errors were categorized on Rasmussen's behavior levels, heuristic review and usability testing identified complimentary sets of problems. The heuristic review identified significant more skill- and rule-based problems, whereas usability testing identified more challenges at the knowledge-based level. Essentially, heuristic reviews exploit our understanding of human perception and cognition. The design or presentation details that may impede or facilitate a person's progress in performing a task are most easily identified. A usability test identifies the gaps that may exist between a system's conceptual model and the user's mental model in the context of how the system is being used. Table 14.5, derived from Straub (2003c), summarizes some basic differences of the two approaches.

Fu et al. concluded that the most effective technique is to use both techniques at different times in the design process. Heuristic reviews should be applied first or early in the design process to identify the skill- or rule-based problems. After these problems are resolved, usability testing can be focused on the higher-level knowledge-based interaction issues without the distractions of the skill- or rule-based problems. Most usability experts now recommend that these two methods always be included in the testing process.

### ***Concurrent versus Retrospective User Comments***

There are two ways for test participants to verbalize their observations and problems to testers: *concurrent*, or as they happen, and *retrospective*, or after the test session is over. Bailey (2003c) has reviewed the research addressing these alternatives. In the retrospective condition, participants are shown a video of the testing session to remind them of the issues they may have confronted.

Concurrent reporting may have two problems. First, it forces people to closely examine a certain portion of the interface, which could impact their performance on the following tasks. Second, immediate reporting may contaminate performance data, like time to complete a task. Wright and Converse (1992) addressed this issue and found verbalizing observations and problems as they happened (as opposed to not doing so) did not impact performance time or number of errors.

Three studies (Bowers and Snyder, 1990; Ohnemus and Biers, 1993; Capra, 2002) found no performance and preference differences between these methods. However, researchers performing the first two studies concluded that the retrospective comments were more useful because they were more explanatory than the concurrent comments, and Capra reported that retrospective comments in general were positively biased, suggesting people were voicing their comments after successfully solving a difficult task, not while struggling to solve a problem. A fourth study (Page and Rahimi, 1995) detected a difference in the kinds of problems reported using the two methods. Concurrent reporting was most valuable for simple issues such as link naming, instruction wording, and error messages. Retrospective reporting seemed best for better understanding more complex problems. Users were unable to articulate why they were having problems while trying to resolve them.

Bailey's conclusions from these studies are as follows:

- There are no differences in the majority of comments made concurrently and those made retrospectively.
- Collecting concurrent comments does not appear to slow down test participants, or cause them to make more errors.
- Valid comments can be collected up to 24 hours after the test. (Ohnemus and Biers).
- Participants tend to comment on their successes and not their struggles (positively biased).
- Concurrent comments are better for certain design problems, such as link naming.
- Retrospective comments are more valuable for helping to resolve complex usability issues.

In conclusion, each testing method described in the preceding pages has strengths and weaknesses. A well-rounded testing program will use a combination of some, or all, of these methods to guarantee the usability of its created product. It is very important that testing start as early as possible in the design process and continue through all developmental stages.

**Table 14.5:** Heuristic Evaluation versus Usability Testing

	HEURISTIC EVALUATION	USABILITY TEST
Addresses this question	Is the design optimized based upon what is known about how people interact with computers?	Can key users - Find the information? - Complete the transaction?
Recommendations derived from	- Current perception and cognition research. - Standards, guidelines, best practices, and experience.	- Direct observation of users doing tasks using system. - Analysis of gap between users' mental model and system conceptual model.
Complimentary benefits	Focuses on what the design brings to the user.	Focuses on what users bring to the design.
Benefits	- Rapid results. - Tactical recommendations. - Comprehensive evaluation.	- Synthesizes recommendations across the task experience. - Contextualizes recommendations to the specific objectives of the system and the limitations of the users.

Derived from Straub (2003c).



## Automated Evaluation Methods

A substantial leap forward in the testing process has been the creation of software tools simulating the behavior of people. These automatic evaluation methods allow usability tests to be performed without requiring real users to perform some of the necessary tasks. An automated evaluation tool can help locate certain kinds of design problems such as missing links, slow loading pages, use of jargon, potential accessibility problems, and so forth. Many commercial automatic evaluation tools are available. Although these tools are useful in uncovering obvious problems, they cannot detect conceptual problems, so they do not eliminate the need for testing using typical users.

## Developing and Conducting a Test

A test of usability requires developing a test plan, selecting test participants, conducting the test, and analyzing the test results.

### The Test Plan

- Define the scope of the test.
- Define the purpose of the test.
  - Performance goals.
  - What the test is intended to accomplish.
- Create a test timetable.
- Define the test methodology.
  - Type of test to be performed.
  - Test limitations.
  - Developer participants.
- Develop scenarios to satisfy the test's purpose.
- Select test participants.
- Identify and schedule the test facility or location.
- Run a pilot test.

An inadequately planned test can waste time and money, and provide flawed or misleading information.

**Scope.** How extensive and detailed will the test be? A test's scope will be influenced by a variety of factors. Determinants include the following issues: (1) The *design stage*: early, middle, or late — the stage of design influences the kinds of prototypes that may exist for the test; (2) the *time available* for the test — this may range from just a few days to a year or more; (3) *finances allocated* for testing — money allocated may range from one percent of a project's cost to more than 10 percent; (4) the project's *novelty* (well-defined or exploratory) — this will influence the kinds of tests feasible to conduct; (5) expected *user numbers* (few or many) and

*interface criticality* (life-critical medical system or informational exhibit) — much more testing depth and length will be needed for systems with greater human impact; (6) the availability of equipment that must be used, and (7) finally, the development team's availability, *experience*, and testing knowledge will also affect the kinds of tests that can be conducted.

**Purpose.** Define the purpose of the test in simple statements, including performance goals, both qualitative and quantitative. To aid assessment of test results, usability measures (or metrics) can be assigned levels; current, best case, planned, and worst case (Whiteside et al., 1988). *Current* is the level of performance of a task for the current system (time to complete, number of errors, and so on.). It provides the benchmark against which the new system will be compared. *Best case* is the performance level that the new system ultimately wants to achieve. *Planned* is the level of performance the current test is expected to achieve. Earliest versions of a system meeting the planned performance level indicate it is safe to move on to the next level of design. *Worst case* is the lowest acceptable level of performance, not the worst that can happen. Failing to achieve the worst-case performance level indicates the new design is not acceptable, and further design (or redesign) is necessary. Another evaluation must be performed to verify the planned (or best level) of performance is achieved.

What the test is expected to accomplish in system learning, use of screens, system navigation, and efficiency of operation must also be defined. Learning issues will center on effectiveness in starting to use the system, recognizing system features, and exploring system features. Screen issues will be directed toward general screen layout, including efficiency and aesthetics, and recognition and understanding of screen-based controls, icons, and messages. Navigational issues involve use of system navigation methods, including menus, buttons, and icons. Operational issues include how many steps are needed to accomplish a task, system responsiveness, and forms of system feedback provided.

**Timetable.** Create a test timetable to keep track of what must be done early in the planning process. Coordination with developers may be needed and a facility reserved. Specific equipment may also have to be reserved. Participants will have to be recruited and scheduled in advance of the test. The timetable can always be adjusted as the planning process moves forward.

**Methodology.** Define the specific *type* of test to be carried out. Also identify any test limitations. Set reasonable expectations. If the test is measuring only a part of a system or its behavior, the results must be interpreted with this limitation in mind. Identify all *participants* from the development team.

**MYTH** Testing would be nice, but it is costly and we don't have time for it.

**Scenarios.** Task scenarios to adequately satisfy the test's purpose must be identified and constructed. Ideally, the entire system will be tested, but time and costs often limit what can actually be done. When time or cost constraints exist, good candidates for testing include the user's most important tasks or the most representative tasks. Always test the functions or features whose design foundation is not as solid as desired. These are features where the trade-off issues were not clear-cut,

not strongly pointing to one design alternative over other possible alternatives. Also, consider testing tasks that have some new design features or functionality added.

After preparing task scenarios, try them out and refine them as necessary. Make sure they are clearly written and capable of being performed in the allocated testing time.

The testing of Web sites will follow the same methodology as that for graphical systems. Web sites, however, present some unique issues that must also be addressed. Many of these important issues are summarized in Table 14.6. This listing is not intended to be all-inclusive. It is simply presented as a reminder of “what not to forget to do.”

**Table 14.6:** Some Things to Remember to Test in Web Site Design

TEST:	ENSURE THAT:
All the browsers, servers, and monitors used.	Different servers and browsers don't introduce adverse new behaviors. Different monitors don't negatively affect color appearance, legibility, and page size.
Different dial-up speeds.	Download times are satisfactory for all users.
The navigation design.	Users know how to find information. Navigation sequence through related information makes sense. Users know where they are in the Web site's structure. Users know how to return to visited points when they are browsing. All links are working properly. Unnecessary links are found and removed.
The visual design style.	The style reflects the site's purpose. The style creates a positive and proper impression of the organization owning the site. The style is compatible with user tasks and needs.
Content legibility and readability.	The design succeeds for all default fonts. All grammar and spelling is correct.
Backgrounds and color.	Backgrounds are compatible with foregrounds. Backgrounds are not distracting. Monitor color rendering differences do not negatively affect site usability.
Graphics and icons.	Graphics are meaningful and efficient. Graphics are not distracting to the user.
Page breaks.	Actual page breaks are obvious. False signals for page breaks do not exist.
Page printing.	The page prints completely and does not bleed off the page.
Accessibility.	Design is compatible with accessibility utilities. Design is compatible with accessibility guidelines.

## ***Test Participants***

---

- Assemble the proper people to participate in the test.
  - Consider allowing users to work in pairs.
  - Select the proper number of participants.
  - Consider providing compensation or incentives.
- 

**Proper people.** Assembling the proper people to participate in the test is critical to its success. Always recruit participants with the proper qualifications, normally those currently performing the job or task where the product will ultimately be used. While the “boss” may profess to be knowledgeable, he or she is usually too far removed from the grind of daily activities and seldom knows exactly what is going on. Also, recruit participants covering the spectrum of user characteristics including age, gender, skills, and experience to allow general conclusions based on the test results. Recruit strangers, not friends. A stranger is less likely to withhold comments that might possibly hurt the tester’s feelings. There will also be less embarrassment if problems are uncovered.

If the participants must fit into a particular user profile, or profiles, then it may be necessary to create a recruitment screener to assess whether or not the person will be suitable. The screener may consist of a questionnaire to be filled out by the prospective participant or, if an interview is conducted, a list of relevant questions to be asked, including gender, age or age range, education attainment, ethnicity, computer and Internet experience, and Web activities. Disabilities, if any, must also be ascertained.

Plan well ahead in assembling test participants. Recruitment two or more weeks ahead of the test will make it likelier people will be able to fit a test session into their schedule.

**Working in pairs.** User observation customarily consists of a single user working alone. Stone et al. (2005) suggests that in some situations recruiting people to work in pairs should be considered. These situations include the following:

- People work cooperatively and share a computer.
- Cultural constraints may make it difficult for people to be critical of an interface to someone in authority. For some people it is easier to discuss an interface with a peer than an evaluator. Snyder (2003) routinely does this with participants because she has found that participants find it easier to chat to each other than to a test facilitator.
- People may work, or prefer to work, in pairs in a collaborative mode in doing their job.

In some cases it may be necessary to include a helper or user advocate. For example,

- Children may require a parent, teacher, or other responsible adult who is known to be present.
- For non-English speakers an interpreter will be needed. Even if the evaluator speaks the participant’s language (as a second language), nuances in the participant’s reactions may be missed and bias the test results.

- Participants with speech impairments or learning or cognitive disabilities that affect speech or understanding may have difficulty communicating. They may not always understand what they are being asked to do. A helper may have to be recruited or the participant asked to bring a helper or interpreter.

When working with an interpreter, advocate, or helper, always address remarks to the participant, not to the intermediary.

**Number of participants.** How many test participants are enough? Using too few people in a test may result in usability problems going undetected. Using too many may waste valuable resources. Research and discussion concerning this topic has raged for years. Straub (2004b) provided a review of the issues upon which the following discussion is partially based.

Virzi (1992) provided one of the earliest recommendations. Applying the law of diminishing returns, he reported that five users would uncover approximately 80 percent of the usability problems in a product. Virzi's recommendation was followed shortly by an identical recommendation by Nielsen (Landauer and Nielsen, 1993; Nielsen, 1993; Nielsen, 2000).

Nielsen (2000), in an analysis of thirteen studies, determined that about 85 percent of a system's usability problems will be identified with five participants, and the law of diminishing returns sets in at this point. It is much better, he reasons, to conduct more types of tests with fewer participants, than to conduct fewer tests with more participants.

Later studies did not find the same result. One study reported that testing five participants revealed only 35 percent of the problems uncovered by a larger number of participants (Spool and Schroeder, 2001), and in another test using 18 people, participant numbers 6 to 18 each identified five or more problems that were not uncovered by participants numbers 1 through 5 (Perfetti and Landesman, 2002).

Faulkner (2003) took a different approach. She evaluated the usability of a Web-based application using 60 participants. Then, using a sampling algorithm, she ran test sizes using 5, 10, 20, 30, 40, 50, and 60 users. On average, she confirmed Nielsen's prediction. Over 100 simulated tests with a sample size of five users identified an average of 85 percent of the usability problems identified by the larger group. She also calculated another figure — the range of the number of usability problems found by people in each test-participant group size. The important results are shown in Table 14.7.

**Table 14.7:** Percentage of Problems Detected by Test Group Size

TEST GROUP SIZE	AVERAGE PERCENTAGE OF PROBLEMS FOUND	LOWEST PERCENTAGE OF PROBLEMS FOUND
5	85%	55%
10	95%	82%
15	97%	90%

Faulkner's data confirmed Nielsen's prediction that 85 percent of usability problems will be found using five participants. However, she also found that increasing the number of participants in a test group certainly appears to increase the likelihood of finding more design problems. Consequently, the research results indicate that for routine usability tests conducted early in the development process, five participants may be enough. For important tests conducted later in development, a larger number of people will be required. In other words, start with a small number of participants and increase testing sample sizes as iterative testing continues. The more mature a product becomes, the larger sample sizes will be needed to discover the diminishing number of remaining problems.

One caveat — if a system such as a Web site has several highly distinct groups of users (children, parents, and senior citizens, for example) then their interface behaviors may differ. In this case, each different group must be tested. Nielsen recommends using three to four users from each category if one is testing two groups of users, and three users from each category if one is testing three or more user groups. Other experts recommend, however, using a larger number of participants in each category. Also, do not forget to include users with disabilities.

One problem that has to be considered in selecting test participants is the “no-show” rate. Selected participants are subject to all the daily problems encountered in life; illness, an unforeseen personal event, weather, traffic, and so forth. Nielsen (2003), in a survey of a large number of usability professionals, reported that the participant “no-show” rate was 11 percent. This means, on the average, one out of nine users fail to show up as promised. While no-show rates are variable and not predictable for any single test, it is a good idea to recruit an additional person or two more than deemed minimally necessary to conduct a test.

**Compensation and incentives.** Some form of thank you is usually in order for test participants. For company employees recruited for an informal test, a verbal thank you may be all that is necessary. For more formal evaluations conducted with the approval of the participant's manager, a thank-you letter to the employee and the manager may be desirable. The majority of companies do not offer extra compensation for test participation, but about one-third in Nielsen's survey provided a small gift as a token of appreciation. In contrast, companies typically provide monetary compensation to people recruited from outside the organization. Finally, always advise selected test participants of what to expect in the test. They will approach the test with less apprehension. Finally, never refer to test participants as “subjects.” While many of us, including myself, have been conditioned to use this term, and have used it for very many years, the correct term today is “evaluator” or “participant.”

### ***Test Facility or Location***

---

- Select the test location
    - In a formal environment — a usability laboratory.
    - In an informal environment — an office or conference room.
    - In the field at the actual work location.
    - Unmoderated remote testing.
-

A usability test can be conducted in a variety of places ranging from a formal usability laboratory to the participant's actual work location. Tests that are undertaken in the user's own environment are referred to as *field studies*. Evaluations undertaken somewhere else are called *controlled studies*. Controlled studies may be *formal*, being performed in a specially-designed laboratory, or *informal*, occurring in a conference room or office.

**Usability laboratory.** A laboratory is a fixed specially constructed facility designed and fitted for conducting formal, controlled tests. The advantages of a laboratory from a data-collection perspective are

- It can be designed with one-way mirrors for observers and note takers. More people are able to participate in, or simply watch, the test. The importance of usability testing can actually be demonstrated to other visiting personnel of the organization whose system is being tested.
- It is easily equipped with recording devices such as video cameras and video monitors or projectors.
- The physical environment — lighting, temperature, and so on — can be more precisely controlled. Actually work environmental conditions can be simulated.
- Distractions and disturbances can be minimized.

If the test is being held in a usability laboratory, the test facility should resemble the location where the system will be used. The biggest disadvantage of a laboratory is its construction cost. Usability labs may be available for rent, however. Its fixed location may not be convenient for all participants, though.

**Office or conference room.** Informal controlled tests can be held in an office, conference room, or a hotel meeting room. To collect test data, laboratories are now available in portable units that can be easily shipped and set up at remote facilities. Portable labs now possess most of the components incorporated within specially constructed labs. The benefits of a portable lab, according to Koyani (2006), are as follows:

- Scheduling tests is easier. Meeting rooms are easier to locate and reserve.
- More flexibility exists in establishing a test location. The test can be held at locations close to the user's workplace.
- Renting a formal laboratory usually costs more than to rent of portable lab.
- An informal setting is satisfactory for collecting qualitative data.
- Some usability specialists consider a fixed laboratory an artificial environment and prefer a more natural environment.

**Field studies.** An evaluation may also be conducted at the participant's workplace. An advantage is that information about the environment, including distractions and interruptions can be collected. Disadvantages include being difficult to arrange and set up; test disruptions because of the participant being called away to address other issues or problems; distractions for adjacent workers, management objections; and the workplace being unsuitable for visitors.

**Unmoderated remote testing.** Recently there has been an interest in another form of testing called unmoderated remote testing. This method asks participants to complete prescribed tasks remotely, either at home or in another non-laboratory site, while performance measures are collected. An evaluator, or observer, does not physically monitor task performance. Schulte-Mecklenbeck and Huber (2003) compared Web-based opened-ended tasks performed in a lab versus at home by different groups of users. Lab users took twice as much time and used twice as many clicks as the home users. The authors postulated that lab users may have felt more pressure to perform well and/or perceived the lab evaluation as “more important.”

Tullis et al. (2002) performed a similar study using “closed-ended” tasks (the results being either right or wrong). No differences between the success rates and completion times were found between the two groups. They concluded that remote testing worked well and had benefits that complimented the lab study. They recommended a combination of both remote and lab testing to cover the range of design issues. Remote testing deserves additional study.

### ***Pilot Test***

---

- Choose a participant.
  - Design and assemble the test environment.
  - Run the pilot test.
- 

A pilot test evaluates the planned testing session. It is a practice session that tries out the evaluation materials and equipment, the planned time schedule, and the overall running of the session. It helps ensure that all the materials are properly in place and the test will work as planned. Without a pilot test, the first participant in the actual test becomes the pilot instead. Then, changes in materials and procedures must be made quickly. Stone et al. (2005) suggests a pilot test should encompass the following activities.

**Participants.** A participant is chosen in the normal way. It is less important that participant be representative of the planned user group, more important that it be someone who can be confidently worked with.

**Test environment.** Conduct the pilot test in the same location of the actual test, or in as similar a place as possible. Materials needed include computer or paper prototype, other necessary video or recording equipment, evaluation materials, and any other required tools such as paper and pencils. Practice assembling all needed equipment. Also, develop a checklist of all items and include the incentives if they are being offered.

**Running the test.** Conduct the test in the same way the actual evaluation will be run. All evaluators who will conduct the actual test should attend. They should observe, take notes, and facilitate the user just as will be done in the actual test. Ensure that all equipment is working properly, materials are clear, data collection procedures are working, and the test time schedule is adhered to. Note detected



problems that must be corrected. Analyze and interpret the data to see if any important test facets have been overlooked. Try and hold the pilot test before finalizing recruitment for the actual test. This will allow time for any necessary test refinements to be made.

## Test Conduct and Data Collection

To collect usable data, the test should begin only after the proper preparation. Then, the data must be properly and accurately recorded. Finally, the test must be concluded and followed up properly. Following are guidelines for conducting a usability test. Many are from Schrier (1992).

### *Usability Test Guidelines*

- 
- Before starting the test,
    - Ensure that all necessary test materials and equipment are available.
    - Explain that the objective is to test the software, not the participants.
    - Explain how the test materials and records will be used.
    - If a consent agreement is to be signed, explain all information on it.
    - If verbal protocols will be collected, let participants practice thinking aloud.
    - Ensure that all participants' questions are answered and that participants are comfortable with all procedures.
  - During the test,
    - Minimize the number of people who will interact with the participants.
    - If observers will be in the room, limit them to two or three.
    - Provide a checklist for recording
      - Times to perform tasks.
      - Errors made in performing tasks.
      - Unexpected user actions.
      - System features used/not used.
      - Difficult/easy-to-use features.
      - System bugs or failures.
    - Record techniques and search patterns that participants employ when attempting to work through a difficulty.
    - If participants are thinking aloud, record assumptions and inferences being made.
    - Record the session with a tape recorder or video camera.
    - Do not interrupt participants unless absolutely necessary.
    - If participants need help, provide some response.
      - Provide encouragement or hints.
      - Give general hints before specific hints.
      - Record the number of hints given.
    - Watch carefully for signs of stress in participants:
      - Sitting for long times doing nothing.
      - Blaming themselves for problems.
      - Flipping through documentation without really reading it.

- Provide short breaks when needed.
  - Maintain a positive attitude, no matter what happens.
  - After the test,
    - Hold a post-session interview with participants.
    - Consider using a questionnaire to collect interview data.
    - Tell participants what has been learned in the test.
    - Provide a follow-up questionnaire that asks participants to evaluate the product or tasks performed, if necessary.
    - If videotaping, use tapes only in proper ways.
      - Respect participants' privacy.
      - Get written permission to use tapes.
- 

### **Before Starting the Test**

First, ensure that all the necessary test materials are available and all test equipment is running properly. Then, brief all participants before conducting the test. Most people participating in a test will approach it with some anxiety. Fears may exist that they themselves are being judged, or apprehension may be caused by a general fear of the unknown, a common human trait. These fears must be put to rest. Before the test begins, all participants must be told exactly what will happen in the test. Explain that the test objective is to evaluate the software, not the participants themselves. Also explain how the test materials and records will be used. If participants will be signing a consent agreement, review and explain all information on it before it is signed. If verbal protocols will be collected, that is, if the participants are going to be asked to think aloud, let participants practice this process. For most people this is not a common experience, and it may require getting used to. Do not start the test session until all participants' questions are answered and people are comfortable with all of the test procedures. Providing this kind of information, and preparation, will enable participants to relax faster at the start of the test.

Finally, the participant's health should be considered. Be cautious in using test participants who have a cold. With a cold, people will respond more slowly, detect fewer problems, and have reduced energy and motivation levels (Matthews et al., 2001). The same problems also exist for evaluators themselves, of course.

### **During the Test**

Minimize the number of people who will interact with participants. Many and strange voices must be avoided because they can be very distracting and disturbing. If observers will be in the room during the test, limit their number to two or three. Observers must never talk during the test.

For data recording, provide observers with a checklist reminding them what to record and for use in actually recording data. Useful information to collect includes the time needed to perform each of the test tasks, errors made, any unexpected actions taken by the participants, how often system features are used, those features that are not used, difficulties in using features, features that are particularly easy to use, and system bugs or system failures. When participants encounter a difficulty, record the

techniques and search patterns they employ when attempting to work through the difficulty. Eye-tracking equipment can be useful for recording exactly where the user is looking. This equipment has the ability to discriminate between glances and longer eye fixations. If participants are thinking aloud, record the assumptions and inferences they make as they proceed. If practical, record the test with a tape recorder or video camera. This will permit more leisurely review of the test session later. Details missed during the session will be uncovered, and comparisons can be made between the approaches and activities of the different participants. The entire design team will also be allowed to later review and evaluate the test results.

Use automated data collection methods whenever possible. Automated collection testing allows more rapid data collection and analysis, the use of a larger numbers of users, and faster analysis and evaluation. Never interrupt a test participant unless it is absolutely necessary. If, however, it is sensed that participants need help, provide a response of some kind, first through encouragement and then through hints. Provide general hints before specific hints, and record the number of hints given. Watch carefully for signs of stress. If it is detected, again give encouragement or provide hints, or provide a break. Signs of stress include a participant's sitting for a long time doing nothing, blaming himself or herself for problems, and flipping through documentation without really reading it. Make sure participants know to ask for a short break when needed. Remember, the participant is more important than the test. The test can always be performed at another time with another person. Finally, maintain a positive attitude no matter what happens (everything probably will). A tester with a negative attitude will influence the participants in the same way, and the data collected will be contaminated.

**MAXIM** Experienced designers have the wisdom and humility to understand that extensive usability testing is a necessity.

### After the Test

At the test's conclusion, hold a closing interview or debriefing with the participants. During this interview, questions that occurred to the tester during the actual test can be asked. The participants can also ask questions, and provide explanations about why certain actions were taken. If a video recording of the session was made, the recording can be viewed to stimulate questions and comments.

To structure the interview, consider using a questionnaire. A questionnaire will ensure that important questions are not forgotten. Answers from different people can also be compared and some quantitative data compiled. ("Four of five people said this aspect of the design was poor.") Several questionnaires are available for use in usability testing. Two of the best known are the *Website Analysis and MeasureMent Inventory* (WAMMI) and the *Software Usability Measurement Inventory* (SUMI). Both have been thoroughly validated.

Tullis and Stetson (2004) compared five available questionnaires (not including WAMMI and SUMI) and concluded the *System Usability Scale* (Brooke, 1996) gave the most reliable results. If a commercially-available questionnaire is used, consider whether it has to be modified to reflect the system being tested and to be meaningful to the users. Much more information on questionnaire design can be found in Barnett

(2005). If a questionnaire is used, it should be used as a basis for part of the interview, so an opportunity exists to explore the reasons why the participant answered as was done.

Tell the participants some of what has been learned. This will make the participants feel that their effort was worthwhile. Also, if necessary, provide follow-up questionnaires that ask participants to evaluate the product or tasks performed. If videotaping is performed, respect the participant's privacy when the tape is later shown. If necessary, obtain the participant's written permission to later use the tape.

## Analyze, Modify, and Retest

---

- Compile the data from all test participants.
  - List the problems the participants had.
  - Sort the problems by priority and frequency.
  - Develop solutions for the problems.
  - Modify the prototype as necessary.
  - Test the system again, and again.
- 

**Data analysis.** Upon completion of the test the data collected will have to be collated, summarized, and reviewed to identify any usability problems. *Collation* involves organizing in a meaningful way the large amount of data on paper that has been collected. Common techniques used by practitioners include entering manually collected information on paper forms, a word processor, a spreadsheet, or simple index cards. Printouts of computer-collected data will also have to be collated. Collated data whose source is a test participant should be traceable back to that participant. Evaluator notes and comments should also be identifiable as such. Typically, the collation process is performed before the analysis process begins. By doing this, it will be easier not to get sidetracked by specific comments and ensure that all relevant data is available before recommendations are generated.

*Summarization* involves extracting and compiling key data from that which has been collated. It may also involve performing simple mathematical calculations to compute averages, times, and ranges.

*Reviewing* the summarized data is accomplished to see if the usability requirements have been met. If this review shows that the established usability requirement has been met, then no further analysis is required. If, however, the analysis reveals problems such as too many errors, inability to complete a task, long task completion times, or violations of design standards or guidelines, then a usability problem (or defect) exists. Usability problems often lead to user confusion, errors, delays, or failures to accomplish something. List the problems the participants had, and sort the problems by priority and frequency. See Table 14.2 for a problem-rating scheme. Make the results available to the entire design team for analysis, again to provide multiple insights into problem solutions. Then, develop solutions for the problems. Get expert advice if the solutions are not obvious.

Another word of caution is in order, however. Severity ratings of usability problems by experts still remain more of an art than a science. Several studies (Catani and Beers, 1998; Jacobsen, et al. 1998; Cockton and Woolrych, 2001; Hertzum and Jacobson, 2001; Law and Hvannberg, 2004; Molich, 2005) report that agreement among various usability experts concerning the number of usability problems in a system, or the relative severity of these problems, is seldom consistent. Therefore, developers should address *all* usability issues that have been detected in every test.

**Prototype modification.** Prototypes must, of course, be modified based on the design recommendations made during testing, and the solutions decided upon. Unfortunately, new usability problems may inadvertently be created when fixing problems identified by previous usability testing, so testing must continue until all defects are uncovered and eliminated.

**Test again.** The testing process continues in an iterative manner until all problems are satisfactorily solved and all criteria are met. After the prototyping is complete and all code written, a final system test must be performed to ensure that no software bugs exist, and that performance meets all specifications. The screens and interface must also be again tested to ensure that all established usability criteria are being met. The design steps and methods are identical to those for prototype testing.

## Evaluate the Working System

- 
- Collect information on actual system usage through
    - Interviews and focus group discussions.
    - Surveys.
    - Support line.
    - Online suggestion box or trouble reporting.
    - Online bulletin board.
    - User newsletters and conferences.
    - User performance data logging.
  - Respond to users who provide feedback.
- 

Testing never stops with system implementation. The interface, like any part of a system, must be continually evaluated to ensure that it is achieving its design objectives. Problems detected can be corrected in system enhancements and new releases. This type of evaluation is necessary for a variety of reasons.

**Problems will have slipped through the cracks.** In spite of all the pre-implementation testing performed, problems will still exist. It is impossible to exercise all aspects of a system in a testing environment as thoroughly as is done in actual system use. Also, actual use can capture the experiences of all users, not simply those used in the various testing phases.

**Initially impressive features may later be regarded as frustrating or completely ignored.** Initial impressions may change over time. Some parts of a system, which in the testing process were seen as neat or helpful, may, in everyday use, be found to have little or no value.

**Experienced users are more sensitive to time delays.** Response times that seemed adequate in the testing process may, as users become experienced, become a source of great irritation.

**Customizable interfaces may change.** Over a period of time, customization may change the interface so much that original features are lost and tasks become more difficult to perform.

**The external environment may have changed.** The original system hardware used in the testing process may change over time. Added elements or features may affect the system's usability in a negative way, or allow enhancement of the interface. Many of the techniques used in requirements determination and prototype evaluation can also be applied in this evaluation phase.

Evaluation information can be collected in the following ways:

**Interviews and focus group discussions.** Individual user interviews can identify specific areas of concern. These can be pursued to the level of detail needed. Focus groups can then be conducted to determine the generality of the identified problems.

**Surveys.** A survey or questionnaire is administered to users to obtain feedback on usability problems. These surveys can use e-mail, the Web, or traditional mail.

**Support line.** Information collected by the unit that helps customers with day-to-day problems can be analyzed (Customer Support, Technical Support, Help Desk, and so on).

**Online suggestion box or trouble reporting.** An online suggestion box can be implemented to solicit comments or enhancement suggestions.

**Online bulletin board.** An electronic bulletin board can be implemented to solicit comments, questions, or enhancement suggestions.

**User newsletters and conferences.** Improvements can be solicited from customer groups who publish newsletters or convene periodically to discuss software usage.

**User performance data logging.** The system software can capture users' usage patterns and problems doing real tasks. When one uses log data, however, a user's goals and needs can only be surmised.

**Respond to users who provide feedback.** When feedback is received, respond in an appreciative manner to the person providing it. A positive relationship will be established, and the person will be more likely to provide additional feedback. A failure to respond will discourage future suggestions and indicate to the user that his or her needs are not important. Finally, if a system change is made based upon a suggestion, inform the user or users making the suggestion of the change.

## Additional Reading

---

For the reader who needs much-more-detailed information on usability testing, the following are recommended. For a demonstration of the importance of usability testing to the success of a software development project, see Bias and Mayhew's *Cost-Justifying Usability* (1994). For a step-by-step guide, including checklists and insights, see Dumas and Redish's *A Practical Guide to Usability Testing* (1993). For a practical handbook, see Nielsen's *Usability Engineering* (1993). For discussions of usability testing methods, see Nielsen and Mack's *Usability Inspection Methods* (1994). Extensive accessibility testing guidelines will be found in the Web sites of IBM and Microsoft.

For useful additional information on user interface design the following texts are also recommended:

Fowler, S. L., and Stanwick, V. R. (2004). *Web Application Design Handbook*. San Francisco, CA, Morgan Kaufman Publishers.

Koyani, S. J., Bailey, R. W., and Nall, J. R. (2004). *Research-Based Web Design and Usability Guidelines*. National Cancer Institute.

Stone, D., Jarrett, C., Woodroffe, M. and Minocha, S. (2005). *User Interface Evaluation and Design*. San Francisco, CA, Morgan Kaufmann.

For an excellent compilation of universal design principles see

Lidwell, W., Holden, K. and Butler, J. (2003). *Universal Principles of Design*, Gloucester, MA, Rockport Publishers.

For excellent information on designing paper forms and questionnaires that are easy to use see

Barnett, R. (2005). *Forms for People*. Robert Barnett and Associates Pty Ltd, Belconnen, ACT, Australia.

Useful Web sites that provide regular and timely updates on user interface design issues are

Human Factors International's User Interface Design Newsletter at <http://www.humanfactors.com>

Jacob Nielsen's Alertbox at <http://www.useit.com/alertbox>

The US National Cancer Institute's Guidelines Web site at <http://www.usability.gov> and its Usability Information Web site at <http://www.webusability.com>

## **A Final Word**

---

Enjoy your journey through the wonderful world of user interface development. Application of these many principles in design will aid greatly in creating a product that satisfies all your client's needs. A happy and satisfied client, of course, also means a happy and satisfied developer. Good luck!





Future Vision

# FUTURE VISION BIE

By K B Hemanth Raj

**Visit :** <https://hemanthrajhemu.github.io>

## Quick Links for Faster Access.

**CSE 8<sup>th</sup> Semester** - <https://hemanthrajhemu.github.io/CSE8/>

**ISE 8<sup>th</sup> Semester** - <https://hemanthrajhemu.github.io/ISE8/>

**ECE 8<sup>th</sup> Semester** - <https://hemanthrajhemu.github.io/ECE8/>

## 8<sup>th</sup> Semester CSE - TEXTBOOK - NOTES - QP - SCANNER & MORE

**17CS81 IOT** - <https://hemanthrajhemu.github.io/CSE8/17SCHEME/17CS81/>

**17CS82 BDA** - <https://hemanthrajhemu.github.io/CSE8/17SCHEME/17CS82/>

**17CS832 UID** - <https://hemanthrajhemu.github.io/CSE8/17SCHEME/17CS832/>

**17CS834 SMS** - <https://hemanthrajhemu.github.io/CSE8/17SCHEME/17CS834/>

## 8<sup>th</sup> Semester Computer Science & Engineering (CSE)

**8<sup>th</sup> Semester CSE Text Books:** <https://hemanthrajhemu.github.io/CSE8/17SCHEME/Text-Book.html>

**8<sup>th</sup> Semester CSE Notes:** <https://hemanthrajhemu.github.io/CSE8/17SCHEME/Notes.html>

**8<sup>th</sup> Semester CSE Question Paper:** <https://hemanthrajhemu.github.io/CSE8/17SCHEME/Question-Paper.html>

**8<sup>th</sup> Semester CSE Scanner:** <https://hemanthrajhemu.github.io/CSE8/17SCHEME/Scanner.html>

**8<sup>th</sup> Semester CSE Question Bank:** <https://hemanthrajhemu.github.io/CSE8/17SCHEME/Question-Bank.html>

**8<sup>th</sup> Semester CSE Answer Script:** <https://hemanthrajhemu.github.io/CSE8/17SCHEME/Answer-Script.html>

## Contribution Link:

<https://hemanthrajhemu.github.io/Contribution/>

**Stay Connected... get Updated... ask your queries...**

**Join Telegram to get Instant Updates:**

<https://telegram.me/joinchat/AAAAFTtp8kuvCHALxuMaQ>

**Contact: MAIL:** [futurevisionbie@gmail.com](mailto:futurevisionbie@gmail.com)

**INSTAGRAM:** [www.instagram.com/futurevisionbie/](http://www.instagram.com/futurevisionbie/)