PAYRANGE - EXHIBIT 2012
Alliance Laundry v. PayRange
PGR2025-00028

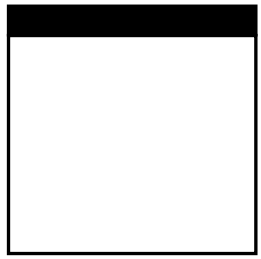
WILEY

Third Edition

The Essential Guide to User Interface Design

An Introduction to GUI Design Principles and Techniques

Wilbert O. Galitz



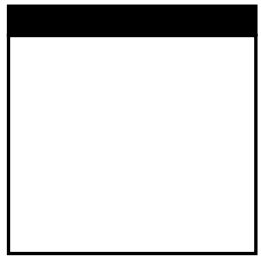
The Essential Guide to User Interface Design

An Introduction to GUI Design Principles and Techniques

Third Edition

Wilbert O. Galitz

Wiley Publishing, Inc.



The Essential Guide to User Interface Design

An Introduction to GUI Design Principles and Techniques

Third Edition

Wilbert O. Galitz

Wiley Publishing, Inc.

The Essential Guide to User Interface Design: An Introduction to GUI Design Principles and Techniques, Third Edition

Published by
Wiley Publishing, Inc.
10475 Crosspoint Boulevard
Indianapolis, IN 46256
www.wiley.com

Copyright © 2007 by Wilbert O. Galitz. All rights reserved.

Published by Wiley Publishing, Inc., Indianapolis, Indiana

Published simultaneously in Canada

ISBN: 978-0-470-05342-3

Manufactured in the United States of America

10 9 8 7 6 5 4 3 2 1

No part of this publication may be reproduced, stored in a retrieval system or transmitted in any form or by any means, electronic, mechanical, photocopying, recording, scanning or otherwise, except as permitted under Sections 107 or 108 of the 1976 United States Copyright Act, without either the prior written permission of the Publisher, or authorization through payment of the appropriate per-copy fee to the Copyright Clearance Center, 222 Rosewood Drive, Danvers, MA 01923, (978) 750-8400, fax (978) 646-8600. Requests to the Publisher for permission should be addressed to the Legal Department, Wiley Publishing, Inc., 10475 Crosspoint Blvd., Indianapolis, IN 46256, (317) 572-3447, fax (317) 572-4355, or online at http://www.wiley.com/go/permissions.

Limit of Liability/Disclaimer of Warranty: The publisher and the author make no representations or warranties with respect to the accuracy or completeness of the contents of this work and specifically disclaim all warranties, including without limitation warranties of fitness for a particular purpose. No warranty may be created or extended by sales or promotional materials. The advice and strategies contained herein may not be suitable for every situation. This work is sold with the understanding that the publisher is not engaged in rendering legal, accounting, or other professional services. If professional assistance is required, the services of a competent professional person should be sought. Neither the publisher nor the author shall be liable for damages arising herefrom. The fact that an organization or Website is referred to in this work as a citation and/or a potential source of further information does not mean that the author or the publisher endorses the information the organization or Website may provide or recommendations it may make. Further, readers should be aware that Internet Websites listed in this work may have changed or disappeared between when this work was written and when it is read.

For general information on our other products and services or to obtain technical support, please contact our Customer Care Department within the U.S. at (800) 762-2974, outside the U.S. at (317) 572-3993 or fax (317) 572-4002.

Library of Congress Cataloging-in-Publication Data

Galitz, Wilbert O.
The essential guide to user interface design : an introduction to GUI design principles and techniques / Wilbert O. Galitz. — 3rd ed.
p. cm.

Includes bibliographical references and index.
ISBN-13: 978-0-470-05342-3 (paper/website)
ISBN-10: 0-470-05342-9 (paper/website)
1. Graphical user interfaces (Computer systems) I. Title.
QA76.9.U83G33 2007
005.4'37—dc22

2006038755

Trademarks: Wiley and the Wiley logo are registered trademarks of John Wiley & Sons, Inc. and/or its affiliates, in the United States and other countries, and may not be used without written permission. All other trademarks are the property of their respective owners. Wiley Publishing, Inc., is not associated with any product or vendor mentioned in this book.

Wiley also publishes its books in a variety of electronic formats. Some content that appears in print may not be available in electronic books.

To my wife and business partner, Sharon, for many years of love and support in our home and office.

To our grandchildren, Mitchell, Barry, Deirdra, and Spencer Galitz, Lauren and Scott Roepel, and Shane and Emily Watters. May one or more of them pick up the writing torch.

About the Author

Wilbert (Bill) O. Galitz is an internationally respected consultant, author, and instructor with a long and illustrious career in Human Factors and user-interface design. For many years he has consulted, lectured, written about, and conducted seminars and workshops on these topics worldwide. He is now the author of eleven books, and his first book, *Human Factors in Office Automation* (1980), was critically acclaimed internationally. This book was the first to address the entire range of human factors issues involved in business information systems. As a result, he was awarded the Administrative Management Society's Olsten Award. Other books have included *User-Interface Screen Design* and *It's Time to Clean Your Windows*. He has long been recognized as a world authority on the topic of screen design.

Bill's career now spans more than 45 years in information systems, and he has been witness to the amazing transformation of technology over this time span. His career began in 1961 with the System Development Corporation, where he was a Training Consultant for the SAGE North American Air Defense System. SAGE was the world's first large-scale display-based system. Before forming his own consulting company in 1981, he worked for CNA Insurance and the Insurance Company of North America (now CIGNA), where he designed the user-interfaces and developed screen and interface design standards for a variety of business information systems. His work experience also includes an appointment at South Africa's National Institute for Personnel Research and a number of years with UNIVAC (now UNISYS). At UNIVAC he performed the human engineering of the company's first commercial display terminal and completed a pioneering study on the operational aspects of large-scale computer systems.

A native of Chicago, Bill possesses a B.A. in Psychology from Lake Forest College in Illinois and an M.S. in Industrial Psychology from Iowa State University. He currently resides in Surprise, Arizona.

Credits

Executive Editor
Robert Elliott

Development Editor
Ed Connor

Technical Editor
Robert Barnett

Copy Editor
Mildred Sanchez

Editorial Manager
Mary Beth Wakefield

Production Manager
Tim Tate

Vice President and Executive Group Publisher
Richard Swadley

Vice President and Executive Publisher
Joseph B. Wikert

Project Coordinator
Erin Smith

Graphics and Production Specialists
Denny Hager
Jennifer Mayberry

Quality Control Technician
Brian H. Walls

Proofreading
Broccoli Information Management

Indexing
Anne Leach

Anniversary Logo Design
Richard Pacifico

Contents

About the Author v

Preface xix

Acknowledgments xxvii

Part 1 The User Interface—An Introduction and Overview 1

Chapter 1 The Importance of the User Interface 3

Defining the User Interface 4
The Importance of Good Design 4
The Benefits of Good Design 5
A Brief History of the Human-Computer Interface 7
Introduction of the Graphical User Interface 7
The Blossoming of the World Wide Web 8
A Brief History of Screen Design 10
What's Next? 12

Chapter 2 Characteristics of Graphical and Web User Interfaces 13

Interaction Styles 13
Command Line 14
Menu Selection 14
Form Fill-in 14
Direct Manipulation 15
Anthropomorphic 15
The Graphical User Interface 16
The Popularity of Graphics 16
The Concept of Direct Manipulation 17
Graphical Systems: Advantages and Disadvantages 19
Characteristics of the Graphical User Interface 24

The Web User Interface 28
The Popularity of the Web 29
Characteristics of a Web Interface 29
The Merging of Graphical Business Systems and the Web 39
Characteristics of an Intranet versus the Internet 39
Extranets 40
Web Page versus Application Design 40
Principles of User Interface Design 44
Principles for the Xerox STAR 44
General Principles 45
Part 1 Exercise 58
What's Next? 58

Part 2 The User Interface Design Process 59
Obstacles and Pitfalls in the Development Path 59
Designing for People: The Seven Commandments 60
Usability 64
Usability Assessment in the Design Process 65
Common Usability Problems 65
Some Practical Measures of Usability 68
Some Objective Measures of Usability 69

Step 1 Know Your User or Client 71
Understanding How People Interact with Computers 71
The Human Action Cycle 72
Why People Have Trouble with Computers 73
Responses to Poor Design 74
People and Their Tasks 76
Important Human Characteristics in Design 76
Perception 76
Memory 78
Sensory Storage 79
Visual Acuity 80
Foveal and Peripheral Vision 81
Information Processing 81
Mental Models 82
Movement Control 83
Learning 83
Skill 84
Performance Load 84
Individual Differences 85
Human Considerations in the Design of Business Systems 87
The User's Knowledge and Experience 87
The User's Tasks and Needs 92
The User's Psychological Characteristics 95
The User's Physical Characteristics 96
Human Interaction Speeds 100

Performance versus Preference 101
Methods for Gaining an Understanding of Users 102
Step 1 Exercise 102

Step 2 Understand the Business Function 103
Business Definition and Requirements Analysis 104
Information Collection Techniques 104
Defining the Domain 112
Considering the Environment 112
Possible Problems in Requirements Collection 113
Determining Basic Business Functions 113
Understanding the User's Work 114
Developing Conceptual Models 115
The User's New Mental Model 120
Design Standards or Style Guides 120
Value of Standards and Guidelines 121
Customized Style Guides 124
Design Support and Implementation 125
System Training and Documentation Needs 125
Training 126
Documentation 126
Step 2 Exercise 126

Step 3 Understand the Principles of Good Interface and Screen Design 127
Human Considerations in Interface and Screen Design 128
How to Discourage the User 128
What Users Want 130
What Users Do 130
Interface Design Goals 131
The Test for a Good Design 132
Screen and Web Page Meaning and Purpose 132
Organizing Elements Clearly and Meaningfully 133
Consistency 133
Starting Point 135
Ordering of Data and Content 136
Navigation and Flow 139
Visually Pleasing Composition 141
Distinctiveness 161
Focus and Emphasis 162
Conveying Depth of Levels or a Three-Dimensional Appearance 165
Presenting Information Simply and Meaningfully 168
Application and Page Size 178
Application Screen Elements 184
Organization and Structure Guidelines 220
The Web — Web sites and Web Pages 230

	Intranet Design Guidelines	258
	Extranet Design Guidelines	259
	Small Screens	259
	Weblogs	260
	Statistical Graphics	261
	Types of Statistical Graphics	273
	Flow Charts	283
	Technological Considerations in Interface Design	284
	Graphical Systems	284
	Web Systems	287
	The User Technology Profile Circa 2006	292
	Examples of Screens	293
	Example 1	293
	Example 2	297
	Example 3	300
	Example 4	301
	Example 5	302
	Example 6	303
	Example 7	305
	Step 3 Exercise	306
Step 4	**Develop System Menus and Navigation Schemes**	**307**
	Structures of Menus	308
	Single Menus	308
	Sequential Linear Menus	309
	Simultaneous Menus	309
	Hierarchical or Sequential Menus	310
	Connected Menus	311
	Event-Trapping Menus	313
	Functions of Menus	313
	Navigation to a New Menu	314
	Execute an Action or Procedure	314
	Displaying Information	314
	Data or Parameter Input	314
	Content of Menus	314
	Menu Context	315
	Menu Title	315
	Choice Descriptions	315
	Completion Instructions	315
	Formatting of Menus	315
	Consistency	316
	Display	316
	Presentation	316
	Organization	317
	Complexity	320
	Item Arrangement	321
	Ordering	321

Groupings 323
Selection Support Menus 325
Phrasing the Menu 328
Menu Titles 329
Menu Choice Descriptions 330
Menu Instructions 332
Intent Indicators 332
Keyboard Shortcuts 333
Selecting Menu Choices 337
Initial Cursor Positioning 337
Choice Selection 338
Defaults 339
Unavailable Choices 340
Mark Toggles or Settings 340
Toggled Menu Items 341
Web Site Navigation 342
Web Site Navigation Problems 343
Web Site Navigation Goals 344
Web Site Navigation Design 345
Maintaining a Sense of Place 367
Kinds of Graphical Menus 369
Menu Bar 369
Pull-Down Menu 371
Cascading Menus 375
Pop-Up Menus 377
Tear-Off Menus 379
Iconic Menus 380
Pie Menus 380
Graphical Menu Examples 382
Example 1 382

Step 5 Select the Proper Kinds of Windows 385
Window Characteristics 385
The Attraction of Windows 386
Constraints in Window System Design 388
Components of a Window 390
Frame 390
Title Bar 391
Title Bar Icon 391
Window Sizing Buttons 392
What's This? Button 393
Menu Bar 393
Status Bar 394
Scroll Bars 394
Split Box 394
Toolbar 394
Command Area 395

Size Grip 395
Work Area 395
Window Presentation Styles 395
Tiled Windows 396
Overlapping Windows 397
Cascading Windows 398
Picking a Presentation Style 399
Types of Windows 399
Primary Window 400
Secondary Windows 401
Dialog Boxes 407
Property Sheets and Property Inspectors 408
Message Boxes 411
Palette Windows 413
Pop-Up Windows 413
Organizing Window Functions 414
Window Organization 414
Number of Windows 415
Sizing Windows 416
Window Placement 417
The Web and the Browser 419
Browser Components 419
Step 5 Exercise 422

Step 6 Select the Proper Interaction Devices 423
Input Devices 423
Characteristics of Input Devices 424
Other Input Devices 436
Selecting the Proper Input Device 436
Output Devices 440
Screens 440
Speakers 441
Step 6 Exercise 441

Step 7 Choose the Proper Screen-Based Controls 443
Operable Controls 445
Buttons 445
Text Entry/Read-Only Controls 461
Text Boxes 461
Selection Controls 468
Radio Buttons 468
Check Boxes 478
Palettes 488
List Boxes 493
List View Controls 503
Drop-Down/Pop-Up List Boxes 503

Combination Entry/Selection Controls 509
Spin Boxes 509
Combo Boxes 512
Drop-Down/Pop-Up Combo Boxes 514
Other Operable Controls 517
Slider 517
Tabs 521
Date-Picker 524
Tree View 525
Scroll Bars 526
Custom Controls 531
Presentation Controls 531
Static Text Fields 532
Group Boxes 533
Column Headings 534
ToolTips 535
Balloon Tips 537
Progress Indicators 539
Sample Box 540
Scrolling Tickers 542
Selecting the Proper Controls 542
Entry versus Selection — A Comparison 543
Comparison of GUI Controls 544
Control Selection Criteria 547
Choosing a Control Form 548
Examples 552
Example 1 552
Example 2 553
Example 3 556
Example 4 557
Example 5 558
Example 6 559
Step 7 Exercise 561

Step 8 Write Clear Text and Messages 563
Words, Sentences, Messages, and Text 564
Readability 564
Choosing the Proper Words 565
Writing Sentences and Messages 568
Kinds of Messages 570
Presenting and Writing Text 578
Window Title, Conventions, and Sequence Control Guidance 582
Content and Text for Web Pages 584
Words 584
Page Text 585
Page Title 589

	Headings and Headlines	589
	Instructions	590
	Error Messages	590
	Step 8 Exercise	591
Step 9	**Provide Effective Feedback and Guidance and Assistance**	**593**
	Providing the Proper Feedback	594
	Response Time	594
	Dealing with Time Delays	598
	Blinking for Attention	601
	Use of Sound	602
	Guidance and Assistance	603
	Preventing Errors	603
	Problem Management	604
	Providing Guidance and Assistance	606
	Instructions or Prompting	608
	Help Facility	608
	Contextual Help	613
	Task-Oriented Help	617
	Reference Help	619
	Wizards	620
	Hints or Tips	622
	Step 9 Exercise	623
Step 10	**Provide Effective Internationalization and Accessibility**	**625**
	International Considerations	626
	Localization	626
	Cultural Considerations	627
	Words and Text	628
	Images and Symbols	631
	Color, Sequence, and Functionality	633
	Requirements Determination and Testing	635
	Accessibility	635
	Types of Disabilities	636
	Accessibility Design	636
	Step 10 Exercise	650
Step 11	**Create Meaningful Graphics, Icons, and Images**	**651**
	Icons	652
	Kinds of Icons	652
	Characteristics of Icons	654
	Influences on Icon Usability	654
	Choosing Icons	657
	Choosing Icon Images	659
	Creating Icon Images	659
	Drawing Icon Images	664
	Icon Animation and Audition	665
	The Icon Design Process	667
	Screen Presentation	667

Multimedia 669
Graphics 669
Images 671
Photographs/Pictures 676
Video 677
Diagrams 678
Drawings 681
Animation 681
Audition 683
Combining Mediums 686
Step 11 Exercise 689

Step 12 Choose the Proper Colors 691
Color — What Is It? 692
RGB 694
HSV 694
Dithering 694
Color Uses 695
Color as a Formatting Aid 695
Color as a Visual Code 696
Other Color Uses 696
Possible Problems with Color 696
High Attention-Getting Capacity 696
Interference with Use of Other Screens 697
Varying Sensitivity of the Eye to Different Colors 697
Color-Viewing Deficiencies 697
Color Connotations 698
Cross-Disciplinary and Cross-Cultural Differences 700
Color — What the Research Shows 700
Color and Human Vision 701
The Lens 701
The Retina 701
Choosing Colors 702
Choosing Colors for Categories of Information 703
Colors in Context 703
Usage 704
Discrimination and Harmony 704
Emphasis 706
Common Meanings 706
Location 707
Ordering 708
Foregrounds and Backgrounds 708
Three-Dimensional Look 709
Color Palette, Defaults, and Customization 710
Grayscale 711
Text in Color 712
Monochromatic Screens 712

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

Step 13 Organize and Layout Windows and Pages 727
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

Step 14 Test, Test, and Retest 767
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

References **811**

Index **835**

CHAPTER

2

Characteristics of Graphical and Web User Interfaces

The graphical user interface differed significantly from its text-based forefather. The Web interface differs from a GUI interface in significant ways also. In this chapter, the following characteristics of GUI and Web interfaces are reviewed:

- Interaction styles.
- The concept of direct manipulation.
- The characteristics of graphical interfaces.
- The characteristics of Web interfaces.
- Web page versus Web application design.
- The general principles of user interface design.

Interaction Styles

An interaction style is simply the method, or methods, by which the user and a computer system communicate with one another. Today the designer has a choice of several interaction styles in graphical system or Web page and application design. They are as follows:

- Command line
- Menu selection
- Form fill-in
- Direct manipulation
- Anthropomorphic

The choice of interaction styles to be considered may be limited based upon the type of system being developed and the characteristics of the input and output devices to be used for the interface. A brief description of each interaction style follows. A summary of the advantages and disadvantages of each one is shown in Table 2.1.

Command Line

The command-line interface is the oldest and original user interaction style. It requires the user to press a function key or type a command into a designated entry area on a screen. The commands may be single characters, abbreviations, words, or multiple words and codes. The command-line style is powerful, offering immediate access to system functions. It is also flexible and able to incorporate options or parameters to vary its behavior. One problem with command lines is that they must be remembered and they test one's power of recall. No clues about what commands are available exist on the screen. Another problem is that command lines can be cryptic and obscure with complex syntax. They are also very prone to, and intolerant of, typing errors that can lead to user frustration.

Menu Selection

A menu is a set of options or choices from which a user must choose. On screens, the user selects a choice with a pointing device or keystroke. Typically, some kind of visual feedback is then provided to indicate the option selected. Menu selections can also be provided by voice as exemplified by the "Press 1 to..." encountered after telephone calls to a business or organization. A newer version of telephone voice menus now appearing asks the caller to speak a request (or command), which, hopefully, the voice recognition system will understand. (Is this an auditory command line?)

Screen menus are advantageous because they utilize a person's much stronger powers of recognition, not recall. (More about this in Step 3.) However, menu choice labels must be meaningful and understandable for the menu to be truly effective. Otherwise, speed of use will be degraded and errors increase. Menus can break a complex interaction into small steps, which structure and aid the decision-making process. This is especially helpful for infrequent users who are unfamiliar with the system. On the other hand, many small steps may slow the knowledgeable user. Techniques, however, are available to overcome these problems for the expert. Menus are discussed in more detail in Step 4.

Form Fill-in

The form fill-in style is very useful for collecting information. Today's typical form-structured screen contains a series of controls or fields into which the user either types information or selects an option, or options, from a listing of choices. (Technically, a listing of choices presented to users is also a menu.) In old text-based systems, however, screen forms were composed entirely of fields into which the user had to type information. Screen fill-in forms are derived from their antecedents, paper forms. An advantage of a form is its familiarity. If it is designed well, a form will aid the user in

understanding its purpose and allow fast and easy entry of information. Conversely, a poorly designed screen form can be inefficient and aggravating to complete. Screen form design principles are discussed in Step 3, and screen-based controls are discussed in Step 7.

Direct Manipulation

A direct manipulation interface, as found in graphical systems, enables the user to directly interact with elements presented on the screen. These elements (called objects) replace the keyed entry of commands and menus. Users typically select screen objects and actions by using pointing mechanisms, such as the mouse or joystick, instead of the traditional keyboard. They navigate the screen and execute commands by using menu bars and pull-down menus. The direct manipulation interaction style will be discussed in more detail in the "Graphical User Interface" section later in this chapter.

Anthropomorphic

An anthropomorphic interface tries to interact with people the same way people interact with each other. Anthropomorphic interfaces include spoken natural language dialogues, hand gestures, facial expressions, and eye movements. The development of these kinds of interfaces requires an understanding of human behavior; how people interact with one another, the meaning of gestures and expressions, what people mean when they look at things, and so forth. Wouldn't it be nice, for example, if the system could track eye movement across the screen to a menu, and then recognize the blink of an eye to select the choice being viewed? Or, if a frown elicited the automatic display of a help screen and a smile meant OK?

Many of these interfaces exist, in some form, only in the laboratory or in researchers' thoughts. The most advanced interface is the natural language dialog. Structured subsets of typed or spoken words are now capable of being recognized in limited ways, as exemplified by the telephone voice recognition described above.

With the exception of an anthropomorphic interface, most current systems contain a blend of these interaction styles. The proper mix can be created only after an understanding of the user, the tasks to be performed, and the goals of the system are obtained.

Table 2.1: Some Advantages and Disadvantages of Interaction Styles

STYLE	ADVANTAGES	DISADVANTAGES
Command Line	Powerful	Commands must be memorized
	Flexible	Requires learning
	Appeals to expert users	Intolerant of typing errors
	Conserves screen space	Difficult for casual users

(continued)

Table 2.1 *(continued)*

STYLE	ADVANTAGES	DISADVANTAGES
Menu Selection	Utilizes recognition memory	May slow knowledgeable users
	Reduces interaction complexity	Consumes screen space
	Aids decision-making process	May create complex menu hierarchies
	Minimizes typing	
	Aids casual users	
Form Fill-in	Familiar format	Consumes screen space
	Simplifies information entry	Requires careful and efficient design
	Requires minimal training	Does not prevent typing errors
Direct Manipulation	Faster learning	Greater design complexity
	Easier remembering	Window manipulation requirements
	Exploits visual/spatial cues	Requires icon recognition
	Easy error recovery	Inefficient for touch typists
	Provides context	Increased chance for screen clutter
	Immediate feedback	
Anthropomorphic	Natural	Difficult to implement

The Graphical User Interface

A *user interface*, as recently described, is a collection of techniques and mechanisms to interact with something. In a *graphical* interface, the primary interaction mechanism is a pointing device of some kind. This device is the electronic equivalent to the human hand. What the user interacts with is a collection of elements referred to as *objects*. They can be seen, heard, touched, or otherwise perceived. Objects are always visible to the user and are used to perform tasks. They are interacted with as entities independent of all other objects. People perform operations, called actions, on objects. The operations include accessing and modifying objects by pointing, selecting, and manipulating. All objects have standard resulting behaviors.

The Popularity of Graphics

Graphics revolutionized design and the user interface. A graphical screen bore scant resemblance to its earlier text-based colleagues. Whereas the older text-based screen

possessed a one-dimensional, text-oriented, form-like quality, graphic screens assumed a three-dimensional look. Information floated in windows, small rectangular boxes that seemed to rise above the background plane. Windows could also float above other windows. Controls appeared to rise above the screen and move when activated. Lines appeared to be etched into the screen. Information could appear and disappear as needed, and in some cases text could be replaced by graphical images called icons. These icons could represent objects or actions.

Screen navigation and commands are executed through menu bars and pull-down menus. Menus "pop up" on the screen. In the screen body, selection fields such as radio buttons, check boxes, list boxes, and palettes coexisted with the reliable old text entry field. More sophisticated text entry fields with attached or drop-down menus of alternative options also became available. Screen objects and actions are typically selected through use of pointing mechanisms, such as the mouse or joystick, instead of the traditional keyboard.

Increased computer power and the vast improvement in the display enable a system to react to the user's actions quickly, dynamically, and meaningfully. This new interface was characterized as representing one's "desktop" with scattered notes, papers, and objects such as files, trays, and trash cans arrayed around the screen. It is sometimes referred to as the WIMP interface: windows, icons, menus, and pointing device.

Graphic presentation of information utilizes a person's information-processing capabilities much more effectively than other presentation methods. Properly used, it reduces the requirement for perceptual and mental information recoding and reorganization, and also reduces the memory loads. It permits faster information transfer between computers and people by permitting more visual comparisons of amounts, trends, or relationships; more compact representation of information; and simplification of the perception of structure. Graphics also can add appeal or charm to the interface and permit greater customization to create a unique corporate or organization style.

The Concept of Direct Manipulation

The term used to describe graphical systems with this style of interaction was first used by Shneiderman (1982). He called them "direct manipulation" systems, suggesting that they possess the following characteristics:

The system is portrayed as an extension of the real world. It is assumed that a person is already familiar with the objects and actions in his or her environment of interest. The system simply replicates them and portrays them on a different medium, the screen. A person has the power to access and modify these objects, including windows. A person is allowed to work in a familiar environment and in a familiar way, focusing on the data, not the application and tools. The physical organization of the system, which most often is unfamiliar, is hidden from view and is not a distraction.

Objects and actions are continuously visible. Like one's desktop, objects are continuously visible. Reminders of actions to be performed are also obvious, where labeled buttons replace complex syntax and command names. Cursor action and motion occurs in physically obvious and intuitively natural ways. Nelson (1980)

described this concept as *virtual reality*, a representation of reality that can be manipulated. Hatfield (1981) is credited with calling it WYSIWYG (what you see is what you get). Rutkowski (1982) described it as *transparency*, where one's intellect is applied to the task, not the tool. Hutchins, Hollan, and Norman (1986) considered it direct involvement with the world of objects rather than communicating with an intermediary.

One problem in direct manipulation, however, is that there is no direct analogy on the desk for all necessary windowing operations. A piece of paper on one's desk maintains a constant size, never shrinking or growing. Windows can do both. Solving this problem required embedding a control panel, a familiar concept to most people, in a window's border. This control panel is manipulated, not the window itself.

Actions are rapid and incremental with visible display of results. Because tactile feedback is not yet possible (as would occur with one's hand when one touches something), the results of actions are immediately displayed visually on the screen in their new and current form. Auditory feedback may also be provided. The impact of a previous action is quickly seen, and the evolution of tasks is continuous and effortless.

Incremental actions are easily reversible. Finally, actions, if discovered to be incorrect or not desired, can be easily undone.

Earlier Direct Manipulation Systems

Using the aforementioned definition, the concept of direct manipulation actually preceded the first graphical system. The earliest full-screen text editors possessed similar characteristics. Screens of text resembling a piece of paper on one's desk could be created (extension of real world) and then reviewed in their entirety (continuous visibility). Editing or restructuring could be easily accomplished (through rapid incremental actions) and the results immediately seen. Actions could be reversed when necessary. It took the advent of graphical systems to crystallize the direct manipulation concept, however.

Indirect Manipulation

In practice, direct manipulation of *all* screen objects and actions may not be feasible because of the following:

- The operation may be difficult to conceptualize in the graphical system.
- The graphics capability of the system may be limited.
- The amount of space available for placing manipulation controls in the window border may be limited.
- It may be difficult for people to learn and remember all of the necessary operations and actions.

When this occurs, *indirect manipulation* is provided. Indirect manipulation substitutes words and text, such as pull-down or pop-up menus, for symbols, and substitutes typing for pointing. Most window systems are a combination of both direct and indirect

manipulation. A menu may be accessed by pointing at a menu icon and then selecting it (direct manipulation). The menu itself, however, is a textual list of operations (indirect manipulation). When an operation is selected from the list, by pointing or typing, the system executes it as a command.

Which style of interaction — direct manipulation, indirect manipulation, or a combination of both — is best, under what conditions and for whom, remains a question for which the answer still eludes us.

Graphical Systems: Advantages and Disadvantages

Graphical systems burst upon the office with great promise. The simplified interface they presented was thought to reduce the memory requirements imposed on the user, make more effective use of one's information-processing capabilities, and dramatically reduce system learning requirements. Experience indicates that for many people they have done all these things.

Advantages

The success of graphical systems has been attributed to a host of factors. The following have been commonly referenced in literature and endorsed by their advocates as advantages of these systems:

Symbols recognized faster than text. Research has found that symbols can be recognized faster and more accurately than text, and that the graphical attributes of icons, such as shape and color, are very useful for quickly classifying objects, elements, or text by some common property. An example of a good classification scheme that speeds up recognition is the icons developed for indicating the kind of message being presented to the user of the system. The text of an informational message is preceded by an "i" in a circle, a warning message by an exclamation point, and a critical message by another unique symbol. These icons allow speedy recognition of the type of message being presented.

Faster learning. Research has also found that a graphical, pictorial representation aids learning, and symbols can also be easily learned.

Faster use and problem solving. Visual or spatial representation of information has been found to be easier to retain and manipulate, and leads to faster and more successful problem solving. Symbols have also been found to be effective in conveying simple instructions.

Easier remembering. Because of greater simplicity, it is easier for casual users to retain operational concepts.

More natural. Graphic representations of objects are thought to be more natural and closer to innate human capabilities. In human beings, actions and visual skills emerged before languages. It has also been suggested that symbolic displays are more natural and advantageous because the human mind has a powerful image memory.

Exploits visual/spatial cues. Spatial relationships are usually found to be understood more quickly than verbal representations. Visual thinking is believed to be better than logical thinking.

Fosters more concrete thinking. Displayed objects are directly in the high-level task domain, or directly usable in their presented form. There is no need mentally to decompose tasks into multiple commands with complex syntactic form. The need for abstract thinking is therefore minimized.

Provides context. Displayed objects are visible, providing a picture of the current context.

Fewer errors. More concrete thinking affords fewer opportunities for errors. Reversibility of actions reduces error rates because it is always possible to undo the last step. Error messages are less frequently needed.

Increased feeling of control. The user initiates actions and feels in control. This increases user confidence and hastens system mastery.

Immediate feedback. The results of actions furthering user goals can be seen immediately. Learning is quickened. If the response is not in the desired direction, the direction can be changed quickly.

Predictable system responses. Predictable system responses also speed learning.

Easily reversible actions. The user has more control. This ability to reverse unwanted actions also increases user confidence and hastens system mastery.

Less anxiety concerning use. Hesitant or new users feel less anxiety when using the system because it is so easily comprehended, is easy to control, and has predictable responses and reversible actions.

More attractive. Direct-manipulation systems are more entertaining, clever, and appealing. This is especially important for the cautious or skeptical user.

May consume less space. Icons may take up less space than the equivalent in words. More information can often be packed in a given area of the screen. This, however, is not always the case.

Replaces national languages. Language-based systems are seldom universally applicable. Language translations frequently cause problems in a text-based system. Icons possess much more universality than text and are much more easily comprehended worldwide.

Easily augmented with text displays. Where graphical design limitations exist, direct-manipulation systems can easily be augmented with text displays. The reverse is not true.

Low typing requirements. Pointing and selection controls, such as the mouse or trackball, eliminate the need for typing skills.

Smooth transition from command language system. Moving from a command language to a direct-manipulation system has been found to be easy. The reverse is not true.

Disadvantages

The body of positive research, hypotheses, and comment concerning graphical systems is being challenged by some studies, findings, and opinions that indicate that graphical representation and interaction may not necessarily always be better. Indeed,

in some cases, it may be poorer than pure textual or alphanumeric displays. Trying to force all system components into a graphical format may be doing a disservice to the user. Some also feel that, as graphical systems become increasingly sophisticated and continue to expand, interfaces become increasingly more complex, sometimes arcane, and even bizarre. Among the disadvantages put forth are the following:

Greater design complexity. The elements and techniques available to the graphical screen designer far outnumber those that were at the disposal of the text-based screen designer. Controls and basic alternatives must be chosen from a pile of choices numbering in excess of 50. (Conversely, alternatives available to the text-based screen designer numbered about 15.) This design potential may not necessarily result in better design, unless the choices are thoughtfully selected and consistently and simply applied. Proper window types must also be chosen and colors selected from a seemingly unending rainbow of alternatives. With graphics, the skill of the designer is increasingly challenged. Poor design can undermine acceptance.

Learning still necessary. The first time one encounters many graphical systems, what to do is not immediately obvious. The user may not know the meanings of many words and icons. It is often not possible to guess their meanings, especially the more arbitrary symbols. The user may also have to learn how to use a pointing device. A severe learning and remembering requirement is imposed on many users, and it takes a while to get up to speed. A text-based system could easily be structured to incorporate a set of clear instructions: (1) Do this, (2) now do this, and so on.

System providers estimate that becoming accustomed to a graphical interface should require about eight hours of training. Other experts say the learning time is closer to 20 or 30 hours.

Lack of experimentally-derived design guidelines. The graphical interface is still burdened today by a lack of widely available experimentally-derived design guidelines. Early on, more developer interest existed in solving technical rather than usability issues, so few studies to aid in making design decisions were performed. Today studies being performed in usability laboratories are rarely published. This occurs because of several factors. First, builders of platforms and packages will not publish their study results because they want to maintain a competitive advantage. If they find a better way to do something, or present something, why tell the competition? Let them make the same mistake, or find the answer themselves.

Second, the studies are often specific to a particular function or task. They may not be generally applicable. Third, it takes time and effort to publish something. The developer in today's office seldom has the time. Finally, it is also difficult to develop studies evaluating design alternatives because of increased GUI complexity. Too many variables that must be controlled make meaningful cause-and-effect relationships very difficult to uncover.

Consequently, there is too little understanding of how most design aspects relate to productivity and satisfaction.

Inconsistencies in technique and terminology. Many differences in technique, terminology, and look and feel exist among various graphical system providers, and even among successive versions of the same system. These inconsistencies occur because of copyright and legal implications, product differentiation considerations, and our expanding knowledge of the interface. The result is that learning, and relearning, for both designers and users is much more difficult than it should be.

Working domain is the present. While direct-manipulation systems provide context, they also require the user to work in the present. Hulteen (1988), in a takeoff on WYSIWYG, suggests "What you see is all you get." Walker (1989) argued that language takes you out of the here and now and the visually present. Language, she continues, makes it easier to find things.

Not always familiar. Symbolic representations may not be as familiar as words or numbers. People have been exposed to words and numbers for a long time. Research has found that numeric symbols elicit faster responses than graphic symbols in a visual search task. One developer had to modify a new system during testing by replacing iconic representations with a textual outline format. The users, lawyers, were unfamiliar with icons and demanded a more familiar format.

Human comprehension limitations. Human limitations may also exist in terms of one's ability to deal with the increased complexity of the graphical interface. The variety of visual displays can still challenge all but the most sophisticated users. The number of different icons that can be introduced is also restricted because of limitations in human comprehension. Studies continually find that the number of different symbols a person can differentiate and deal with is much more limited than text. Some researchers note that claims for the easy understanding of pictograms are exaggerated, and that recognizing icons requires much perceptual learning, abstracting ability, and intelligence.

The motor skills required may also challenge all but the most sophisticated users. Correctly double-clicking a mouse, for example, is difficult for some people.

Window manipulation requirements. Window handling and manipulation time are still excessive and repetitive. This wastes time and interrupts the decision-making needed to perform tasks and jobs.

Production limitations. The number of symbols that can be clearly produced using today's technology is still limited. A body of recognizable symbols must be produced that are equally legible and equally recognizable using differing technologies. This is extremely difficult today.

Few tested icons exist. Icons, like typefaces, must appear in different sizes, weights, and styles. As with text, an entire font of clearly recognizable symbols must be developed. It is not a question of simply developing an icon and enlarging or reducing it. Changing an icon's size can differentially affect symbol line widths, open areas, and so forth, dramatically affecting its recognizability. Typeface design is literally the product of 300 years of experimentation and study. Icons must be researched, designed, tested, and then introduced into the marketplace. The consequences of poor or improper design will be confusion and lower productivity for users.

Inefficient for touch typists. For an experienced touch typist, the keyboard is a very fast and powerful device. Moving a mouse or some other pointing mechanism may be slower.

Inefficient for expert users. Inefficiencies develop when there are more objects and actions than can fit on the screen. Concatenation for a command language is impossible.

Not always the preferred style of interaction. Not all users prefer a pure iconic interface. A study comparing commands illustrated by icons, icons with text, or text-only, found that users preferred alternatives with textual captions.

Not always fastest style of interaction. Another study has found that graphic instructions on an automated bank teller machine were inferior to textual instructions.

Increased chances of clutter and confusion. A graphical system does not guarantee elimination of clutter on a screen. Instead, the chance for clutter is increased, thereby increasing the possibility of confusion. How much screen clutter one can deal with is open to speculation. The possibility that clutter may exist is evidenced by the fact that many people, when working with a window, expand it to fill the entire display screen. This may be done to reduce visual screen clutter. Mori and Hayashi (1993) found that visible windows, not the focus of attention, degraded performance in the window being worked on.

The futz and fiddle factor. With the proliferation of computer games, computer usage can be wasteful of time. Stromoski (1993) estimates that five hours a week in the office are spent playing and tinkering. Experts have said that the most used program in Microsoft Windows is Solitaire! Tinkering includes activities such as creating garish documents reflecting almost every object property (font size, style, color, and so on) available.

Futzing and fiddling does have some benefits, however. It is a tool for learning how to use a mouse, for example, and it is a vehicle for exploring the system and becoming familiar with its capabilities. It is of value when done in moderation.

May consume more screen space. Not all applications will consume less screen space. A listing of names and telephone numbers in a textual format will be more efficient to scan than a card file.

Hardware limitations. Good design also requires hardware of adequate power, processing speed, screen resolution, and graphic capability. Insufficiencies in these areas can prevent a graphic system's full potential from being realized.

Some Studies and a Conclusion

The many benefits of one interaction style versus another are anecdotal. This has often made the debate between advocates of graphical and other styles of interaction more emotional than scientific. This is certainly true for many of the arguments. Over the past couple of decades a variety of studies have been performed comparing graphical systems with other interaction styles. In some usability studies graphical systems were found superior, in other studies other interaction techniques were found superior, and in some cases no differences were found. Perhaps the best conclusion was drawn by Whiteside et al. (1985) who compared the usability characteristics of seven systems, including the direct-manipulation, menu, and command language styles of interaction. They found that user performance did not depend on the type of system. There

were large differences in learnability and usability among all. How well the system was *designed* was the best indicator of success, not the style of interaction.

Research and experience have shown that different interface styles also have different strengths and weaknesses. Some concepts and tasks are very hard to convey symbolically and do not seem to be suited for a pure graphical presentation. Other concepts and tasks, however, may be well suited to such an approach. Which tasks are best suited for which styles still needs continuing study. Finally, all users may not like all aspects of a graphical system. The design should reflect this. In summary, the following is clear:

- The design of an interface, and not its interaction style, is the best determinant of ease of use.
- User preferences must be considered in choosing an interaction style.
- In the overwhelming majority of cases, words are more meaningful to users than icons.
- The content of a graphic screen is critical to its usefulness. The wrong presentation or a cluttered presentation may actually lead to greater confusion, not less.
- The success of a graphical system depends on the skills of its designers in following established principles of usability.

Characteristics of the Graphical User Interface

A graphical system possesses a set of defining concepts. Included are sophisticated visual presentation, pick-and-click interaction, a restricted set of interface options, visualization, object orientation, extensive use of a person's recognition memory, and concurrent performance of functions.

Sophisticated Visual Presentation

Visual presentation is the visual aspect of the interface. It is what people see on the screen. The sophistication of a graphical system permits displaying lines, including drawings and icons. It also permits the displaying of a variety of character fonts, including different sizes and styles. The display of 16 million or more colors is possible on some screens. Graphics also permit animation and the presentation of photographs and motion video.

The meaningful interface elements visually presented to the user in a graphical system include windows (primary, secondary, or dialog boxes), menus (menu bar, pull-down, pop-up, cascading), icons to represent objects such as programs or files, assorted screen-based controls (text boxes, list boxes, combination boxes, settings, scroll bars, and buttons), a mouse or other pointing device, and the cursor. The objective is to reflect visually on the screen the real world of the user as realistically, meaningfully, simply, and clearly as possible.

Pick-and-Click Interaction

Elements of a graphical screen upon which some action is to be performed must first be identified. The motor activity required of a person to identify this element for a

proposed action is commonly referred to as *pick*, and the signal to perform an action as *click*. The primary mechanism for performing this pick-and-click is most often the mouse and its buttons. The user moves the mouse pointer to the relevant element (pick) and the action is signaled (click). Pointing allows rapid selection and feedback. The eye, hand, and mind seem to work smoothly and efficiently together.

The secondary mechanism for performing these selection actions is the keyboard. Most systems permit pick-and-click to be performed using the keyboard as well.

Restricted Set of Interface Options

The array of alternatives available to the user is what is presented on the screen or what may be retrieved through what is presented on the screen — nothing less, nothing more. This concept fostered the acronym WYSIWYG.

Visualization

Visualization is a cognitive process that enables people to understand information that is difficult to perceive, because it is either too voluminous or too abstract. It involves changing an entity's representation to reveal gradually the structure and/or function of the underlying system or process. Presenting specialized graphic portrayals facilitates visualization. The best visualization method for an activity depends on what people are trying to learn from the data. The goal is not necessarily to reproduce a realistic graphical image, but to produce one that conveys the most relevant information. Effective visualizations can facilitate mental insights, increase productivity, and foster faster and more accurate use of data.

Object Orientation

A graphical system consists of objects and actions. *Objects* are what people see on the screen. They are manipulated as a single unit. A well-designed system keeps users focused on objects, not on how to carry out actions. Objects can be composed of *subobjects*. For example, an object may be a document. The document's subobjects may be a paragraph, sentence, word, and letter.

IBM's System Application Architecture Common User Access Advanced Interface Design Reference (SAA CUA) (IBM, 1991) breaks objects into three meaningful classes: data, container, and device. *Data objects* present information. This information, either text or graphics, normally appears in the body of the screen. It is essentially the screen-based controls for information collection or presentation organized on the screen.

Container objects are objects that hold other objects. They are used to group two or more related objects for easy access and retrieval. There are three kinds of container objects: the workplace, folders, and workareas. The *workplace* is the desktop, the storage area for all objects. *Folders* are general-purpose containers for long-term storage of objects. *Workareas* are temporary storage folders used for storing multiple objects currently being worked on.

Device objects represent physical objects in the real world, such as printers or trash cans. These objects may contain others for acting upon. A file, for example, may be placed in a printer for printing of its contents.

Microsoft Windows specifies the characteristics of objects depending upon the relationships that exist between them. Objects can exist within the context of other objects, and one object may affect the way another object appears or behaves. These relationships are called collections, constraints, composites, and containers.

A *collection* is the simplest relationship — the objects sharing a common aspect. A collection might be the result of a query or a multiple selection of objects. Operations can be applied to a collection of objects.

A *constraint* is a stronger object relationship. Changing an object in a set affects some other object in the set. A document being organized into pages is an example of a constraint.

A *composite* exists when the relationship between objects becomes so significant that the aggregation itself can be identified as an object. Examples include a range of cells organized into a spreadsheet, or a collection of words organized into a paragraph.

A *container* is an object in which other objects exist. Examples include text in a document or documents in a folder. A container often influences the behavior of its content. It may add or suppress certain properties or operations of objects placed within it, control access to its content, or control access to kinds of objects it will accept.

These relationships help define an object's *type*. Similar traits and behaviors exist in objects of the same object type.

Another important object characteristic is *persistence*. Persistence is the maintenance of a state once it is established. An object's state (for example, window size, cursor location, scroll position, and so on) should always be automatically preserved when the user changes it.

Properties or Attributes of Objects

Objects also have properties or attributes. Properties are the unique characteristics of an object. Properties help to describe an object and can be changed by users. Examples of properties are text styles (such as normal or italics), font sizes (such as 10 or 12 points), or window background colors (such as black or blue).

Actions

In addition to objects, there are actions. People take actions on objects. They manipulate objects in specific ways (commands) or modify the properties of objects (property or attribute specification).

Commands are actions that manipulate objects. They may be performed in a variety of ways, including direct manipulation or a command button. They are executed immediately when selected. Once executed, they cease to be relevant. Examples of commands include opening a document, printing a document, closing a window, and quitting an application.

Property/attribute specification actions establish or modify the attributes or properties of objects. When selected, they remain in effect until deselected. Examples include selecting cascaded windows to be displayed, a particular font style, or a particular color.

The following is a typical property/attribute specification sequence:

1. The user selects an object such as several words of text.
2. The user then selects an action to apply to that object, such as the action Bold.
3. The selected words are made bold and will remain bold until selected and changed again.

A series of actions may be performed on a selected object. Performing a series of actions on an object also permits and encourages system learning through exploration.

Application versus Object or Data Orientation

Earlier graphical systems were usually application-oriented, a continuation of the philosophy that enveloped text-based systems. When a text-based system was developed, it was called an application. As graphical systems evolved, developers usually thought in terms of applications as well. When a real picture of the user began to emerge, it finally became evident that people think in terms of tasks, not applications. They choose objects and then act upon them.

An application-oriented approach takes an action:object approach, like the following:

Action> 1. The user opens an application such as word processing.

Object> 2. The user then selects a file or other object such as a memo.

An object-oriented object:action approach does the following:

Object> 1. The user chooses an object such as a memo.

Action> 2. The user then selects an application such as word processing.

The object:action approach permits people to focus more easily on their task and minimizes the visibility of the operating system and separate applications. Many experienced users may have difficulty switching from one approach to another because an old interaction behavior must be unlearned and a new one learned. New users should not experience these problems because it more accurately reflects a person's thinking. In any one interface, it is critical that a consistent orientation be maintained, either an object:action or an action:object approach.

Views

Views are ways of looking at an object's information. IBM's SAA CUA describes four kinds of views: composed, contents, settings, and help.

Composed views present information and the objects contained within an object. They are typically associated with data objects and are specific to tasks and products being worked with. *Contents* views list the components of objects. *Settings* views permit seeing and changing object properties. *Help* views provide all of the help functions.

Use of Recognition Memory

Continuous visibility of objects and actions encourages use of a person's more powerful recognition memory. This eliminates the "out of sight, out of mind" problem.

Concurrent Performance of Functions

Graphic systems may do two or more things at one time. Multiple programs may run simultaneously. When a system is not busy on a primary task, it may process background tasks (cooperative multitasking). When applications are running as truly separate tasks, the system may divide the processing power into time slices and allocate portions to each application (preemptive multitasking). Data may also be transferred

between programs. It may be temporarily stored on a clipboard for later transfer or automatically swapped between programs.

The Web User Interface

The expansion of the World Wide Web since the early 1990s has been truly amazing. Once simply a communication medium for scientists and researchers, its many and pervasive tentacles have spread deeply into businesses, organizations, and homes around the world. Unlike earlier text-based and GUI systems that were developed and nurtured in an organization's Data Processing and Information Systems groups, the Web's roots were sown in a market-driven society thirsting for convenience and information.

Initially, Web interface design was essentially the design of navigation and the presentation of information. It was about content, not data. In recent years a second and dual focus has emerged — the design of Web applications. Applications previously developed for use on graphical systems have increasingly migrated to the Web for their foundation. Content- or information-focused interface design is typically called *Web page* design. An application-focused interface is usually referred to as *Web application* design. Both styles share many similar features, both being heavily graphical and information rich. Significant differences exist as well, however. Web page interface design is largely a matter of properly balancing the structure and relationships of menus, content, and other linked documents or graphics. The design goal is to build a hierarchy of menus and pages that feels natural, is well structured, is easy to use, and is truthful. The Web page is a navigation environment where people move frequently between pages of information. A Web application is usually designed to collect and process data. Applications typically consume most or all of a screen, and can monopolize the user's attention for a long period of time. Applications also may be kept up and running continuously.

The dividing line between page and application design is not always clear. In general, however, a Web page's design intent is to mostly provide information. An application is designed to let a person do and save something.

Web interface design is difficult for several reasons. First, its underlying design language, HTML, was never intended for creating screens to be used by the general population. Its scope of users was expected to be technical. HTML was limited in objects and interaction styles, and did not provide a means for presenting information in the most effective way for people. Second, browser navigation retreated to the pre-GUI era. This era was characterized by a "command" field whose contents had to be learned, and a navigational organization and structure that lay hidden beneath a mostly dark and blank screen. GUIs eliminated the absolute necessity for a command field, providing menus related to the task and the current contextual situation. Browser navigation is mostly confined to a "Back" and "Forward" concept, but "back-to-where" and "forward-to-where" are often unremembered or unknown. Ill-timed use of the Back button can destroy many minutes worth of work. Remaining navigation was willed to Web pages themselves, where the situation only worsened. Numerous links were provided to destinations unknown, invisible navigation buttons lay unrecognizable on the screen, and linked jumps two paragraphs down the page were indistinguishable from those that went to the Ukraine. A third reason that Web page interface

design is more difficult is the main issues concern information architecture and task flow, neither of which is easy to standardize. It is more difficult because of the availability of the various types of multimedia and the desire of many designers to use something simply because it is available. It is also more difficult because users are ill defined and have greatly variable characteristics. The user's tools are also very variable in nature.

Today the Web interface is somewhat a victim of its poor foundation. It is also a victim of its explosive and haphazard growth. Interface design tools are maturing, research-based design guidelines are becoming increasingly available (and are being applied), and knowledge about users and their needs is expanding. Then, the ultimate goal of a Web that feels natural, is well structured, and is easy to use is beginning to move toward fruition.

The Popularity of the Web

While the introduction of the graphical user interface revolutionized the user interface, the Web revolutionized computing. It enables millions of people scattered across the globe to communicate, access information, publish, and be heard. It enables people to control much of the display and the rendering of Web pages. People can also change aspects such as typography and colors, turn off graphics, decide whether or not to transmit certain data over nonsecure channels, and accept or refuse cookies. Nowhere in the history of computing has the user been given so much control.

Web usage has reflected this popularity. The number of Internet hosts has risen dramatically. In 1984 hosts online exceeded 1,000; in 1987, 10,000; in 1989, 100,000; in 1990, 300,000; in 1992 hosts exceeded one million. Commercialization of the Internet saw even greater expansion of the growth rate. In 1993, Internet traffic was expanding at a 341,634 percent annual growth rate. In 1996, there were nearly 10 million hosts online and 40 million connected people (PBS Timeline). In 2005 the number of Internet hosts exceeded 350 million (Zakon.org, 2006), the number of users one billion (Nielsen, 2005g). The largest percentage of Internet users are in the Asia/Pacific region (36%) according to Morgan Stanley (2005). Percentage of users in other world regions are Europe (24%), North America (23%), South America (5%), and the rest of the world (12%).

User control has had some decided disadvantages for some Web site owners as well. Users have become much more discerning about good design. Slow download times, confusing navigation, confusing page organization, disturbing animation, or other undesirable site features often result in user abandonment of the site for others with a more agreeable interface. People are quick to vote with their mouse, and these warnings should not go unheeded.

Characteristics of a Web Interface

A Web interface possesses many characteristics, some similar to a GUI interface, and, as has already been shown, some different. The following paragraphs examine many of these specific commonalities and differences. Also, the differing characteristics of printed page design and Web page design are compared.

GUI versus Web Page Design

GUI and Web interface design are similar. Both are software designs, they are used by people, they are interactive, they are heavily visual experiences presented through screens, and they are composed of many similar components. Significant differences do exist, however. The following paragraphs highlight the most significant differences. Table 2.2 provides a summary listing. Parts of this discussion are based upon Berry (2000) and Nielsen (1997a).

Devices. In *GUI* design, the characteristics of interface devices such as monitors and modems are well defined, and design variations tend to be restricted. Monitor display capabilities, such as installed fonts and screen size, are established and easily considered in the design process. In *Web* design, no assumptions about the user's interface devices can be made. User devices may range from handheld mechanisms to high-end workstations. (In GUI design, the difference in screen area between a laptop and a high-end workstation is a factor of six; in Web page design, this difference may be as high as 100.) Connection speed bandwidths may also vary by a factor of 1,000. Consequently, WYSIWYG no longer exists in page design. In GUI design, the layout of a screen will look exactly as specified, and the Web page look will be greatly influenced by both the hardware and software. With the Web, the designer has to relinquish full control and share responsibility for the interface with users and their hardware and software.

User focus. *GUI* systems are about well-defined applications and data, and about transactions and processes. Thorough attention must usually be addressed to tasks in need of completion. The *Web* is primarily about information and navigation, an environment where people move back and forth in an unstructured way among many pages of information. Web use is most often characterized by browsing and visual scanning of information to find what information is needed. Increasingly, applications are now being found on the Web, however.

Data/information. *GUI* data is typically created and used by known and trusted sources, people in the user's organization or reputable and reliable companies and organizations. The properties of the system's data are generally known, and the information is typically organized in an understandable and meaningful fashion. A notion of shared data exists, as does a notion of data privacy. The *Web* is full of unknown content typically placed there by others unknown to the user. Typical users don't put information on the Web (except for publishing their own pages). The reliability and truthfulness of found information cannot always be ascertained and trusted. Web content is usually highly variable in organization, and the privacy of the information is often suspect.

User tasks. *GUI* system users install, configure, personalize, start, use, and upgrade programs. They open, use, and close data files. Fairly long times are spent within an individual application, and people become familiar with many of its features and its design. *Web* users link to sites, browse or read pages, fill out forms, register for services, participate in transactions, and download and save content. Movement between pages and sites is often a very rapid activity, with people not gaining familiarity with many sites. The typical Web user has no notion of programs and tends to be much less aware of computer mechanics. Most GUI and Web users to not want to spend the effort required to set up or install anything.

Table 2.2: GUI versus Web Design

	GUI	WEB
Devices	User hardware variations limited. User hardware characteristics well defined. Screens appear exactly as specified.	User hardware variations enormous. Screen appearance influenced by hardware being used.
User Focus	Data and applications.	Information and navigation.
Data/ Information	Typically created and used by known and trusted sources. Properties generally known. Typically placed into system by users or known people and organizations. Typically organized in a meaningful fashion. A notion of private and shared data exists.	Full of unknown content. Source not always trusted. Often not placed onto the Web by users or known people and organizations. Highly variable organization. Privacy often suspect.
User Tasks	Install, configure, personalize, start, use, and upgrade programs. Open, use, and close data files. Fairly long times spent within an application. Familiarity with applications often achieved.	Link to a site, browse or read pages, fill out forms, register for services, participate in transactions, download and save things. Movement between pages and sites very rapid. Familiarity with many sites not established.
User's Conceptual Space	Controlled and constrained by program.	Infinite and generally unorganized.
Presentation Elements	Windows, menus, controls, data, toolbars, messages, and so on. Many transient, dynamically appearing and disappearing. Presented as specified by designer. Generally standardized by toolkits and style guides.	Two components — browser and page. Within page, any combination of text, images, audio, video, and animation. May not be presented as specified by the designer — dependent on browser, monitor, and user specifications. Little standardization.

(continued)

Table 2.2 *(continued)*

	GUI	WEB
Navigation	Through menus, lists, trees, dialogs, and wizards. Not a strong and visible concept. Constrained by design. Generally standardized by toolkits and style guides.	Through links, bookmarks, and typed URLs. Significant and highly visible concept. Few constraints, frequently causing a lost "sense of place." Few standards. Typically part of page design, fostering a lack of consistency.
Context	Enables maintenance of a better sense of context. Restricted navigation paths. Multiple viewable windows.	Poorer maintenance of a sense of context. Single-page entities. Unlimited navigation paths. Contextual clues become limited or are difficult to find.
Interaction	Interactions such as clicking menu choices, pressing buttons, selecting list choices, and cutting/copying/pasting occur within context of active program.	Basic interaction is a single click. This can cause extreme changes in context, which may not be noticed.
Response Time	Nearly instantaneous.	Quite variable, depending on transmission speeds, page content, and so on. Long times can upset the user.
Visual Style	Typically prescribed and constrained by toolkits. Visual creativity allowed but difficult. Little significant personalization.	Fosters a more artistic, individual, and unrestricted presentation style. Complicated by differing browser and display capabilities, and bandwidth limitations. Limited personalization available.
System Capability	Unlimited capability proportional to sophistication of hardware and software.	Limited by constraints imposed by the hardware, browser, software, client support, and user willingness to allow features because of response time, security, and privacy concerns.
Task Efficiency	Targeted to a specific audience with specific tasks. Limited only by the amount of programming undertaken to support it.	Limited by browser and network capabilities. Actual user audience usually not well understood. Often intended for everyone.

Table 2.2 *(continued)*

	GUI	WEB
Consistency	Major objective exists within and across applications. Aided by platform toolkit and design guidelines. Universal consistency in GUI products generally created through toolkits and design guidelines.	Sites tend to establish their own identity. Standards frequently set within a site. Frequent ignoring of GUI guidelines for identical components, especially controls.
User Assistance	Integral part of most systems and applications. Accessed through standard mechanisms. Documentation, both online and offline, usually provided. Personal support desk also usually provided.	No similar help systems. The little available help is built into the page. Customer service support, if provided, oriented to product or service offered.
Integration	Seamless integration of all applications into the platform environment a major objective. Toolkits and components are key elements in accomplishing this objective.	Apparent for some basic functions within most Web sites (navigation, printing, and so on). Sites tend to achieve individual distinction rather than integration.
Security	Tightly controlled, proportional to degree of willingness to invest resources and effort. Not an issue for most home PC users.	Renowned for security exposures. Browser-provided security options typically not understood by average users. When employed, may have function-limiting side effects.
Reliability	Tightly controlled in business systems, proportional to degree of willingness to invest resources and effort.	Susceptible to disruptions caused by user, telephone line and cable providers, Internet service providers, hosting servers, and remotely accessed sites.

User's conceptual space. In a *GUI* environment the user's conceptual space is controlled by the program and application. A user's access to data is constrained, and space is made available where their data can be stored and managed. A *Web* user's space is infinite and generally unorganized. Little opportunity for meaningful organization of personal information exists.

Presentation elements. The main presentation elements for *GUIs* are various kinds of windows, menus, controls, toolbars, messages, and data. Many elements are transient, dynamically appearing and disappearing based upon the current context

of the interaction. They are also generally standardized as a result of the toolkits and style guides used. Elements are presented on screens exactly as specified by the designer. *Web* systems possess two components: the browser and page. Many browsers are substantially GUI applications with traditional GUI presentation elements. Within a page itself, however, any combination of text, images, audio, video, and animation may exist. Complex, cluttered, and visually distracting pages are easy to generate and often exist. This occurs because many designers have focused on implementing that which is new, pretty, or attention getting, with little thought given to usability. Interface style guides and guidelines to aid the design process are not known (or are ignored). Common toolkits and industry conventions, however, are now being proposed and will be slowly adopted. Also contributing to page design problems is the fact that a page may not be presented exactly as specified by the designer. The exact look of a page depends on the browser and monitor being used. Extreme variations in screen sizes for presenting pages exist. The user can change the look of a page by modifying its properties.

Navigation. *GUI* users navigate through structured menus, lists, trees, dialogs, and wizards. Paths are constrained by design (grayed out menu choices, for example), and the navigation mechanisms are standardized by toolkits and style guides. Navigation is a weakly established concept, and is a supplement to more important task functions and actions. Some aspects of a GUI provide a strong sense of navigation: the ellipsis on "to another window" indicators such as "Open...," command buttons such as "OK" and "Cancel" that direct the user's focus to another window, and wizards. Other aspects of GUI design do not provide a strong sense of navigation — pressing an Apply button, for example, does not result in something visible happening.

Web users control their own navigation through links, bookmarks, and typed URLs. Navigation is a significant and highly visible concept with few constraints. The immense size of the Web — and the user's ability to easily wander just about anywhere — frequently causes a lost "sense of place," or "Where am I right now?" feeling. Web navigation has few standards beyond the browser's Back button and underlined links. Typically most navigation is part of page design that fosters a lack of consistency, and often confuses users. Establishing a continual sense of place for the user is a critical aspect of Web page design.

Context. *GUI* systems enable the user to maintain a better sense of context. Paths are restricted, and multiple overlapping windows may be presented and visible, enabling users to remember how what they are doing fits into the overall task picture. *Web* pages are single entities with almost unlimited navigation paths. They do not bring up separate dialog boxes to ask questions, provide or request supplemental information, or present messages. Contextual clues become limited or are hard to find.

Interaction. *GUI* interactions consist of activities such as clicking menu choices, pressing buttons, selecting choices from a list, keying data, and cutting, copying, or pasting within context established by an open window and an active program. The basic *Web* interaction is a single click. This click can cause extreme changes in context such as moving to another site or changing the displayed information

within a site. The user may not notice subtle changes when they occur. Additionally, the browser provides parallel mechanisms such as the Back button that may function differently depending on context. The distinction between an action and a navigation link is not always obvious.

Response time. Compared to the Web, response times with a *GUI* system are fairly stable, if not nearly instantaneous. *Web* response times can be variable, and often aggravatingly slow. Line transmission speeds, system loads, and page content can have a dramatic impact. Long response times can upset and frustrate users.

Visual style. In *GUI* systems, the visual style is typically prescribed and constrained by toolkits. (Exceptions are entertainment and multimedia applications.) Visual creativity in screen design is allowed but it is difficult to do. While some user options and style choices do exist, little opportunity exists for screen personalization. In *Web* page design, a more artistic, individual, and unrestricted presentation style is allowed and encouraged. Much design freedom exists, but differing browser and display capabilities, multiple screen sizes, and bandwidth limitations, often complicate and restrict this freedom. Limited personalization of the system is available, at a browser or site level, for users.

System capability. *GUI* system capabilities are limited only in proportion to the capability of the hardware in terms of speed, memory, and configuration, and the sophistication of the software. The *Web* is more constrained, because it is limited by constraints imposed by the hardware, browser, and software. It is also limited by the willingness of the page owner to provide certain functions and elements, and the willingness of the user to allow features because of response time, security, and privacy issues and concerns.

Task efficiency. *GUI* systems are targeted to a specific audience performing specific tasks. Generally, the efficiency of performing a task is limited only by the amount of programming undertaken to support it. Browser and network capabilities limit *Web* task efficiency. The actual user audience is usually not well understood because many Web sites are intended for everyone.

Consistency. Consistency in *GUI* system design is a major objective in most development efforts. While they are far from perfect, an attempt is made to be consistent both within applications and across applications. Many organizations possess interface and screen design standards and toolkits to aid in the standardization process. Toolkits and guidelines also allow a certain degree of universal consistency in GUI products. In *Web* page design, the heavy emphasis on graphics, a lack of design standards, and the desire of Web sites to establish their own identities results in very little consistency across sites. Web sites often establish standards within a site, but in too many instances developers ignore guidelines existing for GUI components used in Web pages. These problems are found especially in the presentation of screen controls on pages.

User assistance. User assistance is an integral part of most *GUI* systems applications. Users access this assistance through standard mechanisms such as the F1 key and Help menus. Message and status areas are also provided on the screen. Documentation, both online and offline, is normally provided, as is a support desk to answer user questions and provide guidance and assistance. *Web* pages

do not yet provide similar help systems. What little help that is available is built into the page. Customer service support, if provided, is generally oriented to the product or service offered. GUI browsers may provide GUI-type assistance, so the user sees two different assistance approaches. Deficiencies in Web page help then become more obvious.

Integration. A primary goal of most *GUI* applications is the seamless integration of all pieces. Common functions are supported across applications and import/export capabilities exist. Again, toolkits and their components are key elements in accomplishing this objective. In *Web* design, some integration is apparent within a site for basic functions such as navigation and printing. But because sites strive for individual distinction, interoperability between sites is almost nonexistent.

Security. In a *GUI* environment, security and data access can be tightly controlled, in proportion to the degree of willingness of an organization to invest resources and effort. For home applications, security is not an issue for most PC users. The *Web* is renowned for security exposures. This is a major inhibitor of Web use for both businesses and consumers. Browser-provided security options have typically not been well understood by average Web users. When employed, these security options often have function-limiting side effects (such as disabled cookies). Attempts to create a more trustworthy appearance are being made through the use of security levels and passwords to assure users that the Web is a secure environment.

Reliability. Like security, reliability in *GUI* systems is established and controlled in proportion to the degree of willingness of an organization to invest resources and effort. The computer being used influences reliability as does, if applicable, the local area network. Both are in the control of the using organization. *Web* reliability is susceptible to disruptions from many directions. Telephone line and cable providers, Internet service providers, hosting servers, and remotely accessed sites all can contribute to the problem. Accessed applications and user mistakes may also cause reliability problems. A lack of reliability can be a great inhibitor of Web use.

In conclusion, from a design implication perspective, GUI and Web differences can be extensive. Today these differences must be considered in Web design, although many GUI interface design techniques and guidelines are applicable in Web design. In the future many of these GUI-Web differences will diminish or disappear as the discrepancies are addressed by technology.

In developing a Web system, always evaluate each GUI guideline for direct applicability in any development effort. Also, do not simply transport an entire GUI application or design to the Web without evaluating it in terms of the implications described earlier. Some applications or designs may require significant changes, others a simple fine-tuning. One so far unmentioned aspect that both GUI and Web systems do have in common is "Know your user." Involving them throughout the redesign process will ensure the best transition to the Web. (More about knowing your users follows in Step 1.)

Printed Pages versus Web Pages

While Internet history spans a couple of decades, that of the printed page measures more than five and one-half centuries. Research and experience with printed pages

through these centuries has created a fundamental and accepted set of guidelines for editorial style, element presentation, and text organization. Many of the basic guidelines, clear, comprehensive, and consistent, can and are being applied to Web page design. Web page design, however, is different in many aspects from the design of books, documents, newspapers, and other similar materials. These differences require rethinking, researching, and reformulating many of these guidelines for use in Web page design. Many of these differences have already been identified. Others will surface as Web experience grows and research is conducted. In the following paragraphs, the major differences between print and Web page design are briefly described. Implications for Web page design are also summarized.

Page size. Printed pages are generally larger than their Web counterparts. They are also fixed in size, not variable like Web pages. A printed page can be designed as one entity, the designer being assured that the final product will possess an integrated and complete look. Although Web pages are usually designed as a complete entity, they are presented in pieces with dimensions that differ depending on the user's technology (browser, monitor, and so on). The visual impact of the printed page is maintained in hard-copy form, while on the Web all that usually exists are snapshots of page areas. The visual impact of a Web page is substantially degraded, and the user may never see some parts of the page because their existence is not known or requires scrolling to bring into view. The design implications: The top of a Web page is its most important element, and signals that parts of a page lie below the surface must always be provided to the user.

Page rendering. Printed pages are immensely superior to Web pages in rendering. Printed pages are presented as complete entities, and their entire contents are available for reading or review immediately upon appearance. Web page elements are often rendered slowly, depending upon things like line transmission speeds and page content. Dozens of seconds may be consumed waiting for a page to completely appear. Impatient users may not wait and move on to somewhere else. Design implications: Provide page content that downloads fast, and give people elements to read immediately so the sense of passing time is diminished. (The ultimate goal: a bandwidth fast enough to download a Web page as fast as one can turn a paper page.)

Page layout. With the printed page, layout is precise with much attention given to it. With Web pages, layout is more of an approximation, and is negatively influenced by deficiencies in design toolkits and the characteristics of the user's browser and hardware, particularly screen sizes. Design implication: Understand the restrictions and design for the most common user tools.

Page resolution. Today the resolution of displayed print characters still exceeds that of screen characters, and screen reading is still slower than reading from a document. Design implication: Provide an easy way to print long Web documents. (The ultimate goal: a screen resolution sharp enough to render type crisply enough so that screen reading speed reaches that of newspaper reading.)

User focus. Printed pages present people with entire sets of information. Estimations of effort needed to deal with the document are fairly easily made, with size and the nature of the material being strong contributors. Some printed pages may be read sequentially (a novel) and others partially (a newspaper) and somewhat sequentially (the sports section first, perhaps?). Other forms of printed material

may simply be skimmed (a sales brochure), but this skimming is usually systematic in some way. Web pages present people with individual snapshots of information, often with few clues about structure and sequence, and rarely with a few cues about length and depth. People also have a sense that the body of Web information potentially available is unlimited, and that information paths can lead anywhere. With few content size cues available and a huge information base, a common resulting behavior of Web users is to skim the information presented and look for what is most relevant to their task or need. This is done for personal efficiency and so as not to tax one's patience. Design implications: Create easy-to-scan pages and limit the word count of textual content. Also, provide overviews of information organization schemes, clear descriptions of where links lead, and estimations of sizes of linked pages and materials.

Page navigation. Navigating printed materials is as simple as page turning. It is a motor skill learned early in life and never thought of as navigation or a design element. Substantial interaction between pages is rare because the process is essentially sequential. Navigating the Web requires innumerable decisions concerning which of many possible links should be followed. It requires asking oneself questions such as these: What is at the end of this link? Where is it? Will it address my need or solve my problem? Design implications are similar to the aforementioned: Provide overviews of information organization schemes and clear descriptions of where links lead.

Sense of place. With paper documents, you derive a sense of where you are through a mixture of graphic and editorial organization, and size cues supplied by the design of the document. The document is an object with physical characteristics. Paging through printed material is an orderly process, sequential and understandable. Electronic documents provide none of these physical cues. All that is visible is a small collection of text, graphics, and links hinting that much else lies *somewhere* underneath. Moving through the Web links can cause radical shifts in location and context. Paging using the browser's Back button steps one back through links visited and may involve passing through different documents. Fixed locations that provide cues to support one's memory concerning the location of things are nonexistent. All these factors cause a person to easily lose a sense of place and lead to confusion. Design implication: Build cues into Web pages to aid the user in maintaining a sense of place.

Interactivity. Printed page design involves letting the eyes traverse static information, selectively looking at information and using spatial combinations to make page elements enhance and explain each other. Web design involves letting the hands move the information (scrolling, pointing, expanding, clicking, and so on) in conjunction with the eyes. Information relationships, static or dynamic, are expressed chronologically as part of the interaction and user movements. Doing is more memorable and makes a stronger impact than simply seeing. No print precedents exist for this style of interaction. A better understanding of this process (as well as better hardware and software) is needed to enhance interactivity.

Page independence. Because moving between Web pages is so easy, and almost any page in a site can be accessed from anywhere else, pages must be made freestanding. Every page is independent, and its topic and contents must be explained

without assumptions about any previous page seen by the user. Printed pages, which are sequential, fairly standardized in organization, and provide a clear sense of place, are not considered independent. Specific types of content (table of contents, author, index, and so on) are easily found in well-established document locations. Design implication: Provide informative headers and footers on each Web page.

In conclusion, many of the basic print guidelines can be applied to Web page design. As shown earlier, however, printed material design differs from Web page design in many aspects. New guidelines must continue to be developed, implemented, and modified as technology advances and our understanding of Web interaction increases. For the moment, apply existing guidelines where relevant, and new guidelines as necessary. Part 2 of this book describes many of these guidelines. What must be avoided are things that made sense in the print world, but do not meet today's needs in Web interface design.

The Merging of Graphical Business Systems and the Web

The strength of the Web lies in its capability of hosting applications, linking databases, and the processing occurring on a variety of machines within a company or organization. Within a closed system, queries against databases can be made, internal communication performed, and information useful to employees can be made available. Current systems can also be implemented with more traditional GUI interfaces. Graphical business systems and the Web are merging into a common entity. These Web systems are called intranets.

Characteristics of an Intranet versus the Internet

An intranet has many of the same characteristics as the Internet. They differ, however, in some important ways. The following discussion is partly based upon Nielsen (1997b):

Users. The users of intranets, being organization employees, know a lot about the organization, its structure, its products, its jargon, and its culture. Internet sites are used by customers and others who know much less about the organization, and often care less about it. The intranet user's characteristics and needs can be much more specifically defined than those of the general Internet user.

Tasks. An intranet is used for an organization's everyday activities, including complex transactions, queries, and communications. The Internet is mainly used to find information, with a supplementary use being simple transactions.

Type of information. An intranet contains detailed information needed for organizational functioning. Information is often be added or modified. The Internet usually presents more stable information: marketing and customer or client information, reports, and so forth.

Amount of information. Typically, an intranet site is much larger than an organization's Internet site. Massive amounts of information and processes seem to be

needed to make an organization function. It has been estimated that an intranet site can be ten to one hundred times larger than its corresponding public site.

Hardware and software. Because intranets exist in a controlled environment, the kinds of computers, monitors, browsers, and other software can be restricted or standardized. The need for cross-platform compatibility is minimized or eliminated, permitting more predictable design. Upgraded communications also permit intranets to run from a hundred to a thousand times faster than typical Internet access can. This allows the use of rich graphics and multimedia, screen elements that contribute to very slow download times for most Internet users.

Design philosophy. Implementation on the intranet of current text-based and GUI applications will present a user model similar to those that have existed in other domains. This will cause a swing back to more traditional GUI designs — designs that will also incorporate the visual appeal of the Web, but eliminate many of its useless, promotional, and distracting features. The resulting GUI hybrids will be richer and much more effective.

Some specific intranet design guidelines are discussed in Part 2 of this book.

Extranets

An extranet is a special set of intranet Web pages that can be accessed from outside an organization or company. Typical examples include those for letting customers check on an order's status or letting suppliers view requests for proposals. An extranet is a blend of the public Internet and the intranet, and its design should reflect this. Some specific extranet design guidelines are also discussed in Part 2.

Web Page versus Application Design

As previously mentioned, the dividing line between page and application design is not always clear. In general a Web page's design intent is mostly to provide information. An application is designed to let a person do and save something. Fowler and Stanwick (2004), upon whom the following discussion is entirely based, suggest that the difference can be presented in a continuum, anchored at one end by a Web page and at the other by a Web application. User expectations and activities, and the objective of the program and system determine where on the Web page–application continuum a particular design falls. As an aid to understanding where a particular design will reside, Fowler and Stanwick say the following system dimensions should be determined.

What is the nature of the relationship with the user?

It is probably an application if:

- Users must use the program.
- Users must identify themselves or log in to be able to do anything.
- The program must be reliable, and system failure will be immediately noticed.
- Work is remembered from earlier sessions.

It is probably a Web page if:

- The system does not know or care who the users are.
- Login is required to simply give access to more information than would be received anonymously.
- System failure may not be noticed.

It is in the middle of the continuum if:

- Minimal information such as a credit card number or address is remembered.

What is the conversation style?

It is probably an application if:

- The style is formal.

It is probably a Web page if:

- The style is informal, causal, and generic.

It is in the middle of the continuum if:

- The style is polite but friendly.

What is the nature of the interaction?

It is probably an application if:

- A large amount of data is being entered.
- Complex tasks are being performed.
- Data is being created, manipulated, and permanently stored.

It is probably a Web page if:

- No data is entered or changed.

It is in the middle of the continuum if:

- A small amount of data is entered and possibly stored.
- Milestones, checkpoints, and endpoints are expected and included.

What is the frequency of use?

It is probably an application if:

- It is used constantly or daily.
- It is used for long periods, such as four to eight hours.
- It is used to help resolve emergencies.

It is probably a Web page if:

- It is used only occasionally or erratically.
- It is used for only a few minutes at a time.
- It is used to find out about something.

It is in the middle of the continuum if:

- It is used periodically or episodically.

What is the perceived distance of the provider?

It is probably an application if:

- It is viewed as being local in origin.
- It is viewed as being controlled locally by a data administrator.
- The response time is fast.

It is probably a Web page if:

- The origin is unknown.
- The origin is viewed as originating somewhere else in the country or overseas.
- The response time is slow.

It is in the middle of the continuum if:

- The origin cannot be ascertained or doesn't matter.

Are the interactions in real time?

It is probably an application if:

- Data is fed in real time.
- The information is critical.
- A delay is life-threatening.
- Long response delays exist but users remain.

It is probably a Web page if:

- Time is irrelevant.
- Long response delays exist and cause users to exit.

It is in the middle of the continuum if:

- Response is near real time.

How much help will users need?

It is probably an application if:

- Intense training programs and experience are needed to use and become experts.

It is probably a Web page if:

- Every visit is a one-time session, minimizing or eliminating the need for help.

It is in the middle of the continuum if:

- A minimum amount of experience, training, or help material is needed.

What is the interaction style?

It is probably an application if:

- The navigation is controlled.
- Controls are complex.
- The syntax is object:action.

- There is little or no reversibility.
- Drag-and-drop manipulation is expected.
- Exit requires users to log off.

It is probably a Web page if:

- The navigation is flexible and simple.
- Controls are simple.
- No object selection is required.
- Actions are easily reversed using the Back button.
- No drag-and-drop manipulation exists.
- Single-clicking links are used to navigate.

It is in the middle of the continuum if:

- The navigation is flexible.
- Simple data collection exists.
- User exits by closing the Web browser.

What is the presentation style?

It is probably an application if:

- The style is subdued and serious.

It is probably a Web page if:

- The style is colorful, graphic, and possibly animated.
- The controls are obvious and self-explanatory.

It is in the middle of the continuum if:

- The style is cooler but attractive.

What, if any, standards are followed?

It is probably an application if:

- Platform standards compliance exists.
- The style resembles or matches GUI standards.
- It is used with other applications that set expectations.

It is probably a Web page if:

- Few cross-site standards are followed.
- Only intra-site consistency exists.

It is in the middle of the continuum if:

- Some common patterns exist.

Because Web pages and Web applications cover a range of interaction and visual styles, there is no one set of design standards that cover all possible uses. The best solution is to use existing GUI or Web guidelines or standards that fit the particular design situation: application, a mix of both, or a Web page. Keep in mind, however, that many interface design guidelines are applicable across the entire continuum.

Principles of User Interface Design

An interface must really be just an extension of a person. This means that the system and its software must reflect a person's capabilities and respond to his or her specific needs. It should be useful, accomplishing some business objectives faster and more efficiently than the previously used method or tool. It must also be easy to learn, because people want to do, not learn to do. Finally, the system must be easy and fun to use, and evoke a sense of pleasure and accomplishment, not tedium and frustration.

The interface should serve as both a connector and a separator: a connector in that it ties the user to the power of the computer, and a separator in that it minimizes the possibility of the participants damaging one another. While the damage the user inflicts on the computer tends to be physical (a frustrated pounding of the keyboard), the damage caused by the computer is more psychological (a threat to one's self-esteem).

Throughout the history of the human-computer interface, various researchers and writers have attempted to define a set of general principles of interface design. What follows is a compilation of these principles. They reflect not only what we know today, but also what we think we know today. Many are based on research, others on the collective thinking of behaviorists working with user interfaces. These principles will continue to evolve, expand, and be refined as our experience with GUIs and the Web increases. We will begin with the first set of published principles, which are for the Xerox STAR.

Principles for the Xerox STAR

The design of the Xerox STAR was guided by a set of principles that evolved over its lengthy development process (Smith, et al., 1982; Verplank, 1988). These principles established the foundation for graphical interfaces and are as follows:

The illusion of manipulable objects. Displayed objects that are selectable and manipulable must be created. A design challenge is to invent a set of displayable objects that are represented meaningfully and appropriately for the intended application. It must be clear that these objects can be selected, and how to select them must be self-evident. When they are selected should also be obvious, because it should be clear that the selected object will be the focus of the next action. Verplank called this "graphics with handles on it." Stand-alone icons easily fulfilled this requirement. The handles for windows were placed in the borders (window-specific commands, pop-up menus, scroll bars, and so on).

Visual order and viewer focus. Attention must be drawn, at the proper time, to the important and relevant elements of the display. Effective visual contrast between various components of the screen is used to achieve this goal (STAR was monochromatic, so color was not used). Animation is also used to draw attention, as is sound. Feedback must also be provided to the user. Because the pointer is usually the focus of viewer attention, it is a useful mechanism for providing this feedback (by changing shapes).

Revealed structure. The distance between one's intention and the effect must be minimized. Most often, the distance between intention and effect is lengthened

as system power increases. The relationship between intention and effect must be tightened and made as apparent as possible to the user. The underlying structure is often revealed during the selection process.

Consistency. Consistency aids learning. Consistency is provided in such areas as element location; grammar; font shapes, styles, and sizes; selection indicators; and contrast and emphasis techniques.

Appropriate effect or emotional impact. The interface must provide the appropriate emotional effect for the product and its market. Is it a corporate, professional, and secure business system? Should it reflect the fantasy, wizardry, and bad puns of computer games?

A match with the medium. The interface must also reflect the capabilities of the device on which it will be displayed. Quality of screen images will be greatly affected by a device's resolution and color-generation capabilities.

General Principles

The design goals in creating a user interface are described in the following section. They are fundamental to the design and implementation of all effective interfaces, including both GUI and Web. These principles are general characteristics of the interface, and they apply to all aspects. Specific guidelines on how to implement many of these goals are presented in Part 2. These guidelines are presented alphabetically, and not in order of importance. They are derived from the various principles described in Galitz (1992), IBM (1991, 2001), Lidwell et al. (2003), Mayhew (1992), Microsoft (1992, 1995, 2001), Norman (1988), Open Software Foundation (1993), Verplank (1988), and the World Wide Web Consortium (2001).

Accessibility

- Systems should be designed to be usable, without modification, by as many people as possible.

A system should be usable by people of diverse abilities, without special design or modification. Originally, the term *accessibility* in design was directed toward making a system usable for people with disabilities. Recently, it has become obvious that accommodations for people with disabilities could benefit all users. So, the definition of accessibility has been expanded to cover all users of systems.

Four characteristics of accessible design are perceptibility, operability, simplicity, and forgiveness. *Perceptibility* assures that a system's design can be perceived, regardless of a person's sensory abilities. *Operability* assures that a system's design can be used, regardless of a person's physical abilities. *Simplicity* assures that all users can easily understand and use the system, regardless of experience, literacy, or concentration level. *Forgiveness* assures that a system minimizes the occurrence of, and consequences of, errors. These concepts are described more thoroughly in the following paragraphs.

Aesthetically Pleasing

- Provide visual appeal by following these presentation and graphic design principles:
  — Provide meaningful contrast between screen elements.
  — Create groupings.
  — Align screen elements and groups.
  — Provide three-dimensional representation.
  — Use color and graphics effectively and simply.

A design aesthetic, or visually pleasing composition, is attractive to the eye. It draws attention subliminally, conveying a message clearly and quickly. Visual appeal makes a computer system accessible and inviting. A lack of visually pleasing composition is disorienting, obscures the intent and meaning, and slows down and confuses the user. Research has shown that people perceive more-aesthetic designs to be easier to use than less-aesthetic designs — whether they are easy or not (Kurosu and Kashimura, 1995). Additionally, good aesthetics have been found to create a positive attitude toward a design, to make people more tolerant of design problems, to aid creative thinking, and to aid problem solving (Norman, 2002). Visual appeal is terribly important today because most human-computer communication occurs in the visual realm. Visual appeal is provided by following the presentation and graphic design principles to be discussed, including providing meaningful contrast between screen elements, creating spatial groupings, aligning screen elements, providing three-dimensional representation, and using color and graphics effectively. Good design combines power, functionality, and simplicity with a pleasing appearance.

Availability

- Make all objects available at all times.
- Avoid the use of modes.

All aspects of a system should be available at any time and in any sequence. Avoid the use of modes, states of the interface in which normally available actions are no longer available. Modes restrict the ability of the user to interact with the system.

Clarity

- The interface should be visually, conceptually, and linguistically clear including:
  — Visual elements
  — Functions
  — Metaphors
  — Words and text

The interface must be clear in visual appearance, concept, and wording. Visual elements should be understandable, relating to the user's real-world concepts and functions. Metaphors, or analogies, should be realistic and simple. Interface words and text should be simple, unambiguous, and free of computer jargon.

Compatibility

- Provide compatibility with the following:
  — The user
  — The task and job
  — The product
- Adopt the user's perspective.

User compatibility. Design must be appropriate and compatible with the needs of the user or client. Effective design starts with understanding the user's needs and adopting the user's point of view. One very common error among designers is to assume that users are all alike. A glance around the office should quickly put this assumption to rest. Another common error is to assume that all users think, feel, and behave exactly like the developer. Studies have proven otherwise. Users have quite different needs, aspirations, and attitudes than developers. A system reflecting only the knowledge and attitudes of its designers cannot be successful. "Know the user" is *the* fundamental principle in interface design. User compatibility can happen only if understanding truly occurs.

Task and job compatibility. The organization of a system should match the tasks a person must do to perform the job. The structure and flow of functions should permit easy transition between tasks. The user must never be forced to navigate between applications or many screens to complete routine daily tasks.

Product compatibility. The intended user of a new system is often the user of other systems or earlier versions of the new system. Habits, expectations, and a level of knowledge have been established and will be brought to bear when learning the new system. If these habits, expectations, and knowledge cannot be applied to the new system, confusion results and learning requirements are greatly increased. While compatibility across products must always be considered in relation to improving interfaces, making new systems compatible with existing systems will take advantage of what users already know and reduce the necessity for new learning.

Configurability

- Permit easy personalization, configuration, and reconfiguration of settings to do the following:
  — Enhance a sense of control.
  — Encourage an active role in understanding.

The interface should be tailorable to individual users' needs and desires. Easy personalization and customization through configuration and reconfiguration of a system enhances a sense of control, encourages an active role in understanding, and allows for personal preferences and differences in experience levels. It also leads to higher user satisfaction.

Some people will prefer to personalize a system to better meet their preferences. Other people will not, accepting what is given. Still others will experiment with reconfiguration and then give up, running out of patience or time. For these latter groups of users, a good default configuration must be provided.

Consistency

- A system should look, act, and operate the same throughout. Similar components should:
  - — Have a similar look.
  - — Have similar uses.
  - — Operate similarly.
- The same action should always yield the same result.
- The function of elements should not change.
- The position of standard elements should not change.

Design consistency is the common thread that runs throughout these guidelines. Consistency is uniformity in appearance, placement, and behavior within the user interface. It is the cardinal rule of all design activities. Consistency is important because it can reduce requirements for human learning by allowing skills learned in one situation to be transferred to another like it. While any new system must impose some learning requirements on its users, it should avoid encumbering productive learning with nonproductive, unnecessary activity. Consistency also aids learning of the system's mental model.

In addition to increased learning requirements, inconsistency in design has several other prerequisites and by-products, including:

- More specialization by system users.
- Greater demand for higher skills.
- More preparation time and less production time.
- More frequent changes in procedures.
- More error-tolerant systems (because errors are more likely).
- More kinds of documentation.
- More time to find information in documents.
- More unlearning and learning when systems are changed.
- More demands on supervisors and managers.
- More things to do wrong.

Inconsistencies in design are caused by differences in people. Several designers might each design the same system differently. Inconsistencies also occur when those performing design activities are pressured by time constraints. All too often the solutions in those cases are exceptions that the user must learn to handle. People, however, perceive a system as a single entity. To them, it should look, act, and feel similar throughout. Excess learning requirements become a barrier to users achieving and maintaining high performance and can ultimately influence user acceptance of the system.

Can consistency make a big difference? One study found that user thinking time nearly doubled when the position of screen elements, such as titles and field captions, was varied on a series of menu screens.

Design consistency is achieved by developing and applying design standards or guidelines. In the late 1980s the computer industry and many using organizations finally awakened to the need for them, and a flurry of graphical user interface guideline documents were developed and published. These guidelines specify the appearance and behavior of the GUI. They describe the windows, menus, and various controls available, including what they look like and how they work. They also provide some guidance on when to use the various components.

Examples of industry-produced guidelines include Apple's *Macintosh Human Interface Guidelines* (1992b), Digital Equipment Corporation's *XUI Style Guide* (1988), IBM's *System Application Architecture Common User Access* (1987, 1989a, 1989b, 1991), IBM's *Object-Oriented Interface Design: IBM Common User Access Guidelines* (1992), Sun Microsystem's *OPEN LOOK: Graphical User Interface Application Style Guidelines* (1990), Open Software Foundation's *OSF/MOTIF Style Guide* (1993), and Microsoft's *The Windows Interface: An Application Design Guide* (1992) and *The Windows Interface Guidelines for Software Design* (1995).

The Web has burst upon the scene with few standards and guidelines to direct design. Many GUI and printed material principles are applicable but they have been applied in a haphazard manner. New research-based guidelines are desperately needed.

Organizations working on traditional interface guidelines or standards include the International Standards Organization (ISO), the American National Standards Institute (ANSI), and the Human Factors and Ergonomics Society (Billingsley, 1996). The development of Web design guidelines has been one focus of the World Wide Web Consortium (2001).

Control

- The user must control the interaction.
  - — Actions should result from explicit user requests.
  - — Actions should be performed quickly.
  - — Actions should be capable of interruption or termination.
  - — The user should never be interrupted for errors.
- The context maintained must be from the perspective of the user.
- The means to achieve goals should be flexible and compatible with the user's skills, experiences, habits, and preferences.
- Avoid modes because they constrain the actions available to the user.
- Permit the user to customize aspects of the interface, while always providing a proper set of defaults.

People should be able to exercise control over what the system does. Control is feeling in charge, feeling that the system is responding to your actions. Feeling that a machine is controlling you is demoralizing and frustrating. The interface should present a tool-like appearance. Control is achieved when a person, working at his or her own pace, is able to determine what to do, how to do it, and then is easily able to get it

done. Simple, predictable, consistent, flexible, customizable, and passive interfaces provide control. Lack of control is signaled by unavailable systems, long delays in system responses, surprising system actions, tedious and long procedures that cannot be circumvented, difficulties in obtaining necessary information, and the inability to achieve the desired results. The feeling of control has been found to be an excellent mitigator of the work stress associated with many automated systems.

The means to achieve goals should be flexible and compatible with the user's needs, including skills, experiences, habits, and preferences. A system can accommodate varying needs by offering multiple ways to accomplish a task. New users benefit from structured interactions that provide minimal choices, prompts, constraints, and easy access to help. Experts want less-structured interactions that provide direct access to functions and bypass the support provided to beginners.

In general, avoid modes because they restrict the actions available to the user at any given time. If modes must be used, they should be visually obvious (for example, a changed mouse pointer shape). Allow the user to customize aspects of the interface, while always providing a proper set of defaults.

Directness

- Provide direct ways to accomplish tasks.
  - — Available alternatives should be visible.
  - — The effect of actions on objects should be visible.

Tasks should be performed directly. Available alternatives should be visible, reducing the user's mental workload. Directness is also best provided by the object:action sequence of direct-manipulation systems. Tasks are performed by directly selecting an object, selecting an action to be performed, and then seeing the action being performed.

Efficiency

- Minimize eye and hand movements, and other control actions.
  - — Transitions between various system controls should flow easily and freely.
  - — Navigation paths should be as short as possible.
  - — Eye movement through a screen should be obvious and sequential.
- Anticipate the user's wants and needs whenever possible.

Eye and hand movements must not be wasted. One's attention must be captured by relevant elements of the screen when needed. Sequential eye movements between screen elements should be predictable, obvious, and short. Web pages must be easily scannable. All navigation paths should be as short as possible. Manual transitions between various system controls should also be as short as possible. Avoid frequent transitions between input devices such as the keyboard and mouse.

Always try to anticipate the user's wants and needs. At each step in a process, present to the user all the information and tools needed to complete the process. Do not require the user to search for and gather necessary information and tools.

Familiarity

- Employ familiar concepts and use a language that is familiar to the user.
- Keep the interface natural, mimicking the user's behavior patterns.
- Use real-world metaphors.

Build on the user's existing knowledge, especially that they have gained from experience in the real world. Build into the interface concepts, terminology, workflows, and spatial arrangements that are already familiar to the user. Operations should mimic one's behavior patterns; dialogs should mimic one's thought processes and vocabulary. Icons or images should look like the real-world objects they represent. Familiar concepts enable people to get started and become productive quickly.

Flexibility

- A system must be sensitive to the differing needs of its users, enabling a level and type of performance based upon:
  - — Each user's knowledge and skills.
  - — Each user's experience.
  - — Each user's personal preference.
  - — Each user's habits.
  - — The conditions at that moment.

Flexibility is the system's ability to respond to individual differences in people. Permit people to choose the method of interaction that is most appropriate to their situation. People should be able to interact with a system in terms of their own particular needs including knowledge, experience, and personal preference. Flexibility is accomplished by providing multiple ways to access application functions and perform tasks. It is also accomplished through permitting system customization. Another benefit of flexibility is that it contributes to increased user control. A flexible system is a versatile system.

Flexibility is not without dangers. Generally as flexibility increases, a system's usability decreases. Highly flexible systems can confuse inexperienced people, causing them to make more errors. For this reason, flexibility appears desirable only for experienced users. Novice users should not be exposed to system flexibility at the start, but only as they gain experience. The concept of progressive disclosure, to be discussed in the simplicity guideline to follow, is also applicable here.

Another problem with flexibility is that it may not always be used; some people prefer to continue doing things in the way they first learned. Many factors may account for this including an unwillingness to invest in additional learning, or perhaps new ways may not be obvious. The former problem may be addressed by making the new ways as easy and safe to learn as possible, the latter by including in training and reference materials not only information about how to do things, but also when they are likely to be useful.

Forgiveness

- Tolerate and forgive common and unavoidable human errors.
- Prevent errors from occurring whenever possible.
- Protect against possible catastrophic errors.
- Provide constructive messages when an error does occur.

It is often said "to err is human." The corollary to that statement, at least in computer systems, might be "to forgive is good design." People will make mistakes; a system should tolerate those that are common and unavoidable. A forgiving system keeps people out of trouble.

People like to explore and learn by trial and error. A system oversensitive to erroneous inputs will discourage users from exploring and trying new things. Learning will be inhibited, and people will be overcautious, working slowly and carefully to avoid mistakes. Productivity will then suffer. A fear of making a mistake and not being able to recover from it has always been a primary contributor to fear of dealing with computers.

Prevent errors from occurring by anticipating where mistakes may occur and designing to prevent them. Permit people to review, change, and undo actions whenever necessary. Make it very difficult to perform actions that can have tragic results. When errors do occur, present clear instructions on how to correct them.

Immersion

- Foster immersion within tasks.

Immersion is a state of mental focus so intense that awareness and sense of the "real world" is lost. When immersion exists, the general result is a feeling of joy and satisfaction (Lidwell, et al., 2003). When a person's perceptual and cognitive systems are underutilized, one can become bored and apathetic. When overtaxed, stress and frustration can result. Immersion is fostered by one or more of the following conditions being present (Csikszentmihalyi 1991, in Lidwell, et al., 2003):

- Challenges that can be overcome.
- Context where a person can focus without significant distraction.
- Clearly defined goals.
- Immediate feedback about actions and overall performance.
- A feeling of control over actions, activities, and the environment.

Immersion is characterized by the following:

- A loss of awareness of the worries and frustrations of everyday life.
- A loss of concern regarding matters of self (e.g., awareness of hunger or thirst).
- A modified sense of time (e.g., hours pass in what seems like minutes).

In design, provide conditions that foster immersion: challenges, clearly defined goals, a feeling of control, and feedback. Also, minimize visual and auditory distractions in the interface, as well as in the using environment.

Obviousness

- A system should be easily learned and understood. A user should know the following:
  — What to look at
  — What it is
  — What to do
  — When to do it
  — Where to do it
  — Why to do it
  — How to do it
- The flow of actions, responses, visual presentations, and information should be in a sensible order that is easy to recollect and place in context.

All objects and controls should be visible and intuitive. Their functions should be identifiable. The design of a control should suggest how it is to be operated. This suggestibility is often referred to as *affordance*. When the design of an object or control has high affordance, its use being obvious, it is easy for people to know how to interact with it.

A system should be understandable, flowing in a comprehensible, obvious, and meaningful order. Strong clues to the operation of objects should be presented. The steps to complete a task should be obvious. Reading and digesting long explanations should never be necessary.

Operability

- Ensure that a system's design can be used by everyone, regardless of physical abilities.

Operability requires that a system always be usable, regardless of a person's physical abilities. Operability is achieved by minimizing repetitive actions and sustained physical effort, fostering control use through making their intents obvious and their sizes large enough for easy activation, and positioning screen information and controls so they can be easily accessed whether sitting or standing. Operability also requires that the design be compatible with assistive technologies.

Perceptibility

- Assure that a system's design can be perceived, regardless of a person's sensory abilities.

Every user must be able to perceive a design, regardless of sensory abilities. Designing for perceptibility involves using redundant coding methods (e.g., iconic as well as textual presentation, color as well as monochromatic presentation). Also provide compatibility with assistive sensory technologies.

Positive First Impression

The use of many of today's computer systems is discretionary in nature. This is especially true for Web sites. A person's initial impression of a system greatly influences subsequent perceptions and attitudes, and the quality of subsequent interactions. This impression is often formed at the entry point into the system. A poor first impression may cause annoyance or even cause immediate system rejection. Lidwell et al. (2003) suggests that the key elements of the entry point are minimal barriers, points of prospect, and progressive lures.

Minimal barriers. Barriers to entry should not encumber entry points. Barriers may be functional or aesthetic. Functional barriers include slow-loading pages, excessive advertisements, illegible typefaces, and confusing screen organization. Aesthetic barriers include visually noisy screens and overuse of color.

Points of prospect. Users should become quickly oriented and enabled to clearly survey available options. Screens or pages should be easily scannable, provide good orientation cues, and clear navigation options. Options should be capable of being easily reviewed with minimal distractions or disruptions, such as those caused by animated screen elements.

Progressive lures. Lures should exist to pull people through the entry point. Progressive lures can be compelling headlines, pictures of popular products, or obvious links to other interesting destinations. Progressive lures get people to incrementally approach, enter, and move through the system or page entry point.

Predictability

- The user should be able to anticipate the natural progression of each task.
  - — Provide distinct and recognizable screen elements.
  - — Provide cues to the result of an action to be performed.
- Do not bundle or combine actions.
- All expectations should be fulfilled uniformly and completely.

Tasks, displays, and movement through a system should be anticipatable based on the user's previous knowledge or experience. All actions should lead to results the user expects. Screen elements should be distinct and recognizable. Current operations should provide clues as to what will come next. Anticipation, or predictability, reduces mistakes and enables the user to complete tasks more quickly. All expectations possessed by the user should be fulfilled uniformly and completely. Predictability is greatly enhanced by design consistency.

Avoid bundling or combining actions because the user may not be able to anticipate the side effect. Instead of implementing composite actions, make actions independent. Provide ways for users to combine actions when they wish to do so.

Recovery

- A system should permit:
  — Commands or actions to be abolished or reversed.
  — Immediate return to a certain point if difficulties arise.
- Ensure that users never lose their work as a result of:
  — An error on their part.
  — Hardware, software, or communication problems.

A person should be able to retract or reverse an action by issuing an *undo* command. Knowing that an action can be reversed reduces much of the distress of new users, who often worry about doing something wrong. The return point could be the previous action, previous screen, a recent closure point, or the beginning of some predetermined period, such as back 10 screens or some number of minutes. Reversing or abolishing an action is analogous to using an eraser to eliminate a pencil mark on a piece of paper.

The goal is stability, or returning easily to the right track when a wrong track has been taken. Recovery should be obvious, automatic, and easy and natural to perform. In short, it should be hard to get into deep water or go too far astray. Easy recovery from an action greatly facilitates learning by trial and error and exploration. If an action is not reversible, and its consequences are critical, it should be made difficult to accomplish. Always ensure that users never lose their work as a result of their own errors or technical glitches.

Responsiveness

- The system must rapidly respond to the user's requests.
- Provide immediate acknowledgment for all user actions:
  — Visual.
  — Textual.
  — Auditory.

A user request must be responded to quickly. Knowledge of results, or feedback, is a necessary learning ingredient. It shapes human performance and instills confidence. All requests to the system must be acknowledged in some way. Feedback may be visual, such as a change in the shape of the mouse pointer, or textual, such as text taking the form of a message. It may also be auditory, consisting of a unique sound or tone.

Never leave the screen blank for more than a moment, because the user may think the system has failed. If a request requires an unusually long processing time, or one that is longer than customary, provide an interim "in-progress" message. Also provide some unique form of communication if a user action results in a problem or possible problem.

Substantial or more informative feedback is most important for the casual or new system user. Expert users are often content to receive more modest feedback.

Safety

- Protect the user from making mistakes.
  - — Provide visual cues, reminders, lists of choices, and other aids either automatically or upon request.

To eliminate the opportunity for mistakes and confusion, always provide memory support for the user. Never rely upon a person to remember something. If the system possesses the information, it should provide it to the user.

Simplicity

- Provide as simple an interface as possible.
- Ways to provide simplicity:
  - — Use progressive disclosure, hiding things until they are needed.
    - Present common and necessary functions first.
    - Prominently feature important functions.
    - Hide more sophisticated and less frequently used functions.
  - — Provide an obvious visual hierarchy.
  - — Provide defaults.
  - — Minimize screen alignment points.
  - — Make common actions simple at the expense of uncommon actions being made harder.
  - — Provide uniformity and consistency.
  - — Eliminate unnecessary elements.

Simplicity is achieved when everyone can easily understand and use a system regardless of experience, literacy, or concentration level. Simplicity is the opposite of complexity. Complexity is a measure of the number of choices available at any point in the human-computer interaction. A great deal of functionality and power is usually associated with high complexity. Complexity most often overwhelms and confuses new and casual users of systems. Complex systems are often not fully used, or used ineffectively, because a person may follow known but more cumbersome methods instead of easier but undiscovered or unfamiliar methods.

A system lacking complexity may have a different set of faults. It may be tedious to use or it may not accomplish much. It is better, however, to provide less functionality that will get effectively used than to provide too much functionality and yield an interface hopelessly complex and extremely difficult to use. Complexity, then, is a two-edged sword. To effectively solve problems, it must be present without being apparent. The goal, then, is to provide a complex system while masking the complexity through a simple interface. There are several ways to minimize this complexity.

Progressive disclosure. Introduce system components gradually so the full complexity of the system is not visible at first encounter. Teach fundamentals first. Then, slowly introduce advanced or more sophisticated functions. This is called the layered, or spiral, approach to learning. Such an approach was first described

by Carroll and Carrithers (1984) who called it the "Training-Wheels System." They found that disabling portions of the system that were not needed and could lead to errors and confusion improved system learning efficiency.

Provide an obvious visual hierarchy. Depending upon user tasks, establish a hierarchy of importance of screen elements through prominence. Size, contrasting colors, and position are some of the techniques that can be used to structure a screen and focus the user's attention.

Provide defaults. Providing defaults is another form of system layering. When a system is first presented, provide a set of defaults for all system-configurable items. The new user will not be burdened with these decisions and can concentrate on the fundamentals first. Defaults can later be changed, if desired, as experience increases.

Minimize screen alignment points. Several alignment points of elements displayed on a screen are associated with greater screen visual complexity. Minimizing these alignment points minimizes visual complexity. This concept is discussed in more detail in Step 3.

Make common actions simple. Make common actions within a system easier to accomplish than uncommon actions. The benefit will be greater overall system efficiency.

Provide uniformity and consistency. Inconsistency is really a foolish form of complexity. It forces a person to learn that things that appear different are not different.

Eliminate unnecessary elements. Eliminate screen clutter through "subtractive design." Unnecessary elements are distracting and consume a portion of a person's cognitive capabilities. If something does not contribute to effective system use, remove it.

Transparency

- Permit the user to focus on the task or job, without concern for the mechanics of the interface.
  - — Workings and reminders of workings inside the computer should be invisible to the user.

Never force the user to think about the technical details of the system. One's thoughts must be directed to the task, not the computer communication process. Reminders of the mechanics of the interface occur through the use of technical jargon, the heavy use of codes, and the presentation of computer concepts and representations.

Trade-Offs

- Final design will be based on a series of trade-offs balancing often-conflicting design principles.
- People's requirements always take precedence over technical requirements.

Design guidelines often cover a great deal of territory and often conflict with one another or with technical requirements. In such conflicts the designer must weigh the alternatives and reach a decision based on trade-offs concerning accuracy, time, cost, and ease of use. Making these trade-offs intelligently requires a thorough understanding of the user and all design considerations. The ultimate solution will be a blend of experimental data, good judgment, and the important user needs.

This leads to a second cardinal rule of system development: *Human requirements always take precedence over technical requirements.* It may be easier for the designer to write a program or build a device that neglects user ease, but final system judgment will always come down to one simple fact: How well does the system meet the needs of the user?

Visibility

- A system's status and methods of use must be clearly visible.
- Improve visibility through:
  — Hierarchical organization.
  — Context sensitivity.

Systems are more usable when they clearly indicate their status, the possible actions that can be taken, and the results of actions once they are performed. The visibility principle is based upon the fact that one's power of recognition is much stronger than one's power of recall. It is especially useful for managing complex systems. Two solutions providing visibility while managing complexity are hierarchical organization and context sensitivity.

Hierarchical organization places information or controls into logical categories, and then hides them under a parent control such as a menu. A category name remains visible but the information remains hidden until activated.

Context sensitivity presents and hides information and controls based upon the existing system context. While relevant information and controls are made highly visible, irrelevant or unavailable functions are hidden until they become relevant.

The degree of visibility of information and controls should correspond to their relevance.

Part 1 Exercise

An exercise for Part 1 can be found on this book's companion Web site, www.wiley.com/college/galitz.

What's Next?

Let's now move ahead to Part 2, the interface design process. Specific guidelines for good interface design are presented within the context of GUI and Web characteristics, and the general principles just described.