Figure 1.2 Novel forms of interactive products embedded with computational power (clockwise from top left):

(i) Electrolux screen-fridge that provides a range of functionality, including food management where recipes are displayed, based on the food stored in the fridge.

(ii) an IBM prototype of a color electronic ink page, intended for e-newspapers that can ‘typeset’ themselves and update while being light enough to carry around.

(iii) 'geek chic', a Levi jacket equipped with a fully integrated computer network (body area network), enabling the wearer to be fully connected to the web.

(iv) Barney, an interactive cuddly toy that makes learning enjoyable.

Figure 1.11 2D and 3D buttons. Which are easier to distinguish between?
Figure 2.1  An example of augmented reality. Virtual and physical worlds have been combined so that a digital image of the brain is superimposed on the person's head, providing a new form of medical visualization.

Figure 2.14  The i-room project at Stanford: a graphical rendering of the Interactive Room Terry Winograd's group is researching, which is an innovative technology-rich prototype workspace, integrating a variety of displays and devices. An overarching aim is to explore new possibilities for people to work together (see http://graphics.stanford.EDU/projects/iwork/).
Figure 2.6 Recent direct-manipulation virtual environments

(a) Virtue (Daniel Reid, 1999, www-pablo.cs.uiuc.edu/ProjectNRNirtue) enables software developers to directly manipulate software components and their behavior.

(b), (c) Crayoland (Dave Pape, www.ncsa.uiuc.edu) is an interactive virtual environment where the child in the image on the right uses a joystick to navigate through the space. The child is interacting with an avatar in the flower world.
Figure 3.7 Dynalinking used in the **PondWorld** software. In the background is a simulation of a pond ecosystem, comprising perch, stickleback, beetles, tadpoles, and weeds. In the foreground is a food web diagram representing the same ecosystem but at a more abstract level. The two are dynalinked: changes made to one representation are reflected in the other. Here the user has clicked on the arrow between the tadpole and the weed represented in the diagram. This is shown in the **PondWorld** simulation as the tadpole eating the weed. The dynalinking is accompanied by a narrative explaining what is happening and sounds of dying organisms.

Figure 3.9 A see-through handset — transparency does not mean simply showing the insides of a machine but involves providing a good system image.
Figure 4.1 The rooftop garden in **BowieWorld**, a collaborative virtual environment (CVE) supported by Worlds.com. The User takes part by "dressing up" as an avatar. There are hundreds of avatars to choose from, including penguins and real people. Once avatars have entered a world, they can explore it and chat with other avatars.
Figure 5.3 Examples of aesthetically pleasing interactive products: iMac, Nokia cell phone and IDEO’s digital radio for the BBC.

Figure 5.9 Virtual screen characters:

(a) Aibo, the interactive dog.

(b) Ananova, the virtual newscaster.

(c) Ecyas, the German virtual pop star.
Figure 5.11
Herman the bug watches as a student chooses roots for a plant in an Alpine meadow.

Figure 5.12 The Woggles interface, with icons and slider bars representing emotions, speech, and actions.
Figure 5.13 Rea the real estate agent welcoming the user to look at a condo.

Figure 7.3(b) The KordGrip being used underwater

Figure 15.8 The first foam models of a mobile communicator for children.
INTERACTION DESIGN
beyond human-computer interaction

John Wiley & Sons, Inc.
Library of Congress Cataloging in Publication Data:
Preece, Jennifer.
Interaction design: beyond human–computer interaction / Jennifer Preece, Yvonne Rogers, Helen Sharp.
p. cm.
Includes bibliographical references and index.
1. Human-computer interaction. I. Rogers, Yvonne. II. Sharp, Helen. III. Title.
QA76.9.H85 P72 2002
004’.01’9—dc21
Printed in the United States of America
200106730

10 9 8 7 6
Welcome to *Interaction Design: Beyond Human-Computer Interaction*, and our interactive website at [ID-Book.com](http://ID-Book.com)

This textbook is for undergraduate and masters students from a range of backgrounds studying classes in human-computer interaction, interaction design, web design, etc. A broad range of professionals and technology users will also find this book useful, and so will graduate students who are moving into this area from related disciplines.

Our book is called *Interaction Design: Beyond Human-Computer Interaction* because it is concerned with a broader scope of issues, topics, and paradigms than has traditionally been the scope of human-computer interaction (HCI). This reflects the exciting times we are living in, when there has never been a greater need for interaction designers and usability engineers to develop current and next-generation interactive technologies. To be successful they will need a mixed set of skills from psychology, human-computer interaction, web design, computer science, information systems, marketing, entertainment, and business.

What exactly do we mean by interaction design? In essence, we define interaction design as:

"designing interactive products to support people in their everyday and working lives".

This entails creating user experiences that enhance and extend the way people work, communicate, and interact. Now that it is widely accepted that HCI has moved beyond designing computer systems for one user sitting in front of one machine to embrace new paradigms, we, likewise, have covered a wider range of issues. These include ubiquitous computing and pervasive computing that make use of wireless and collaborative technologies. We also have tried to make the book up-to-date with many examples from contemporary research.

The book has 15 chapters and includes discussion of how cognitive, social, and affective issues apply to interaction design. A central theme is that design and evaluation are interleaving, highly iterative processes, with some roots in theory but which rely strongly on good practice to create usable products. The book has a 'hands-on' orientation and explains how to carry out a variety of techniques. It also has a strong pedagogical design and includes many activities (with detailed comments), assignments, and the special pedagogic features discussed below.

The style of writing is intended to be accessible to students, as well as professionals and general readers, so it is conversational and includes anecdotes, cartoons, and case studies. Many of the examples are intended to relate to readers' own experiences. The book and the associated website encourage readers to be active when reading and to think about seminal issues. For example, one feature we have included in the book is the "dilemma," where a controversial topic is aired. The aim is for readers to understand that much of interaction design needs consid-
eration of the issues, and that they need to learn to weigh-up the pros and cons and be prepared to make trade-offs. We particularly want readers to realize that there is rarely a right or wrong answer although there are good designs and poor designs.

This book is accompanied by a website, which provides a variety of resources and interactivities. The website offers a place where readers can learn how to design websites and other kinds of multimedia interfaces. Rather than just provide a list of guidelines and design principles, we have developed various interactivities, including online tutorials and step-by-step exercises, intended to support learning by doing.

Special features

We use both the textbook and the web to teach about interaction design. To promote good pedagogical practice we include the following features:

Chapter design

Each chapter is designed to motivate and support learning:

- **Aims** are provided so that readers develop an accurate model of what to expect in the chapter.
- **Key points** at the end of the chapter summarize what is important.
- **Activities** are included throughout the book and are considered an essential ingredient for learning. They encourage readers to extend and apply their knowledge. Comments are offered directly after the activities, because pedagogic research suggests that turning to the back of the text annoys readers and discourages learning.
- **An assignment** is provided at the end of each chapter. This can be set as a group or individual project. The aim is for students to put into practice and consolidate knowledge and skills either from the chapter that they have just studied or from several chapters. Some of the assignments build on each other and involve developing and evaluating designs or actual products. Hints and guidance are provided on the website.
- **Boxes** provide additional and highlighted information for readers to reflect upon in more depth.
- **Dilemmas** offer honest and thought-provoking coverage of controversial or problematic issues.
- **Further reading** suggestions are provided at the end of each chapter. These refer to seminal work in the field, interesting additional material, or work that has been heavily drawn upon in the text.
- **Interviews** with nine practitioners and visionaries in the field enable readers to gain a personal perspective of the interviewees' work, their philosophies, their ideas about what is important, and their contributions to the field.
- **Cartoons** are included to make the book enjoyable.
ID-Book.com website

The aim of the website is to provide you with an opportunity to learn about interaction design in ways that go "beyond the book." Additional in-depth material, hands-on interactivities, a student's corner and informal tutorials will be provided. Specific features planned include:

- Hands-on interactivities, including designing a questionnaire, customizing a set of heuristics, doing a usability analysis on 'real' data, and interactive tools to support physical design.
- Recent case studies.
- Student's corner where you will be able to send in your designs, thoughts, written articles which, if suitable, will be posted on the site at specified times during the year.
- Hints and guidance on the assignments outlined in the book.
- Suggestions for additional material to be used in seminars, lab classes, and lectures.
- Key terms and concepts (with links to where to find out more about them).

Readership

This book will be useful to a wide range of readers with different needs and aspirations.

*Students* from Computer Science, Software Engineering, Information Systems, Psychology, Sociology, and related disciplines studying courses in Interaction Design and Human-Computer Interaction will learn the knowledge, skills, and techniques for designing and evaluating state-of-the-art products, and websites, as well as traditional computer systems.

*Web and Interaction Designers,* and *Usability Professionals* will find plenty to satisfy their need for immediate answers to problems as well as for building skills to satisfy the demands of today's fast moving technical market.

*Users,* who want to understand why certain products can be used with ease while others are unpredictable and frustrating, will take pleasure in discovering that there is a discipline with practices that produce usable systems.

*Researchers* and *developers* who are interested in exploiting the potential of the web, wireless, and collaborative technologies will find that, as well as offering guidance, techniques, and much food for thought, a special effort has been made to include examples of state-of-the-art systems.

In the next section we recommend various routes through the text for different kinds of readers.

How to use this book

Interaction Design is not a linear design process but is essentially iterative and some readers and experienced instructors will want to find their own way through the chapters. Others, and particularly those with less experience, may prefer to
work through chapter by chapter. Readers will also have different needs. For example, students in Psychology will come with different background knowledge and needs from those in Computer Science. Similarly, professionals wanting to learn the fundamentals in a one-week course have different needs. This book and the website are designed for using in various ways. The following suggestions are provided to help you decide which way is best for you.

From beginning to end

There are fifteen chapters so students can study one chapter per week during a fifteen-week semester course. Chapter 15 contains design and evaluation case studies. Our intention is that these case studies help to draw together the contents of the rest of the book by showing how design and evaluation are done in the real world. However, some readers may prefer to dip into them along the way.

Getting a quick overview

For those who want to get a quick overview or just the essence of the book, we suggest you read Chapters 1, 6, and 10. These chapters are recommended for everyone.

Suggestions for computer science students

In addition to reading Chapters 1, 6, and 10, Chapters 7 and 8 contain the material that will feel most familiar to any students who have been introduced to software development. These chapters cover the process of interaction design and the activities it involves, including establishing requirements, conceptual design, and physical design. The book itself does not include any coding exercises, but the website will provide tools and widgets with which to interact.

For those following the ACM-IEEE Curriculum (2001)*, you will find that this text and website cover most of this curriculum. The topics listed under each of the following headings are discussed in the chapters shown:

- **HC1** Foundations of Human-Computer Interaction (Chapters 1–5, 14, website).
- **HC2** Building a simple graphical user interface (Chapters 1, 6, 8, 10 and the website).
- **HC3** Human-Centered Software Evaluation (Chapters 1, 10–15, website).
- **HC4** Human-Centered Software Design (Chapters 1, 6–9, 15).
- **HC5** Graphical User-Interface Design (Chapters 2 and 8 and the website. Many relevant examples are discussed in Chapters 1–5 integrated with discussion of cognitive and social issues).

*ACM–IEEE Curriculum (2001) [computer.org/education/cc2001/] is under development at the time of writing this book.
Suggestions for information systems students

Information systems students will benefit from reading the whole text, but instructors may want to find additional examples of their own to illustrate how issues apply to business applications. Some students may be tempted to skip Chapters 3-5 but we recommend that they should read these chapters since they provide important foundational material. This book does not cover how to develop business cases or marketing.

Suggestions for psychology and cognitive science students

Chapters 3-5 cover how theory and research findings have been applied to interaction design. They discuss the relevant issues and provide a wide range of studies and systems that have been informed by cognitive, social, and affective issues. Chapters 1 and 2 also cover important conceptual knowledge, necessary for having a good grounding in interaction design.

Practitioner and short course route

Many people want the equivalent of a short intensive 2-5 day course. The best route for them is to read Chapters 1, 6, 10 and 11 and dip into the rest of the book for reference. For those who want practical skills, we recommend Chapter 8.

Plan your own path

For people who do not want to follow the "beginning-to-end" approach or the suggestions above, there are many ways to use the text. Chapters 1, 6, 10 and 11 provide a good overview of the topic. Chapter 1 is an introduction to key issues in the discipline and Chapters 6 and 10 offer introductions to design and evaluation. Then go to Chapters 2-5 for user issues, then on to the other design chapters, 2-9, dipping into the evaluation chapters 10-14 and the case studies in 15. Another approach is to start with one or two of the evaluation chapters after first reading Chapters 1, 6, 10 and 11, then move into the design section, drawing on Chapters 2-5 as necessary.

Web designer route

Web designers who have a background in technology and want to learn how to design usable and effective websites are advised to read Chapters 1, 7, 8, 13 and 14.
These chapters cover key issues that are important when designing and evaluating the usability of websites. A worked assignment runs through these chapters.

**Usability professionals’ route**

Usability professionals who want to extend their knowledge of evaluation techniques and read about the social and psychological issues that underpin design of the web, wireless, and collaborative systems are advised to read Chapter 1 for an overview, then select from Chapters 10–14 on usability testing. Chapters 3, 4, and 5 provide discussion of seminal user issues (cognitive, social, and affective aspects). There is new material throughout the rest of the book, which will also be of interest for dipping into as needed. This group may also be particularly interested in Chapter 8 which, together with material on the book website, provides practical design examples.

**Acknowledgements**

Many people have helped to make this book a reality. We have benefited from the advice and support of our many professional colleagues across the world, our students, friends, and families and we thank you all. We also warmly thank the following people for reviewing the manuscript and making many helpful suggestions for improvements: Liam Bannon, Sara Bly, Penny Collings, Paul Dourish, Jean Gasen, Peter Gregor, Stella Mills, Rory O'Connor, Scott Toolson, Terry Winograd, Richard Furuta, Robert J.K. Jacob, Blair Nonnecke, William Buxton, Carol Traynor, Blaise Liffich, Jan Scott, Sten Hendrickson, Ping Zhang, Lyndsay Marshall, Gary Perlman, Andrew Dillon, Michael Harrison, Mark Crenshaw, Laurie Dingers, David Carr, Steve Howard, David Squires, George Weir, Marilyn Tremaine, Bob Fields, Frances Slack, Ian Graham, Alan O’Callaghan, Sylvia Wilbur, and several anonymous reviewers. We also thank Geraldine Fitzpatrick, Tim and Dirk from DSTC (Australia) for their feedback on Chapters 1 and 4, Mike Scaife, Harry Brignull, Matt Davies, the HCCS Masters students at Sussex University (2000–2001), Stephanie Wilson and the students from the School of Informatics at City University and Information Systems Department at UMBC for their comments.

We are particularly grateful to Sara Bly, Karen Holtzblatt, Jakob Nielsen, Abigail Sellen, Suzanne Robertson, Gitta Salomon, Ben Shneiderman, Gillian Crampston Smith, and Terry Winograd for generously contributing in-depth interviews.

Lili Cheng and her colleagues allowed us to use the HutchWorld case study. Bill Killam provided the TRZS case study. Keith Cogdill supplied the MEDLINE-plus case study. We thank Lili, Bill, and Keith for supplying the basic reports and commenting on various drafts. Jon Lazar and Dorine Andrews contributed material for the section on questionnaires, which we thank them for.

We are grateful to our Editors Paul Crockett and Gaynor Redvers-Mutton and the production team at Wiley: Maddy Lesure, Susannah Barr, Anna Melhorn, Gemma Quilter, and Ken Santor. Without their help and skill this book would not have been produced. Bill Zobrist and Simon Plumtree played a significant role in persuading us to work with Wiley and we thank them too.
About the authors

The authors are all senior academics with a background in teaching, researching, and consulting in the UK, USA, Canada, Australia, and Europe. Having worked together on two other successful text books, they bring considerable experience in curriculum development, using a variety of media for distance learning as well as face-to-face teaching. They have considerable knowledge of creating learning texts and websites that motivate and support learning for a range of students.

All three authors are specialists in interaction design and human-computer interaction (HCI). In addition they bring skills from other discipline-. Yvonne Rogers is a cognitive scientist, Helen Sharp is a software engineer, and Jenny Preece works in information systems. Their complementary knowledge and skills enable them to cover the breadth of concepts in interaction design and HCI to produce an interdisciplinary text and website. They have collaborated closely, supporting and commenting upon each other’s work to produce a high degree of integration of ideas with one voice. They have shared everything from initial concepts, through writing, design and production.
## Contents

### Chapter 1

**What is interaction design?**  1

1.1 Introduction  1

1.2 Good and poor design  2

1.2.1 What to design  4

1.3 What is interaction design?  6

1.3.1 The makeup of interaction design  6

1.3.2 Working together as a **multidisciplinary** team  9

1.3.3 Interaction design in business  10

1.4 What is involved in the process of interaction design?  12

1.5 The **goals** of interaction design  13

1.5.1 Usability **goals**  14

1.5.2 User experience **goals**  18

1.6 More on usability: design and usability principles  20

1.6.1 Heuristics and usability principles  26

Interview with Gitta Salomon  31

### Chapter 2

**Understanding and conceptualizing interaction**  35

2.1 Introduction  35

2.2 Understanding the problem space  36

2.3 Conceptual models  39

2.3.1 Conceptual models based on activities  41

2.3.2 Conceptual models based on **objects**  51

2.3.3 A case of mix and match?  54

2.4 Interface metaphors  55

2.5 Interaction paradigms  60

2.6 From conceptual models to physical design  64

Interview with Terry Winograd  70

### Chapter 3

**Understanding users**  73

3.1 Introduction  73

3.2 What is cognition?  74

3.3 Applying knowledge from the **physical** world to the digital world  90

3.4 Conceptual frameworks for cognition  92

3.4.1 Mental models  92
Chapter 4  Designing for collaboration and communication  105
  4.1 Introduction 105
  4.2 Social mechanisms used in communication and collaboration 106
    4.2.1 Conversational mechanisms 107
    4.2.2 Designing collaborative technologies to support conversation 110
    4.2.3 Coordination mechanisms 118
    4.2.4 Designing collaborative technologies to support coordination 122
    4.2.5 Awareness mechanisms 124
    4.2.6 Designing collaborative technologies to support awareness 126
  4.3 Ethnographic studies of collaboration and communication 129
  4.4 Conceptual frameworks 130
    4.4.1 The language/action framework 130
    4.4.2 Distributed cognition 133
    Interview with Abigail Sellen 138

Chapter 5  Understanding how interfaces affect users  141
  5.1 Introduction 141
  5.2 What are affective aspects? 142
  5.3 Expressive interfaces 143
  5.4 User frustration 147
    5.4.1 Dealing with user frustration 152
  5.5 A debate: the application of anthropomorphism to interaction design 153
  5.6 Virtual characters: agents 157
    5.6.1 Kinds of agents 157
    5.6.2 General design concerns 160

Chapter 6  The process of interaction design  165
  6.1 Introduction 165
  6.2 What is interaction design about? 166
    6.2.1 Four basic activities of interaction design 168
    6.2.2 Three key characteristics of the interaction design process 170
  6.3 Some practical issues 170
    6.3.1 Who are the users? 171
Chapter 7

**Identifying needs and establishing requirements**

7.1 Introduction 201

7.2 What, how, and why? 202
   7.2.1 What are we trying to achieve in this design activity? 202
   7.2.2 How can we achieve this? 202
   7.2.3 Why bother? The importance of getting it right 203
   7.2.4 Why establish requirements? 204

7.3 What are requirements? 204
   7.3.1 Different kinds of requirements 205

7.4 Data gathering 210
   7.4.1 Data-gathering techniques 211
   7.4.2 Choosing between techniques 215
   7.4.3 Some basic data-gathering guidelines 216

7.5 Data interpretation and analysis 219

7.6 Task description 222
   7.6.1 Scenarios 223
   7.6.2 Use cases 226
   7.6.3 Essential use cases 229

7.7 Task analysis 231
   7.7.1 Hierarchical Task Analysis (HTA) 231

Interview with Suzanne Robertson 236

Chapter 8

**Design, prototyping and construction** 239

8.1 Introduction 239

8.2 Prototyping and construction 240
   8.2.1 What is a prototype? 240
   8.2.2 Why prototype? 241
   8.2.3 Low-fidelity prototyping 243
   8.2.4 High-fidelity prototyping 245
   8.2.5 Compromises in prototyping 246
8.2.6 Construction: from design to implementation 248
8.3 Conceptual design: moving from requirements to first design 249
  8.3.1 Three perspectives for developing a conceptual model 250
  8.3.2 Expanding the conceptual model 257
  8.3.3 Using scenarios in conceptual design 259
  8.3.4 Using prototypes in conceptual design 262
8.4 Physical design: getting concrete 264
  8.4.1 Guidelines for physical design 266
  8.4.2 Different kinds of widget 268
8.5 Tool support 275

Chapter 9  User-centered approaches to interaction design 279
  9.1 Introduction 279
  9.2 Why is it important to involve users at all? 280
    9.2.1 Degrees of involvement 281
  9.3 What is a user-centered approach? 285
  9.4 Understanding users' work: applying ethnography in design 288
    9.4.1 Coherence 293
    9.4.2 Contextual Design 295
  9.5 Involving users in design: Participatory Design 306
    9.5.1 PICTIVE 307
    9.5.2 CARD 309
Interview with Karen Holtzblatt 313

Chapter 10  Introducing evaluation 317
  10.1 Introduction 317
  10.2 What, why, and when to evaluate 318
    10.2.1 What to evaluate 318
    10.2.2 Why you need to evaluate 319
    10.2.3 When to evaluate 323
  10.3 HutchWorld case study 324
    10.3.1 How the team got started: early design ideas 324
    10.3.2 How was the testing done? 327
    10.3.3 Was it tested again? 333
    10.3.4 Looking to the future 334
  10.4 Discussion 336

Chapter 11  An evaluation framework 339
  11.1 Introduction 339
11.2 Evaluation paradigms and techniques 340
   11.2.1 Evaluation paradigms 341
   11.2.2 Techniques 345
11.3 DECIDE: A framework to guide evaluation 348
   11.3.1 Determine the goals 348
   11.3.2 Explore the questions 349
   11.3.3 Choose the evaluation paradigm and techniques 349
   11.3.4 Identify the practical issues 350
   11.3.5 Decide how to deal with the ethical issues 351
   11.3.6 Evaluate, interpret, and present the data 355
11.4 Pilot studies 356

Chapter 12 Observing users 359
12.1 Introduction 359
12.2 Goals, questions and paradigms 360
   12.2.1 What and when to observe 361
   12.2.2 Approaches to observation 363
12.3 How to observe 364
   12.3.1 In controlled environments 365
   12.3.2 In the field 368
   12.3.3 Participant observation and ethnography 370
12.4 Data collection 373
   12.4.1 Notes plus still camera 374
   12.4.2 Audio recording plus still camera 374
   12.4.3 Video 374
12.5 Indirect observation: tracking users' activities 377
   12.5.1 Diaries 377
   12.5.2 Interaction logging 377
12.6 Analyzing, interpreting and presenting data 379
   12.6.1 Qualitative analysis to tell a story 380
   12.6.2 Qualitative analysis for categorization 381
   12.6.3 Quantitative data analysis 384
   12.6.4 Feeding the findings back into design 384
Interview with Sara Bly 387

Chapter 13 Asking users and experts 389
13.1 Introduction 389
13.2 Asking users: interviews 390
   13.2.1 Developing questions and planning an interview 390
13.2.2 Unstructured interviews 392
13.2.3 Structured interviews 394
13.2.4 Semi-structured interviews 394
13.2.5 Group interviews 396
13.2.6 Other sources of interview-like feedback 397
13.2.7 Data analysis and interpretation 398

13.3 Asking users: Questionnaires 398
13.3.1 Designing questionnaires 398
13.3.2 Question and response format 400
13.3.3 Administering questionnaires 404
13.3.4 Online questionnaires 405
13.3.5 Analyzing questionnaire data 407

13.4 Asking experts: Inspections 407
13.4.1 Heuristic evaluation 408
13.4.2 Doing heuristic evaluation 410
13.4.3 Heuristic evaluation of websites 412
13.4.4 Heuristics for other devices 419

13.5 Asking experts: walkthroughs 420
13.5.1 Cognitive walkthroughs 420
13.5.2 Pluralistic walkthroughs 423

Interview with Jakob Nielsen 426

Chapter 14  Testing and modeling users 429
14.1 Introduction 429
14.2 User testing 430
14.2.1 Testing MEDLINEplus 432
14.3 Doing user testing 438
14.3.1 Determine the goals and explore the questions 439
14.3.2 Choose the paradigm and techniques 439
14.3.3 Identify the practical issues: Design typical tasks 439
14.3.4 Identify the practical issues: Select typical users 440
14.3.5 Identify the practical issues: Prepare the testing conditions 441
14.3.6 Identify the practical issues: Plan how to run the tests 442
14.3.7 Deal with ethical issues 443
14.3.8 Evaluate, analyze, and present the data 443

14.4 Experiments 443
14.4.1 Variables and conditions 444
14.4.2 Allocation of participants to conditions 445
Contents

14.4.3 Other practical issues 446
14.4.4 Data collection and analysis 446
14.5 Predictive models 448
  14.5.1 The W M S model 449
  14.5.2 The Keystroke level model 450
  14.5.3 Benefits and limitations of W M S 453
  14.5.4 Fitts' Law 454
Interview with Ben Shneiderman 457

Chapter 15 Design and evaluation in the real world: communicators and advisory systems 461
  15.1 Introduction 461
  15.2 Key Issues 462
  15.3 Designing mobile communicators 463
    15.3.1 Background 463
    15.3.2 Nokia's approach to developing a communicator 464
    15.3.3 Philip's approach to designing a communicator for children 474
  15.4 Redesigning part of a large interactive phone-based response system 482
    15.4.1 Background 483
    15.4.2 The redesign 483

Reflections from the Authors 491

References 493

Credits 503

Index 509
As predicted by many visionaries, devices everywhere are getting "smarter." My camera has a multi-modal hierarchical menu and form interface. Even my toaster has a microprocessor. Computing is not just for computers anymore. So when the authors wrote the subtitle "beyond human-computer interaction," they wanted to convey that the book generalizes the human side to people, both individuals and groups, and the computer side to desktop computers, handheld computers, phones, cameras...maybe even toasters.

My own interest in this book is motivated by having been a software developer for 20 years, during which time I was a professor and consultant for 12. Would the book serve as a textbook for students? Would it help bring software development practice into a new age of human-centered interaction design?

A textbook for students...

More than anything, I think students need to be motivated, inspired, challenged, and I think this book, particularly Chapters 1–5, will do that. Many students will not have the motivating experience of seeing projects and products fail because of a lack of attention, understanding, and zeal for the user, but as I read the opening chapters, I imagined students thinking, "This is what I've been looking for!" The interviews will provide students with the wisdom of well-chosen experts: what's important, what worked (or didn't), and why. I see students making career choices based on this motivating material.

The rest of the book covers the art and some of the science of interaction design, the basic knowledge needed by practitioners and future innovators. Chapters 6–9 give a current view of analysis, design, and prototyping, and the book's website should add motivating examples. Chapters 10–14 cover evaluation in enough depth to facilitate understanding, not just rote application. Chapter 15 brings it all together, adding more depth. For each topic, there are ample pointers to further reading, which is important because interaction design is not a one-book discipline.

Finally, the book itself is pedagogically well designed. Each chapter describes its aims, contains examples and subtopics, and ends with key points, assignments, and an annotated bibliography for more detail.

A guide for development teams...

When I lead or consult on software projects, I face the same problem over and over: many people in marketing and software development—these are the people who have the most input into design, but it applies to any members of multidisciplinary teams—have little knowledge or experience building systems with a user-centered
focus. A user-centered focus requires close work with users (not just customer-buyers), from analysis through design, evaluation, and maintenance. A lack of user-centered focus results in products and services that often do not meet the needs of their intended users. Don Norman’s design books have convinced many that these problems are not unique to software, so this book’s focus on interaction design feels right.

To help software teams adopt a user-centered focus, I’ve searched for books with end-to-end coverage from analysis, to design, to implementation (possibly of prototypes), to evaluation (with iteration). Some books have tried to please all audiences and have become encyclopedias of user interface development, covering topics worth knowing, but not in enough detail for readers to understand them. Some books have tried to cover theory in depth and tried to appeal to developers who have little interest in theory. Whatever the reasons for these choices, the results have been lacking. This book has chosen fewer topics and covered them in more depth; enough depth, I think, to put the ideas into practice. I think the material is presented in a way that is understandable by a wide audience, which is important in order for the book to be useful to whole multidisciplinary teams.

A recommended book . . .

I’ve been waiting for this book for many years. I think it’s been worth the wait.

As the director of the HCI Bibliography project (www.hcibib.org), a free-access HCI portal receiving a half-million hits per year, I receive many requests for suggestions for books, particularly from students and software development managers. To answer that question, I maintain a list of recommended readings in ten categories (with 20,000 hits per year). Until now, it’s been hard to recommend just one book from that list. I point people to some books for motivation, other books for process, and books for specific topics (e.g., task analysis, ergonomics, usability testing). This book fits well into half the categories in my list and makes it easier to recommend one book to get started and to have on hand for development.

I welcome the commitment of the authors to building a website for the book. It’s a practice that has been adopted by other books in the field to offer additional information and keep the book current. The site also presents interactive content to aid in tasks like conducting surveys and heuristic evaluations. I look forward to seeing the book’s site present new materials, but as director of www.hcibib.org, I hope they use links to instead of re-inventing existing resources.

Gary Perlman
Columbus
October 2001
About Gary Perlman

Gary Perlman is a consulting research scientist at the OCLC–Online Computer Library Center (www.oclc.org) where he works on user interfaces for bibliographic and full-text retrieval. His research interests are in making information technology more useful and usable for people.

He has also held research and academic positions at Bell Labs in Murray Hill, New Jersey; Wang Institute of Graduate Studies; Massachusetts Institute of Technology; Carnegie-Mellon University; and The Ohio State University. Dr. Perlman's Ph.D. is in experimental psychology from the University of California, San Diego. He is the author of over 75 publications in the areas of mathematics education, statistical computing, hypertext, and user interface development. He has lectured and consulted internationally since 1980.

He is best known in the HCI community as the director of the HCI Bibliography (www.hcilib.org), a free-access online resource of over 20,000 records searched hundreds of thousands of times each year.

A native of Montreal, Canada, Gary now lives in Columbus, Ohio with his wife and two sons.
Chapter 1

What is interaction design?

1.1 Introduction
1.2 Good and poor design
   1.2.1 What to design
1.3 What is interaction design?
   1.3.1 The makeup of interaction design
   1.3.2 Working together as a multidisciplinary team
   1.3.3 Interaction design in business
1.4 What is involved in the process of interaction design?
1.5 The goals of interaction design
   1.5.1 Usability goals
   1.5.2 User experience goals
1.6 More on usability: design and usability principles

1.1 Introduction

How many interactive products are there in everyday use? Think for a minute about what you use in a typical day: cell phone, computer, personal organizer, remote control, soft drink machine, coffee machine, ATM, ticket machine, library information system, the web, photocopier, watch, printer, stereo, calculator, video game... the list is endless. Now think for a minute about how usable they are. How many are actually easy, effortless, and enjoyable to use? All of them, several, or just one or two? This list is probably considerably shorter. Why is this so?

Think about when some device caused you considerable grief—how much time did you waste trying to get it to work? Two well-known interactive devices that cause numerous people immense grief are the photocopier that doesn't copy the way they want and the VCR that records a different program from the one they thought they had set or none at all. Why do you think these things happen time and time again? Moreover, can anything be done about it?

Many products that require users to interact with them to carry out their tasks (e.g., buying a ticket online from the web, photocopying an article, pre-recording a TV program) have not necessarily been designed with the users in mind. Typically, they have been engineered as systems to perform set functions. While they may work effectively from an engineering perspective, it is often at the expense of how the system will be used by real people. The aim of interaction design is to redress this concern by
Chapter 1  What is interaction design?

bringing usability into the design process. In essence, it is about developing interactive products that are easy, effective, and enjoyable to use—from the users' perspective.

In this chapter we begin by examining what interaction design is. We look at the difference between good and poor design, highlighting how products can differ radically in their usability. We then describe what and who is involved in interaction design. In the last part of the chapter we outline core aspects of usability and how these are used to assess interactive products. An assignment is presented at the end of the chapter in which you have the opportunity to put into practice what you have read, by evaluating an interactive product using various usability criteria.

The main aims of the chapter are to:

- Explain the difference between good and poor interaction design.
- Describe what interaction design is and how it relates to human-computer interaction and other fields.
- Explain what usability is.
- Describe what is involved in the process of interaction design.
- Outline the different forms of guidance used in interaction design.
- Enable you to evaluate an interactive product and explain what is good and bad about it in terms of the goals and principles of interaction design.

1.2  Good and poor design

A central concern of interaction design is to develop interactive products that are usable. By this is generally meant easy to learn, effective to use, and provide an enjoyable user experience. A good place to start thinking about how to design usable interactive products is to compare examples of well and poorly designed ones. Through identifying the specific weaknesses and strengths of different interactive systems, we can begin to understand what it means for something to be usable or not. Here, we begin with an example of a poorly designed system—voice mail—that is used in many organizations (businesses, hotels, and universities). We then compare this with an answering machine that exemplifies good design.

Imagine the following scenario. You're staying at a hotel for a week while on a business trip. You discover you have left your cell (mobile) phone at home so you have to rely on the hotel's facilities. The hotel has a voice-mail system for each room. To find out if you have a message, you pick up the handset and listen to the tone. If it goes "beep beep beep" there is a message. To find out how to access the message you have to read a set of instructions next to the phone.

You read and follow the first step:

"1. Touch 491".

The system responds, "You have reached the Sunny Hotel voice message center. Please enter the room number for which you would like to leave a message."

\*We use the term interactive products generically to refer to all classes of interactive systems, technologies, environments, tools, applications, and devices.
You wait to hear how to listen to a recorded message. But there are no further instructions from the phone. You look down at the instruction sheet again and read: "2. Touch*, your room number, and #". You do so and the system replies, "You have reached the mailbox for room 106. To leave a message type in your password." You type in the room number again and the system replies, "Please enter room number again and then your password."

You don't know what your password is. You thought it was the same as your room number. But clearly not. At this point you give up and call reception for help. The person at the desk explains the correct procedure for recording and listening to messages. This involves typing in, at the appropriate times, the room number and the extension number of the phone (the latter is your password, which is different from the room number). Moreover, it takes six steps to access a message and five steps to leave a message. You go out and buy a new cell phone.

What is problematic with this voice-mail system?

- It is infuriating.
- It is confusing.
- It is inefficient, requiring you to carry out a number of steps for basic tasks.
- It is difficult to use.
- It has no means of letting you know at a glance whether any messages have been left or how many there are. You have to pick up the handset to find out and then go through a series of steps to listen to them.
- It is not obvious what to do: the instructions are provided partially by the system and partially by a card beside the phone.

Now consider the following phone answering machine. Figure 1.1 shows two small sketches of an answering machine phone. Incoming messages are represented using physical marbles. The number of marbles that have moved into the pinball-like chute indicates the number of messages. Dropping one of these marbles into a slot in the machine causes the recorded message to play. Dropping the same marble into another slot on the phone dials the caller who left the message.

![Figure 1.1 Two small sketches showing answering phone.](image-url)
Chapter 1  What is interaction design?

How does the "marble" answering machine differ from the voice-mail system?

- It uses familiar physical objects that indicate visually at a glance how many messages have been left.
- It is aesthetically pleasing and enjoyable to use.
- It only requires one-step actions to perform core tasks.
- It is a simple but elegant design.
- It offers less functionality and allows anyone to listen to any of the messages.

The marble answering machine was designed by Durrell Bishop while a student at the Royal College of Art in London (described by Crampton-Smith, 1995). One of his goals was to design a messaging system that represented its basic functionality in terms of the behavior of everyday objects. To do this, he capitalized on people's everyday knowledge of how the physical world works. In particular, he made use of the ubiquitous everyday action of picking up a physical object and putting it down in another place. This is an example of an interactive product designed with the users in mind. The focus is on providing them with an enjoyable experience but one that also makes efficient the activity of receiving messages. However, it is important to note that although the marble answering machine is a very elegant and usable design, it would not be practical in a hotel setting. One of the main reasons is that it is not robust enough to be used in public places, for example, the marbles could easily get lost or taken as souvenirs. Also, the need to identify the user before allowing the messages to be played is essential in a hotel setting. When considering the usability of a design, therefore, it is important to take into account where it is going to be used and who is going to use it. The marble answering machine would be more suited in a home setting—provided there were no children who might be tempted to play with the marbles!

1.2.1  What to design

Designing usable interactive products thus requires considering who is going to be using them and where they are going to be used. Another key concern is understanding the kind of activities people are doing when interacting with the products. The appropriateness of different kinds of interfaces and arrangements of input and output devices depends on what kinds of activities need to be supported. For example, if the activity to be supported is to let people communicate with each other at a distance, then a system that allows easy input of messages (spoken or written) that can be readily accessed by the intended recipient is most appropriate. In addition, an interface that allows the users to interact with the messages (e.g., edit, annotate, store) would be very useful.

The range of activities that can be supported is diverse. Just think for a minute what you can currently do using computer-based systems: send messages, gather information, write essays, control power plants, program, draw, plan, calculate, play games—to name but a few. Now think about the number of interfaces and interactive devices that are available. They, too, are equally diverse:
multimedia applications, virtual-reality environments, speech-based systems, personal digital assistants and large displays—to name but a few. There are also many ways of designing the way users can interact with a system (e.g., via the use of menus, commands, forms, icons, etc.). Furthermore, more and more novel forms of interaction are appearing that comprise physical devices with embedded computational power, such as electronic ink, interactive toys, smart fridges, and networked clothing (See Figure 1.2 on Color Plate 1). What this all amounts to is a multitude of choices and decisions that confront designers when developing interactive products.

A key question for interaction design is: how do you optimize the users' interactions with a system, environment or product, so that they match the users' activities that are being supported and extended? One could use intuition and hope for the best. Alternatively, one can be more principled in deciding which choices to make by basing them on an understanding of the users. This involves:

- taking into account what people are good and bad at
- considering what might help people with the way they currently do things
- thinking through what might provide quality user experiences
- listening to what people want and getting them involved in the design
- using "tried and tested" user-based techniques during the design process

The aim of this book is to cover these aspects with the goal of teaching you how to carry out interaction design. In particular, it focuses on how to identify users' needs, and from this understanding, move to designing usable, useful, and enjoyable systems.

**ACTIVITY 1.1** How does making a phone call differ when using:

- a public phone box
- a cell phone?

How have these devices been designed to take into account (a) the kind of users, (b) type of activity being supported, and (c) context of use?

**Comment**

(a) Public phones are designed to be used by the general public. Many have Braille embossed on the keys and speaker volume control to enable people who are blind and hard of hearing to use them.

Cell phones are intended for all user groups, although they can be difficult to use for people who are blind or have limited manual dexterity.

(b) Most phone boxes are designed with a simple mode of interaction: insert card or money and key in the phone number. If engaged or unable to connect the money or card is returned when the receiver is replaced. There is also the option of allowing the caller to make a follow-on call by pressing a button rather than collecting the money and reinserting it again. This function enables the making of multiple calls to be more efficient.
Chapter 1 What is interaction design?

Cell phones have a more complex mode of interaction. More functionality is provided, requiring the user to spend time learning how to use them. For example, users can save phone numbers in an address book and then assign these to "hotkeys," allowing them to be called simply through pressing one or two keys.

(c) Phone boxes are intended to be used in public places, say on the street or in a bus station, and so have been designed to give the user a degree of privacy and noise protection through the use of hoods and booths.

Cell phones have been designed to be used any place and any time. However, little consideration has been given to how such flexibility affects others who may be in the same public place (e.g., sitting on trains and buses).

1.3 What is interaction design?

By interaction design, we mean

*designing interactive products to support people in their everyday and working lives.*

In particular, it is about creating user experiences that enhance and extend the way people work, communicate and interact. Winograd (1997) describes it as "the design of spaces for human communication and interaction." In this sense, it is about finding ways of supporting people. This contrasts with software engineering, which focuses primarily on the production of software solutions for given applications. A simple analogy to another profession, concerned with creating buildings, may clarify this distinction. In his account of interaction design, Terry Winograd asks how architects and civil engineers differ when faced with the problem of building a house. Architects are concerned with the people and their interactions with each other and within the house being built. For example, is there the right mix of family and private spaces? Are the spaces for cooking and eating in close proximity? Will people live in the space being designed in the way it was intended to be used? In contrast, engineers are interested in issues to do with realizing the project. These include practical concerns like cost, durability, structural aspects, environmental aspects, fire regulations, and construction methods. Just as there is a difference between designing and building a house, so too, is there a distinction between interaction design and software engineering. In a nutshell, interaction design is related to software engineering in the same way as architecture is related to civil engineering.

1.3.1 The makeup of interaction design

It has always been acknowledged that for interaction design to succeed many disciplines need to be involved. The importance of understanding how users act and react to events and how they communicate and interact together has led people from a variety of disciplines, such as psychologists and sociologists, to become involved. Likewise, the growing importance of understanding how to design different kinds of interactive media in effective and aesthetically pleasing ways has led to a
diversity of other practitioners becoming involved, including graphic designers, artists, animators, photographers, film experts, and product designers. Below we outline a brief history of interaction design.

In the early days, engineers designed hardware systems for engineers to use. The computer interface was relatively straightforward, comprising various switch panels and dials that controlled a set of internal registers. With the advent of monitors (then referred to as visual display units or VDUs) and personal workstations in the late ’70s and early ’80s, interface design came into being (Grudin, 1990). The new concept of the user interface presented many challenges:

*Terror. You have to confront the documentation. You have to learn a new language. Did you ever use the word ‘interface’ before you started using the computer?*


One of the biggest challenges at that time was to develop computers that could be accessible and usable by other people, besides engineers, to support tasks involving human cognition (e.g., doing sums, writing documents, managing accounts, drawing plans). To make this possible, computer scientists and psychologists became involved in designing user interfaces. Computer scientists and software engineers developed high-level programming languages (e.g., BASIC, Prolog), system architectures, software design methods, and command-based languages to help in such tasks, while psychologists provided information about human capabilities (e.g., memory, decision making).

The scope afforded by the interactive computing technology of that time (i.e., the combined use of visual displays and interactive keyboards) brought about many new challenges. Research into and development of graphical user interfaces (GUI for short, pronounced "goo-ee") for office-based systems took off in a big way. There was much research into the design of widgets (e.g., menus, windows, palettes, icons) in terms of how best to structure and present them in a GUI.

In the mid ’80s, the next wave of computing technologies—including speech recognition, multimedia, information visualization, and virtual reality—presented even more opportunities for designing applications to support even more people. Education and training were two areas that received much attention. Interactive learning environments, educational software, and training simulators were some of the main outcomes. To build these new kinds of interactive systems, however, required a different kind of expertise from that of psychologists and computer programmers. Educational technologists, developmental psychologists, and training experts joined in the enterprise.

As further waves of technological development surfaced in the ’90s—networking, mobile computing, and infrared sensing—the creation of a diversity of applications for all people became a real possibility. All aspects of a person's life—at home, on the move, at school, at leisure as well as at work, alone, with family or friends—began to be seen as areas that could be enhanced and extended by designing and integrating various arrangements of computer technologies. New ways of learning, communicating, working, discovering, and living were envisioned.
What is interaction design?

In the mid '90s, many companies realized it was necessary again to extend their existing multidisciplinary design teams to include professionals trained in media and design, including graphical design, industrial design, film, and narrative. Sociologists, anthropologists, and dramaturgists were also brought on board, all having quite a different take on human interaction from psychologists. This wider set of

---

**Box 1.1 The Relationship between Interaction Design, Human-Computer Interaction, and Other Approaches**

We view interaction design as fundamental to all disciplines, fields, and approaches that are concerned with researching and designing computer-based systems for people (see Figure 1.3). The best-known interdisciplinary field is human-computer interaction (HCI), which is “concerned with the design, evaluation, and implementation of interactive computing systems for human use and with the study of major phenomena surrounding them” (ACM SIGCHI, 1992, p. 6). Until the early '90s, the focus of HCI was primarily designing interfaces for single users. In response to a growing concern for the need also to support multiple individuals working together using computer systems, the interdisciplinary field of computer-supported cooperative work (CSCW) emerged (Greif, 1988). Information systems is another area concerned with the application of computing technology in domains like business, health, and education. Other fields related to interaction design include human factors, cognitive ergonomics, and cognitive engineering. All are concerned with designing systems to match users’ goals, however, each has a different focus and methodology.

![Diagram showing the relationship between interaction design, human-computer interaction, and other approaches.](image)

**Figure 1.3 Relationship among contributing academic disciplines, design practices, and interdisciplinary fields concerned with interaction design.**

---
people were thought to have the right mix of skills and understanding of the different application areas necessary to design the new generation of interactive systems. For example, designing a reminder application for the family requires understanding how families interact; creating an interactive story kit for children requires understanding how children write and understand narrative, and developing an interactive guide for art-gallery visitors requires appreciating what people do and how they move through public spaces.

Now in the '00s, the possibilities afforded by emerging hardware capabilities—e.g., radio-frequency tags, large interactive screens, and information appliances—has led to a further realization that engineers, who know about hardware, software, and electronics are needed to configure, assemble, and program the consumer electronics and other devices to be able to communicate with each other (often referred to as middleware).

### 1.3.2 Working together as a multidisciplinary team

Bringing together so many people with different backgrounds and training has meant many more ideas being generated, new methods being developed, and more creative and original designs being produced. However, the downside is the costs involved. The more people there are with different backgrounds in a design team, the more difficult it can be to communicate and progress forward the designs being generated. Why? People with different backgrounds have different perspectives and ways of seeing and talking about the world (see Figure 1.4). What one person values as important others may not even see (Kim, 1990). Similarly, a computer scientist's understanding of the term representation is often very different from a graphic designer's or a psychologist's.

![Figure 1.4](image) Four different team members looking at the same square, but each seeing it quite differently.
10 Chapter 1 What is interaction design?

What this means in practice is that confusion, misunderstanding, and communication breakdowns can often surface in a team. The various team members may have different ways of talking about design and may use the same terms to mean quite different things. Other problems can arise when a group of people is "thrown" together who have not worked as a team. For example, the Philips Vision of the Future Project found that its multidisciplinary teams—who were responsible for developing ideas and products for the future—experienced a number of difficulties, namely, that project team members did not always have a clear idea of who needed what information, when, and in what form (Lambourne et al., 1997).

**ACTIVITY 1.2** In practice, the makeup of a given design team depends on the kind of interactive product being built. Who do you think would need to be involved in developing:

(a) a public kiosk providing information about the exhibits available in a science museum?

(b) an interactive educational website to accompany a TV series?

**Comment**

Each team will need a number of different people with different skill sets. For example, the first interactive product would need:

(a) graphic and interaction designers, museum curators, educational advisors, software engineers, software designers, usability engineers, ergonomists

The second project would need:

(b) TV producers, graphic and interaction designers, teachers, video experts, software engineers, software designers, usability engineers

In addition, as both systems are being developed for use by the general public, representative users, such as school children and parents, should be involved.

In practice, design teams often end up being quite large, especially if they are working on a big project to meet a fixed deadline. For example, it is common to find teams of fifteen people or more working on a website project for an extensive period of time, like six months. This means that a number of people from each area of expertise are likely to be working as part of the project team.

### 1.3.3 Interaction design in business

Interaction design is big business. In particular, website consultants, start-up companies, and mobile computing industries have all realized its pivotal role in successful interactive products. To get noticed in the highly competitive field of web products requires standing out. Being able to say that your product is easy and effective to use is seen as central to this. Marketing departments are realizing how branding, the number of hits, customer return rate, and customer satisfaction are greatly affected by the usability of a website. Furthermore, the presence or absence of good interaction design can make or break a company.
One infamous dot.com fashion clothes company that failed to appreciate the importance of good interaction design paid heavily for its oversight, becoming bankrupt within a few months of going public. Their approach had been to go for an "all singing and all dancing," glossy 3D graphical interface. One of the problems with this was that it required several minutes to download. Furthermore, it often took more than 20 minutes to place an order by going through a painfully long and slow process of filling out an online form—only to discover that the order was not successful. Customers simply got frustrated with the site and never returned.

In response to the growing demand for interaction design, an increasing number of consultancies are establishing themselves as interaction design experts. One such company is Swim, set up by Gitta Salomon to assist clients with the design of interactive products (see the interview with her at the end of this chapter). She points out how often companies realize the importance of interaction design but don't know how to do it themselves. So they get in touch with companies, like Swim, with their partially developed products and ask them for help. This can come in the form of an expert "crit" in which a detailed review of the usability and design of the product is given (for more on expert evaluation, see Chapter 13). More extensively, it can involve helping clients create their products.

Another established design company that practices interaction design is IDEO, which now has many branches worldwide. Drawing on over 20 years of experience in the area, they design products, services, and environments for other companies, pioneering new user experiences (Spreenberg et al., 1995). They have developed

---

**BOX 1.2 What's In a Name? From Interface Designers to Information Architects**

Ten years ago, when a company wanted to develop an interface for an interactive product it advertised for interface designers. Such professionals were primarily involved in the design and evaluation of widgets for desktop applications. Now that the potential range of interactive products has greatly diversified, coupled with the growing realization of the importance of getting the interface right, a number of other job descriptions have begun to emerge. These include:

- **interactive/interaction designers** (people involved in the design of all the interactive aspects of a product, not just the graphic design of an interface)
- **usability engineers** (people who focus on evaluating products, using usability methods and principles)
- **web designers** (people who develop and create the visual design of websites, such as layouts)
- **information architects** (people who come up with ideas of how to plan and structure interactive products, especially websites)
- **user-experience designers** (people who do all the above but who may also carry out field studies to inform the design of products)

---

²This happened before the dot.com crash in 2001.
Figure 1.5 An innovative product developed by IDEO: Scout Modo, a wireless handheld device delivering up-to-date information about what’s going on in a city.

thousands of products for numerous clients, each time following their particular brand of user-centered design (see Figure 1.5).

### 1.4 What is involved in the process of interaction design?

Essentially, the process of interaction design involves four basic activities:

1. Identifying needs and establishing requirements.
2. Developing alternative designs that meet those requirements.
3. Building interactive versions of the designs so that they can be communicated and assessed.
4. Evaluating what is being built throughout the process.

These activities are intended to inform one another and to be repeated. For example, measuring the usability of what has been built in terms of whether it is easy to use provides feedback that certain changes must be made or that certain requirements have not yet been met.

Evaluating what has been built is very much at the heart of interaction design. Its focus is on ensuring that the product is usable. It is usually addressed through a user-centered approach to design, which, as the name suggests, seeks to involve users throughout the design process. There are many different ways of achieving this: for example, through observing users, talking to them, interviewing them, testing them using performance tasks, modeling their performance, asking them to fill
in questionnaires, and even asking them to become co-designers. The findings from the different ways of engaging and eliciting knowledge from users are then interpreted with respect to ongoing design activities (we give more detail about all these aspects of evaluation in Chapters 10–14).

Equally important as involving users in evaluating an interactive product is understanding what people currently do. This form of research should take place before building any interactive product. Chapters 3, 4, and 5 cover a lot of this ground by explaining in detail how people act and interact with one another, with information, and with various technologies, together with describing their strengths and weaknesses. Such knowledge can greatly help designers determine which solutions to choose from the many design alternatives available and how to develop and test these further. Chapter 7 describes how an understanding of users' needs can be translated to requirements, while Chapter 9 explains how to involve users effectively in the design process.

A main reason for having a better understanding of users is that different users have different needs and interactive products need to be designed accordingly. For example, children have different expectations about how they want to learn or play from adults. They may find having interactive quizzes and cartoon characters helping them along to be highly motivating, whereas most adults find them annoying. Conversely, adults often like talking-heads discussions about topics, but children find them boring. Just as everyday objects like clothes, food, and games are designed differently for children, teenagers, and adults, so, too, must interactive products be designed to match the needs of different kinds of users.

In addition to the four basic activities of design, there are three key characteristics of the interaction design process:

1. Users should be involved through the development of the project.
2. Specific usability and user experience goals should be identified, clearly documented, and agreed upon at the beginning of the project.
3. Iteration through the four activities is inevitable.

We have already mentioned the importance of involving users and will return to this topic throughout the book. Iterative design will also be addressed later when we talk about the various design and evaluation methods by which this can be achieved. In the next section we describe usability and user experience goals.

### 1.5 The goals of interaction design

Part of the process of understanding users' needs, with respect to designing an interactive system to support them, is to be clear about your primary objective. Is it to design a very efficient system that will allow users to be highly productive in their work, or is it to design a system that will be challenging and motivating so that it supports effective learning, or is it something else? We call these top-level concerns usability goals and user experience goals. The two differ in terms of how they are operationalized, i.e., how they can be met and through what means. Usability
goals are concerned with meeting specific usability criteria (e.g., efficiency) and user experience goals are largely concerned with explicating the quality of the user experience (e.g., to be aesthetically pleasing).

1.5.1 Usability goals

To recap, usability is generally regarded as ensuring that interactive products are easy to learn, effective to use, and enjoyable from the user's perspective. It involves optimizing the interactions people have with interactive products to enable them to carry out their activities at work, school, and in their everyday life. More specifically, usability is broken down into the following goals:

- effective to use (effectiveness)
- efficient to use (efficiency)
  - safe to use (safety)
  - have good utility (utility)
- easy to learn (learnability)
- easy to remember how to use (memorability)

For each goal, we describe it in more detail and provide a key question.

Effectiveness is a very general goal and refers to how good a system is at doing what it is supposed to do.

Question: Is the system capable of allowing people to learn well, carry out their work efficiently, access the information they need, buy the goods they want, and so on?

Efficiency refers to the way a system supports users in carrying out their tasks. The answering machine described at the beginning of the chapter was considered efficient in that it let the user carry out common tasks (e.g., listening to messages) through a minimal number of steps. In contrast, the voice-mail system was considered inefficient because it required the user to carry out many steps and learn an arbitrary set of sequences for the same common task. This implies that an efficient way of supporting common tasks is to let the user use single button or key presses. An example of where this kind of efficiency mechanism has been effectively employed is in e-tailing. Once users have entered all the necessary personal details on an e-commerce site to make a purchase, they can let the site save all their personal details. Then, if they want to make another purchase at that site, they don't have to re-enter all their personal details again. A clever mechanism patented by Amazon.com is the one-click option, which requires users only to click a single button when they want to make another purchase.

Question: Once users have learned how to use a system to carry out their tasks, can they sustain a high level of productivity?

Safety involves protecting the user from dangerous conditions and undesirable situations. In relation to the first ergonomic aspect, it refers to the external conditions where people work. For example, where there are hazardous conditions—like X-ray machines or chemical plants—operators should be able to interact with and control computer-based systems remotely. The second aspect refers to helping any
kind of user in any kind of situation avoid the dangers of carrying out unwanted actions accidentally. It also refers to the perceived fears users might have of the consequences of making errors and how this affects their behavior. To make computer-based systems safer in this sense involves (i) preventing the user from making serious errors by reducing the risk of wrong keys/buttons being mistakenly activated (an example is not placing the quit or delete-file command right next to the save command on a menu) and (ii) providing users with various means of recovery should they make errors. Safe interactive systems should engender confidence and allow the user the opportunity to explore the interface to try out new operations (see Figure 1.6a). Other safety mechanisms include undo facilities and

![Figure 1.6](image)

**Figure 1.6** (a) A safe and an unsafe menu. Which is which and why? (b) Warning dialog message from Eudora.
confirmeratory dialog boxes that give users another chance to consider their intentions (a well-known example used in e-mail applications is the appearance of a dialog box, after the user has highlighted messages to be deleted, saying: "Are you sure you want to delete all these messages?" See Figure 1.6(b)).

**Question:** Does the system prevent users from making serious errors and, if they do make an error, does it permit them to recover easily?

**Utility** refers to the extent to which the system provides the right kind of functionality so that users can do what they need or want to do. An example of a system with high utility is an accounting software package providing a powerful computational tool that accountants can use to work out tax returns. A system with low utility is a software drawing tool that does not allow users to draw free-hand but forces them to use a mouse to create their drawings, using only polygon shapes.

**Question:** Does the system provide an appropriate set of functions that enable users to carry out all their tasks in the way they want to do them?

**Learnability** refers to how easy a system is to learn to use. It is well known that people don't like spending a long time learning how to use a system. They want to get started straight away and become competent at carrying out tasks without too much effort. This is especially so for interactive products intended for everyday use (e.g., interactive TV, email) and those used only infrequently (e.g., videoconferencing). To a certain extent, people are prepared to spend longer learning more complex systems that provide a wider range of functionality (e.g., web authoring tools, word processors). In these situations, CD-ROM and online tutorials can help by providing interactive step-by-step material with hands-on exercises. However, many people find these tedious and often difficult to relate to the tasks they want to do.

---

**BOX 1.3 The Ten-Minute Rule**

A criterion for assessing whether a system is easy to learn is to apply the "ten-minute rule" (Nelson, 1980). It proposes that novice users should be able to learn how to use a system in under 10 minutes. If not, the system fails. As pointed out by Rubinstein and Hersh (1984), many computer systems do not meet this criterion. To make systems easy to learn, they suggest that designers capitalize on people's existing knowledge: "A computer system for architects is not expected to teach architecture. Quite the reverse: the ten-minute rule requires that what an architect already knows be helpful in learning to use the architecture system." (Rubinstein and Hersh, 1984 p. 9).

**When is the ten-minute rule not appropriate?**

The ten-minute rule is a useful rule of thumb for evaluating many kinds of systems. However, it is inappropriate for using with complex systems, where it would be difficult and reckless to think that a user could learn how to use them in under ten minutes. For example, would you feel safe knowing that the pilots flying your plane had only spent ten minutes learning how to use the various devices in the cockpit? You would expect them to have spent considerable time (in addition to the years of training to become a pilot) thoroughly learning how to use the array of controls and displays for that particular kind of plane and what to do if any of them malfunction. Likewise, it is unrealistic to assume that ten minutes is enough to learn a system that provides diverse functionality (e.g., a word processor) or that needs high levels of skill to use (e.g., a video game).
accomplish. A key concern is determining how much time users are prepared to spend learning a system. There seems little point in developing a range of functionality if the majority of users are unable or not prepared to spend time learning how to use it.

**Question:** How easy is it and how long does it take (i) to get started using a system to perform core tasks and (ii) to learn the range of operations to perform a wider set of tasks?

Memorability refers to how easy a system is to remember how to use, once learned. This is especially important for interactive systems that are used infrequently. If users haven't used a system or an operation for a few months or longer, they should be able to remember or at least rapidly be reminded how to use it. Users shouldn't have to keep relearning how to carry out tasks. Unfortunately, this tends to happen when the operations required to be learned are obscure, illogical, or poorly sequenced. Users need to be helped to remember how to do tasks. There are many ways of designing the interaction to support this. For example, users can be helped to remember the sequence of operations at different stages of a task through meaningful icons, command names, and menu options. Also, structuring options and icons so they are placed in relevant categories of options (e.g., placing all the drawing tools in the same place on the screen) can help the user remember where to look to find a particular tool at a given stage of a task.

**Question:** What kinds of interface support have been provided to help users remember how to carry out tasks, especially for systems and operations that are used infrequently?

**Activity 1.3** How long do you think it should take to learn how to use the following interactive products and how long does it actually take most people to learn them? How memorable are they?

(a) using a VCR to play a video
(b) using a VCR to pre-record two programs
(c) using an authoring tool to create a website

**Comment**

(a) To play a video should be as simple as turning the radio on, should take less than 30 seconds to work out, and then should be straightforward to do subsequently. Most people are able to fathom how to play a video. However, some systems require the user to switch to the "video" channel using one or two remote control devices, selecting from a choice of 50 or more channels. Other settings may also need to be configured before the video will play. Most people are able to remember how to play a video once they have used a particular VCR.

(b) This is a more complex operation and should take a couple of minutes to learn how to do and to check that the programming is correct. In reality, many VCRs are so poorly designed that 80% of the population is unable to accomplish this task, despite several attempts. Very few people remember how to pre-record a program, largely because the interaction required to do this is poorly designed, with poor or no feedback, and is often illogical from the user's perspective. Of those, only a few will bother to go through the manual again.
What is interaction design?

(c) A well-designed authoring tool should let the user create a basic page in about 20 minutes. Learning the full range of operations and possibilities is likely to take much longer, possibly a few days. In reality, there are some good authoring tools that allow the user to get started straight away, providing templates that they can adapt. Most users will extend their repertoire, taking another hour or so to learn more functions. However, very few people actually learn to use the full range of functions provided by the authoring tool. Users will tend to remember frequently used operations (e.g., cut and paste, inserting images), especially if they are consistent with the way they are carried out in other software applications. However, less frequently used operations may need to be relearned (e.g., formatting tables).

The usability goals discussed so far are well suited to the design of business systems intended to support working practices. In particular, they are highly relevant for companies and organizations who are introducing or updating applications running on desktop and networked systems—that are intended to increase productivity by improving and enhancing how work gets done. As well as couching them in terms of specific questions, usability goals are turned into usability criteria. These are specific objectives that enable the usability of a product to be assessed in terms of how it can improve (or not) a user's performance. Examples of commonly used usability criteria are time to complete a task (efficiency), time to learn a task (learnability), and the number of errors made when carrying out a given task over time (memorability).

1.5.2 User experience goals

The realization that new technologies are offering increasing opportunities for supporting people in their everyday lives has led researchers and practitioners to consider further goals. The emergence of technologies (e.g., virtual reality, the web, mobile computing) in a diversity of application areas (e.g., entertainment, education, home, public areas) has brought about a much wider set of concerns. As well as focusing primarily on improving efficiency and productivity at work, interaction design is increasingly concerning itself with creating systems that are:

- satisfying
- enjoyable
- fun
- entertaining
- helpful
- motivating
- aesthetically pleasing
- supportive of creativity
- rewarding
- emotionally fulfilling
The goals of designing interactive products to be fun, enjoyable, pleasurable, aesthetically pleasing and so on are concerned primarily with the user experience. By this we mean what the interaction with the system *feels* like to the users. This involves explicating the nature of the user experience in subjective terms. For example, a new software package for children to create their own music may be designed with the primary objectives of being fun and entertaining. Hence, user experience goals differ from the more objective usability goals in that they are concerned with how users experience an interactive product from their perspective, rather than assessing how useful or productive a system is from its own perspective. The relationship between the two is shown in Figure 1.7.

Much of the work on enjoyment, fun, etc., has been carried out in the entertainment and computer games industry, which has a vested interest in understanding the role of pleasure in considerable detail. Aspects that have been described as contributing to pleasure include: attention, pace, play, interactivity, conscious and unconscious control, engagement, and style of narrative. It has even been suggested that in these contexts, it might be interesting to build systems that are *non-easy* to use, providing opportunities for quite different user experiences from those designed based on usability goals (Frohlich and Murphy, 1999). Interacting with a virtual representation using a physical device (e.g., banging a plastic...
hammer to hit a virtual nail represented on the computer screen) compared with using a more efficient way to do the same thing (e.g., selecting an option using command keys) may require more effort but could, conversely, result in a more enjoyable and fun experience.

Recognizing and understanding the trade-offs between usability and user experience goals is important. In particular, this enables designers to become aware of the consequences of pursuing different combinations of them in relation to fulfilling different users' needs. Obviously, not all of the usability goals and user experience goals apply to every interactive product being developed. Some combinations will also be incompatible. For example, it may not be possible or desirable to design a process control system that is both safe and fun. As stressed throughout this chapter, what is important depends on the use context, the task at hand, and who the intended users are.

Below are a number of proposed interactive products. What do you think are the key usability goals and user experience goals for each of them?

(a) a mobile device that allows young children to communicate with each other and play collaborative games
(b) a video and computer conferencing system that allows students to learn at home
(c) an Internet application that allows the general public to access their medical records via interactive TV
(d) a CAD system for architects and engineers
(e) an online community that provides support for people who have recently been bereaved

Comment

(a) Such a collaborative device should be easy to use, effective, efficient, easy to learn and use, fun and entertaining.
(b) Such a learning device should be easy to learn, easy to use, effective, motivating and rewarding.
(c) Such a personal system needs to be safe, easy to use and remember how to use, efficient and effective.
(d) Such a tool needs to be easy to learn, easy to remember, have good utility, be safe, efficient, effective, support creativity and be aesthetically pleasing.
(e) Such a system needs to be easy to learn, easy to use, motivating, emotionally satisfying and rewarding.

1.6 More on usability: design and usability principles

Another way of conceptualizing usability is in terms of design principles. These are generalizable abstractions intended to orient designers towards thinking about different aspects of their designs. A well-known example is feedback: systems should be designed to provide adequate feedback to the users to ensure they know what to
do next in their tasks. Design principles are derived from a mix of theory-based knowledge, experience, and common sense. They tend to be written in a prescriptive manner, suggesting to designers what to provide and what to avoid at the interface—if you like, the do's and don'ts of interaction design. More specifically, they are intended to help designers explain and improve the design (Thimbleby, 1990). However, they are not intended to specify how to design an actual interface (e.g., telling the designer how to design a particular icon or how to structure a web portal) but act more like a set of reminders to designers, ensuring that they have provided certain things at the interface.

A number of design principles have been promoted. The best known are concerned with how to determine what users should see and do when carrying out their tasks using an interactive product. Here we briefly describe the most common ones: visibility, feedback, constraints, mapping, consistency, and affordances. Each of these has been written about extensively by Don Norman (1988) in his bestseller The Design of Everyday Things.

**Visibility**  The importance of visibility is exemplified by our two contrasting examples at the beginning of the chapter. The voice-mail system made the presence and number of waiting messages invisible, while the answer machine made both aspects highly visible. The more visible functions are, the more likely users will be able to know what to do next. In contrast, when functions are "out of sight," it makes them more difficult to find and know how to use. Norman (1988) describes the controls of a car to emphasize this point. The controls for different operations are clearly visible (e.g., indicators, headlights, horn, hazard warning lights), indicating what can be done. The relationship between the way the controls have been positioned in the car and what they do makes it easy for the driver to find the appropriate control for the task at hand.

**Feedback**  Related to the concept of visibility is feedback. This is best illustrated by an analogy to what everyday life would be like without it. Imagine trying to play a guitar, slice bread using a knife, or write using a pen if none of the actions produced any effect for several seconds. There would be an unbearable delay before the music was produced, the bread was cut, or the words appeared on the paper, making it almost impossible for the person to continue with the next strum, saw, or stroke.

Feedback is about sending back information about what action has been done and what has been accomplished, allowing the person to continue with the activity. Various kinds of feedback are available for interaction design—audio, tactile, verbal, visual, and combinations of these. Deciding which combinations are appropriate for different kinds of activities and interactivities is central. Using feedback in the right way can also provide the necessary visibility for user interaction.

**Constraints**  The design concept of constraining refers to determining ways of restricting the kind of user interaction that can take place at a given moment. There are various ways this can be achieved. A common design practice in graphical user interfaces is to deactivate certain menu options by shading them, thereby restrict-
What is interaction design?

Figure 1.8 A menu illustrating restricted availability of options as an example of logical constraining. Shaded areas indicate deactivated options.

ing the user to only actions permissible at that stage of the activity (see Figure 1.8). One of the advantages of this form of constraining is it prevents the user from selecting incorrect options and thereby reduces the chance of making a mistake. The use of different kinds of graphical representations can also constrain a person's interpretation of a problem or information space. For example, flow chart diagrams show which objects are related to which, thereby constraining the way the information can be perceived.

Norman (1999) classifies constraints into three categories: physical, logical, and cultural. Physical constraints refer to the way physical objects restrict the movement of things. For example, the way an external disk can be placed into a disk drive is physically constrained by its shape and size, so that it can be inserted in only one way. Likewise, keys on a pad can usually be pressed in only one way.

Logical constraints rely on people's understanding of the way the world works (cf. the marbles answering machine design). They rely on people's common-sense reasoning about actions and their consequences. Picking up a physical marble and placing it in another location on the phone would be expected by most people to
trigger something else to happen. Making actions and their effects obvious enables people to logically deduce what further actions are required. Disabling menu options when not appropriate for the task in hand provides logical constraining. It allows users to reason why (or why not) they have been designed this way and what options are available.

Cultural constraints rely on learned conventions, like the use of red for warning, the use of certain kinds of audio signals for danger, and the use of the smiley face to represent happy emotions. Most cultural constraints are arbitrary in the sense that their relationship with what is being represented is abstract, and could have equally evolved to be represented in another form (e.g., the use of yellow instead of red for warning). Accordingly, they have to be learned. Once learned and accepted by a cultural group, they become universally accepted conventions. Two universally accepted interface conventions are the use of windowing for displaying information and the use of icons on the desktop to represent operations and documents.

**Mapping** This refers to the relationship between controls and their effects in the world. Nearly all artifacts need some kind of mapping between controls and effects, whether it is a flashlight, car, power plant, or cockpit. An example of a good mapping between control and effect is the up and down arrows used to represent the up and down movement of the cursor, respectively, on a computer keyboard. The mapping of the relative position of controls and their effects is also important. Consider the various musical playing devices (e.g., MP3, CD player, tape recorder). How are the controls of playing, rewinding, and fast forward mapped onto the desired effects? They usually follow a common convention of providing a sequence of buttons, with the play button in the middle, the rewind button on the left and the fast-forward on the right. This configuration maps directly onto the directionality of the actions (see Figure 1.9a). Imagine how difficult it would be if the mappings in Figure 1.9b were used. Look at Figure 1.10 and determine from the various mappings which is good and which would cause problems to the person using it.
What is interaction design?

Consistency  This refers to designing interfaces to have similar operations and use similar elements for achieving similar tasks. In particular, a consistent interface is one that follows rules, such as using the same operation to select all objects. For example, a consistent operation is using the same input action to highlight any graphical object at the interface, such as always clicking the left mouse button. Inconsistent interfaces, on the other hand, allow exceptions to a rule. An example of this is where certain graphical objects (e.g., email messages presented in a table) can be highlighted only by using the right mouse button, while all other operations are highlighted using the left button. A problem with this kind of inconsistency is that it is quite arbitrary, making it difficult for users to remember and making the users more prone to mistakes.

One of the benefits of consistent interfaces, therefore, is that they are easier to learn and use. Users have to learn only a single mode of operation that is applicable to all objects. This principle works well for simple interfaces with limited operations, like a mini CD player with a small number of operations mapped onto separate buttons. Here, all the user has to do is learn what each button represents and select accordingly. However, it can be more problematic to apply the concept of consistency to more complex interfaces, especially when many different operations need to be designed for. For example, consider how to design an interface for an application that offers hundreds of operations (e.g., a word-processing application). There is simply not enough space for a thousand buttons, each of which maps onto an individual operation. Even if there were, it would be extremely difficult and time-consuming for the user to search through them all to find the desired operation.

A much more effective design solution is to create categories of commands that can be mapped into subsets of operations. For the word-processing application, the hundreds of operations available are categorized into subsets of different menus. All commands that are concerned with file operations (e.g., save, open, close) are placed together in the same file menu. Likewise, all commands concerned with formatting text are placed in a format menu. Selecting an operation then becomes a matter of homing in on the right category (menu) of options and scanning it for the desired one, rather than scrolling through one long list. However, the consistency rule of having a visible one-to-one mapping between command and operation is broken. Operations are not immediately visible at the interface, but are instead hidden under different categories of menus. Furthermore, some menu items are immediately visible, when a top-level menu is first pulled down, while others remain hidden until the visible items are scrolled over. Thus, users need to learn what items are visible in each menu category and which are hidden in submenus.

The way the items are divided between the categories of menu items can also appear inconsistent to users. Various operations appear in menus where they do not belong. For example, the sorting operation (very useful for listing references or names in alphabetical order) in Microsoft Word 2001 is in the Table menu (the Mac Version). In the previous Word 98 version, it was in both the Tools and Table menus. I always thought of it as a Tool operation (like Word Count), and became very frustrated to discover that as a default for Word 2001 it is only in the Table menu. This makes it inconsistent for me in two ways: (i) with the previous version and (ii) in the category it has been placed. Of course, I can customize the new ver-
More on usability: design and usability principles

1.6

... so that the menus are structured in the way I think they should be, but this all takes considerable time (especially when I use different machines at work, home, and when travelling).

Another problem with consistency is determining what aspect of an interface to make consistent with what else. There are often many choices, some of which can be inconsistent with other aspects of the interface or ways of carrying out actions. Consider the design problem of developing a mechanism to let users lock their files on a shared server. Should the designer try to design it to be consistent with the way people lock things in the outside world (called external consistency) or with the way they lock objects in the existing system (called internal consistency)? However, there are many different ways of locking objects in the physical world (e.g., placing in a safe, using a padlock, using a key, using a child safety lock), just as there are different ways of locking electronically (e.g., using PIN numbers, passwords, permissions, moving the physical switches on floppy disks). The problem facing designers is knowing which one to be consistent with.

**Affordance** is a term used to refer to an attribute of an object that allows people to know how to use it. For example, a mouse button invites pushing (in so doing activating clicking) by the way it is physically constrained in its plastic shell. At a very simple level, to afford means "to give a clue" (Norman, 1988). When the affordances of a physical object are perceptually obvious it is easy to know how to interact with it. For example, a door handle affords pulling, a cup handle affords grasping, and a mouse button affords pushing. Norman introduced this concept in the late '80s in his discussion of the design of everyday objects. Since then, it has been much popularized, being used to describe how interface objects should be designed so that they make obvious what can be done to them. For example, graphical elements like buttons, icons, links, and scroll bars are talked about with respect to how to make it appear obvious how they should be used: icons should be designed to afford clicking, scroll bars to afford moving up and down, buttons to afford pushing.

Unfortunately, the term affordance has become rather a catch-all phrase, losing much of its potency as a design principle. Norman (1999), who was largely responsible for originally promoting the concept in his book *The Design of Everyday Things* (1988), now despair of the way it has come to be used in common parlance:

"Zput an affordance there," a participant would say, "I wonder if the object affords clicking..." affordances this, affordances that. And no data, just opinion. Yikes! What had I unleashed upon the world? Norman's (1999) reaction to a recent CHI-Web discussion.

He has since tried to clarify his argument about the utility of the concept by saying there are two kinds of affordance: perceived and real. Physical objects are said to have real affordances, like grasping, that are perceptually obvious and do not have to be learned. In contrast, user interfaces that are screen-based are virtual and do not have these kinds of real affordances. Using this distinction, he argues that it does not make sense to try to design for real affordances at the interface—except when designing physical devices, like control consoles, where affordances like pulling and pressing are helpful in guiding the user to know what to do. Alternatively, screen-based
interfaces are better conceptualized as perceived affordances, which are essentially learned conventions. In conclusion, Norman argues that other design concepts—conventions, feedback and cultural and logical constraints—are far more useful for helping designers develop graphical user interfaces.

### 1.6.1 Heuristics and usability principles

When design principles are used in practice they are commonly referred to as heuristics. This term emphasizes that something has to be done with them when they are applied to a given problem. In particular, they need to be interpreted in the design context, drawing on past experience of, for example, how to design feedback and what it means for something to be consistent.

Another form of guidance is usability principles. An example is "speak the user's language." These are quite similar to design principles, except that they tend to be more prescriptive. In addition, whereas design principles tend to be used mainly for informing a design, usability principles are used mostly as the basis for evaluating prototypes and existing systems. In particular, they provide the framework for heuristic evaluation (see Chapter 13). They, too, are called heuristics when used as part of
an evaluation. Below are the ten main usability principles, developed by Nielsen (2001) and his colleagues. Note how some of them overlap with the design principles.

1. Visibility of system status — always keep users informed about what is going on, through providing appropriate feedback within reasonable time
2. Match between system and the real world — speak the users’ language, using words, phrases and concepts familiar to the user, rather than system-oriented terms
3. User control and freedom — provide ways of allowing users to easily escape from places they unexpectedly find themselves, by using clearly marked 'emergency exits'
4. Consistency and standards — avoid making users wonder whether different words, situations, or actions mean the same thing
5. Help users recognize, diagnose, and recover from errors — use plain language to describe the nature of the problem and suggest a way of solving it
6. Error prevention — where possible prevent errors occurring in the first place
7. Recognition rather than recall — make objects, actions, and options visible
8. Flexibility and efficiency of use — provide accelerators that are invisible to novice users, but allow more experienced users to carry out tasks more quickly
9. Aesthetic and minimalist design — avoid using information that is irrelevant or rarely needed
10. Help and documentation — provide information that can be easily searched and provides help in a set of concrete steps that can easily be followed

**Activity 1.5**

One of the main design principles which Nielsen has proselytized, especially for website design, is simplicity. He proposes that designers go through all of their design elements and remove them one by one. If a design works just as well without an element, then remove it. Do you think this is a good design principle? If you have your own website, try doing this and seeing what happens. At what point does the interaction break down?

**Comment**

Simplicity is certainly an important design principle. Many designers try to cram too much into a screenful of space, making it unwieldy for people to find what they are interested in. Removing design elements to see what can be discarded without affecting the overall function of the website can be a salutary lesson. Unnecessary icons, buttons, boxes, lines, graphics, shading, and text can be stripped, leaving a cleaner, crisper, and easier-to-navigate website. However, a certain amount of graphics, shading, coloring, and formatting can make a site aesthetically pleasing and enjoyable to use. Plain vanilla sites with just lists of text and a few hyperlinks may not be as appealing and may put certain visitors off returning. The key is getting the right balance between aesthetic appeal and the right amount and kind of information per page.

Design and usability principles have also been operationalized into even more specific prescriptions called rules. These are guidelines that should be followed. An example is "always place the quit or exit button at the bottom of the first menu list in an application."
Assignment

This assignment is intended for you to put into practice what you have read about in this chapter. Specifically, the objective is to enable you to define usability and user experience goals and to use design and usability principles for evaluating the usability of an interactive product.

Find a handheld device (e.g., remote control, handheld computer, or cell phone) and examine how it has been designed, paying particular attention to how the user is meant to interact with it.

(a) From your first impressions, write down what first comes to mind as to what is good and bad about the way the device works. Then list (i) its functionality and (ii) the range of tasks a typical user would want to do using it. Is the functionality greater, equal, or less than what the user wants to do?

(b) Based on your reading of this chapter and any other material you have come across, compile your own set of usability and user experience goals that you think will be
In this chapter we have looked at what interaction design is and how it has evolved. We examined briefly its makeup and the various processes involved. We pointed out how the notion of usability is fundamental to interaction design. This was explained in some detail, describing what it is and how it is operationalized to assess the appropriateness, effectiveness, and quality of interactive products. A number of high-level design principles were also introduced that provide different forms of guidance for interaction design.
Chapter 1  What is interaction design?

Key points

- Interaction design is concerned with designing interactive products to support people in their everyday and working lives.
- Interaction design is multidisciplinary, involving many inputs from wide-reaching disciplines and fields.
- Interaction design is now big business: many companies want it but don’t know how to do it.
- Optimizing the interaction between users and interactive products requires taking into account a number of interdependent factors, including context of use, type of task, and kind of user.
- Interactive products need to be designed to match usability goals like ease of use and learning.
- User experience goals are concerned with creating systems that enhance the user experience in terms of making it enjoyable, fun, helpful, motivating, and pleasurable.
- Design and usability principles, like feedback and simplicity, are useful heuristics for analyzing and evaluating aspects of an interactive product.

Further reading

Here we recommend a few seminal readings. A more comprehensive list of useful books, articles, websites, videos, and other material can be found at our website.

WINOGRAD, T. (1997) From computing machinery to interaction design. In P. Denning and R. Metcalfe (eds.) Beyond Calculation: the Next Fifty Years of Computing. New York: Springer-Verlag, 149–162. Terry Winograd provides an overview of how interaction design has emerged as a new area, explaining how it does not fit into any existing design or computing fields. He describes the new demands and challenges facing the profession.

NORMAN, D. (1988) The Design of Everyday Things. New York: Doubleday, (especially Chapter 1). Norman’s writing is highly accessible and enjoyable to read. He writes extensively about the design and usability of everyday objects like doors, faucets, and fridges. These examples provide much food for thought in relation to designing interfaces. The Voyager CD-ROM (sadly, now no longer published) of his collected works provides additional videos and animations that illustrate in an entertaining way many of the problems, design ideas and issues raised in the text.

NORMAN, D. (1999) ACM Interactions Magazine, May/June, 38–42. Affordances, conventions and design. This is a short and thought-provoking critique of design principles.

GRUDIN, J. (1990) The computer reaches out: the historical continuity of interface design. In CHI’90 Proc. 261–268. Grudin, J. (1989) The case against user interface consistency. Communications of the ACM, 32(10), 1164–1173. Jonathan Grudin is a prolific writer and many of his earlier works provide thought-provoking and well documented accounts of topical issues in HCI. The first paper talks about how interface design has expanded to cover many more aspects in its relatively short history. The second paper, considered a classic of its time, discusses why the concept of consistency—which had been universally accepted as good interface design up until then—was in fact highly problematic.

Interactions, January/February 2000, ACM. This special issue provides a collection of visions, critiques, and sound bites on the achievements and future of HCI from a number of researchers, designers, and practitioners.

IDEO provides a well illustrated online archive of a range of interactive products it has designed. (see www.ideo.com)
Interview with Gitta Salomon

Gitta Salomon is a consultant interaction designer. She founded Swim Interaction Design Studio (swimstudio.com) in 1996 as a consultancy company to assist clients with the design of interactive products. Recently, many of her clients have included start-up companies, developing web-based and other products, who realize the importance of interaction design in ensuring their products are successful but don't know how to do this. Often they get in touch with Swim with partially developed products and ask for help with their interaction design. Swim has consulted for a range of clients, including Apple Computer, Nike, IBM,DoubleClick, Webex, and RioPort.

YR: What is your approach to interaction design?

GS: I've devised my own definition: interaction design is the design of products that reveal themselves over time. Users don't necessarily see all the functionality in interactive products when they first look at them. For example, the first screen you see on a cell phone doesn't show you everything you can do with it. As you use it, additional functionality is revealed to you. Same thing with a web-based application or a Window's application—as you use them you find yourself in different states and suddenly you can do different things. This idea of revealing over time is possible because there is a microprocessor behind the product and usually there is also a dynamic display. I believe this definition characterizes the kind of products we work on—which is a very wide range, not just web products.

YR: How would you say interaction design has changed in the years since you started Swim?

GS: I don't think what we do has changed fundamentally, but the time frame for product development is much shorter. And seemingly more people think they want interaction design assistance. That has definitely changed. There are more people who don't necessarily know what interaction design is, but they are calling us and saying "we need it." All of a sudden there is a great deal of focus and money on all of these products that are virtual and computationally based, which require a different type of design thinking.

YR: So what were the kinds of projects you were working on when you first started Swim?

GS: They were less web-centric. There was more software application design and a few hardware/software type things. For the last year and a half the focus shifted to almost exclusively web-based applications. However, these are quite similar to software applications—they just have different implementation constraints. Right at the moment, the hardware/software products are starting to pick up again—it does seem that information appliances are going to take off. The nature of the problems we solve hasn't changed much; it's the platform and associated constraints that change.

YR: What would you say are the biggest challenges facing yourself and other consultants doing interaction design these days?

GS: One of the biggest challenges is remembering that half of what we do is the design work and the other half is the communication of that design work. The clients almost never bridge the gap for us: we need to bridge it. We always have to figure out how to deliver the work so it is going to have impact. We are the ones who need to ensure that the client is going to understand it and know what to do with it. That part of the work is oftentimes the most difficult. It means we've got to figure out what is going on internally with the client and decide how what we deliver will be effective. In some cases you just start seeing there is no place to engage with the client. And I think that is a very difficult problem. Most people right now don't have a product development process. They are just going for it. And we have to figure out how to fit into what is best described as a moving train.

YR: And what do you use when you try to communicate with them? Is it a combination of talking, meetings, and reports?

GS: We do a number of different things. Usually we will give them a written document, like a report or a critique of their product. Sometimes we will give them interactive prototypes in Director or HTML, things that simulate what the product experience would feel like. In the written materials, I
often name the things that we all need to be talking about. Then at least we all have a common terminology to discuss things. It is a measure of our success if they start using the words that we gave them, because we truly have influenced their thinking. A lot of times we will give them a diagram of what their system is like, because nobody has ever visualized it. We serve as the visualizers, taking a random assortment of vaguely defined concepts and giving some shape to them. We will make an artifact, which allows them to say "Yes, it is like that" or "No, it's not like that, it's like this..." Without something to point to they couldn't even say to each other "No, that is not what I mean" because they didn't know if they were talking about the same thing. Many times we'll use schematic diagrams to represent system behavior. Once they have these diagrams then they can say "Oh no, we need all this other stuff in there, we forgot to tell you." It seems that nobody is writing complete lists of functionality, requirements specifications, or complete documentation anymore. This means the product ideas stay in somebody's head until we make them tangible through visualization.

YR. So this communication process is just as important as the ideas?

GS: I think it is, a lot of times.

YR: So, how do you start with a client?

GS: For clients who already have something built, I find that usually the best way for us to get started, is to begin with the client doing a comprehensive demo of their product for us. We will usually spend a whole day collecting information. Besides the demo, they tell us about their target market, competitors, and a whole range of things. It then takes a longer period of time for us to use the product and observe other people using it to get a much broader picture. Because the client's own vision of their product is so narrow, we really have to step back from what they initially tell us.

YR: So do you write notes, and then try and put it together afterwards, or—what?

GS: We use all kinds of things. We use notes and video, and we sit around with tracing paper and marker pens. When reviewing the materials, I often
try and bring them together in some sort of thematic way. It's often mind-boggling to bring a software product that's been thrown together into any kind of coherent framework. It's easy to write a shopping list of observations, but we want to assemble a larger structure and framework and that takes several weeks to construct. We need time to reflect and stew on what was done and what maybe should have been done. We need to highlight the issues and put them into some kind of larger order. If you always operate at a low level of detail, like worrying and critiquing the size of a button, you end up solving only local issues. You never really get to the big interaction design problems of the product, the ones that should be solved first.

**YR:** If you're given a prototype or product to evaluate and you discover that it is really bad, what do you do?

**GS:** Well, I never have the guts to go in and say something is fundamentally flawed. And that's maybe not the best strategy anyway, because it's your word against theirs. Instead, I think it is always about making the case for why something is wrong or flawed. Sometimes I think we are like lawyers. We have to assemble the case for what's wrong with the product. We have to make a convincing argument. A lot of times I think the kind of argumentation we do is very much like what lawyers do.

**YR:** Finally, how do you see interaction design moving in the next five years? More of the same kind of problems with new emerging technologies? Or do you think there are going to be more challenges, especially with the hardware/software integration?

**GS:** I think there will be different constraints as new technologies arise. No matter what we are designing, we have to understand the constraints of the implementation. And yes, different things will happen when we get more into designing hardware/software products. There are different kinds of cost constraints and different kinds of interactions you can do when there is special purpose hardware involved. Whereas designing the interaction for applications requires visual design expertise, designing information appliances or other hardware products requires experience with product design. Definitely, there will be some new challenges.

Hopefully, in the next few years, people will stop looking for interaction design rules. There's been a bit of a push towards making interaction design a science lately. Maybe this has happened because so many people are trying to do it and they don't know where to start because they don't have much experience. I'm hoping people will start understanding that interaction design is a design discipline—that there are some guidelines and ways to do good practice—and creativity combined with analytical thinking are necessary to arrive at good products. And then, even more so than now, it is going to get interesting and be a really exciting time.
Chapter 2

Understanding and conceptualizing interaction

2.1 Introduction

Imagine you have been asked to design an application to let people organize, store, and retrieve their email in a fast, efficient and enjoyable way. What would you do? How would you start? Would you begin by sketching out how the interface might look, work out how the system architecture will be structured, or even just start coding? Alternatively, would you start by asking users about their current experiences of saving email, look at existing email tools and, based on this, begin thinking about why, what, and how you were going to design the application?

Interaction designers would begin by doing the latter. It is important to realize that having a clear understanding of what, why, and how you are going to design something, before writing any code, can save enormous amounts of time and effort later on in the design process. Ill-thought-out ideas, incompatible and unusable designs can be ironed out while it is relatively easy and painless to do. Once ideas are committed to code (which typically takes considerable effort, time, and money), they become much harder to throw away—and much more painful. Such preliminary thinking through of ideas about user needs and what

User needs here are the range of possible requirements, including user wants and experiences.
kinds of designs might be appropriate is, however, a skill that needs to be learned. It is not something that can be done overnight through following a checklist, but requires practice in learning to identify, understand, and examine the issues—just like learning to write an essay or to program. In this chapter we describe what is involved. In particular, we focus on what it takes to understand and conceptualize interaction.

The main aims of this chapter are to:

- Explain what is meant by the problem space.
- Explain how to conceptualize interaction.
- Describe what a conceptual model is and explain the different kinds.
- Discuss the pros and cons of using interface metaphors as conceptual models.
- Debate the pros and cons of using realism versus abstraction at the interface.
- Outline the relationship between conceptual design and physical design.

### 2.2 Understanding the problem space

In the process of creating an interactive product, it can be tempting to begin at the "nuts and bolts" level of the design. By this, we mean working out how to design the physical interface and what interaction styles to use (e.g., whether to use menus, forms, speech, icons, or commands). A problem with trying to solve a design problem beginning at this level is that critical usability goals and user needs may be overlooked. For example, consider the problem of providing drivers with better navigation and traffic information. How might you achieve this? One could tackle the problem by thinking straight away about a good technology or kind of interface to use. For example, one might think that augmented reality, where images are superimposed on objects in the real world (see Figure 2.1 on Color Plate 2), would be appropriate, since it can be useful for integrating additional information with an ongoing activity (e.g., overlaying X-rays on a patient during an operation). In the context of driving, it could be effective for displaying information to drivers who need to find out where they are going and what to do at certain points during their journey. In particular, images of places and directions to follow could be projected inside the car, on the dashboard or rear-view mirror. However, there is a major problem with this proposal: it is likely to be very unsafe. It could easily distract drivers, luring them to switch their attention from the road to where the images were being projected.

A problem in starting to solve a design problem at the physical level, therefore, is that usability goals can be easily overlooked. While it is certainly necessary at some point to decide on the design of physical aspects, it is better to make these kinds of design decisions after understanding the nature of the problem space. By this, we mean conceptualizing what you want to create and articulating why you want to do so. This requires thinking through how your design will support people in their everyday or work activities. In particular, you need to ask yourself whether the interactive product you have in mind will achieve what you hope it will. If so,
how? In the above example, this involves finding out what is problematic with existing forms of navigating while driving (e.g., trying to read maps while moving the steering wheel) and how to ensure that drivers can continue to drive safely without being distracted.

Clarifying your usability and user experience goals is a central part of working out the problem space. This involves making explicit your implicit assumptions and claims. Assumptions that are found to be vague can highlight design ideas that need to be better formulated. The process of going through them can also help to determine relevant user needs for a given activity. In many situations, this involves identifying human activities and interactivities that are problematic and working out how they might be improved through being supported with a different form of interaction. In other situations it can be more speculative, requiring thinking through why a novel and innovative use of a new technology will be potentially useful.

Below is another scenario in which the problem space focuses on solving an identified problem with an existing product. Initial assumptions are presented first, followed by a further explanation of what lies behind these (assumptions are highlighted in italics):

A large software company has decided to develop an upgrade of its web browser. They assume that there is a need for a new one, which has better and more powerful functionality. They begin by carrying out an extensive study of people's actual use of web browsers, talking to lots of different kinds of users and observing them using their browsers. One of their main findings is that many people do not use the bookmarking feature effectively. A common finding is that it is too restrictive and underused. In fathoming why this is the case, it was considered that the process of placing web addresses into hierarchical folders was an inadequate way of supporting the user activity of needing to mark hundreds and sometimes thousands of websites such that any one of them could be easily returned to or forwarded onto other people. An implication of the study was that a new way of saving and retrieving web addresses was needed.

In working out why users find the existing feature of bookmarking cumbersome to use, a further assumption was explicated:

- The existing way of organizing saved (favorite) web addresses into folders is inefficient because it takes too long and is prone to errors.

A number of underlying reasons why this was assumed to be the case were further identified, including:

- It is easy to lose web addresses by placing them accidentally into the wrong folders.
- It is not easy to move web addresses between folders.
- It is not obvious how to move a number of addresses from the saved favorite list into another folder simultaneously.
- It is not obvious how to reorder web addresses once placed in folders.
Based on this analysis, a set of assumptions about the user needs for supporting this activity more effectively were then made. These included:

- *If the bookmarking function was improved users would find it more useful and use it more to organize their web addresses.*
- *Users need a flexible way of organizing web addresses they want to keep for further reference or for sending on to other people.*

**A framework for explicating assumptions**

Reasoning through your assumptions about why something might be a good idea enables you to see the strengths and weaknesses of your proposed design. In so doing, it enables you to be in a better position to commence the design process. We have shown you how to begin this, through operationalizing relevant usability goals. In addition, the following questions provide a useful framework with which to begin thinking through the problem space:

- Are there problems with an existing product? If so, what are they? Why do you think there are problems?
- Why do you think your proposed ideas might be useful? How do you envision people integrating your proposed design with how they currently do things in their everyday or working lives?
- How will your proposed design support people in their activities? In what way does it address an identified problem or extend current ways of doing things? Will it really help?

**Activity 2.1** At the turn of the millennium, WAP-enabled (wireless application protocol) phones came into being, that enabled people to connect to the Internet using them. To begin with, the web-enabled services provided were very primitive, being text-based with limited graphics capabilities. Access was very restricted, with the downloaded information being displayed on a very small LCD screen (see Figure 2.2). Despite this major usability drawback, every telecommunication company saw this technological breakthrough as an opportunity to create innovative applications. A host of new services were explored, including text messaging, online booking of tickets, betting, shopping, viewing movies, stocks and shares, sports events and banking.

What assumptions were made about the proposed services? How reasonable are these assumptions?

**Figure 2.2** An early cell phone display. Text is restricted to three or four lines at a time and scrolls line by line, making reading very cumbersome. Imagine trying to read a page from this book in this way! The newer 3G (third generation) phones have bigger displays, more akin to those provided with handheld computers.
Comment

The problem space for this scenario was very open-ended. There was no identifiable problem that needed to be improved or fixed. Alternatively, the new WAP technology provided opportunities to create new facilities and experiences for people. One of the main assumptions is that people want to be kept informed of up-to-the-minute news (e.g., sports, stocks and share prices) wherever they are. Other assumptions included:

- That people want to be able to decide what to do in an evening while on their way home from work (e.g., checking TV listings, movies, making restaurant reservations).
- That people want to be able to interact with information on the move (e.g., reading email on the train).
- That users are prepared to put up with a very small display and will be happy browsing and interacting with information using a restricted set of commands via a small number of tiny buttons.
- That people will be happy doing things on a mobile phone that they normally do using their PCs (e.g., reading email, surfing the web, playing video games, doing their shopping).

It is reasonable to assume that people want flexibility. They like to be able to find out about news and events wherever they are (just look at the number of people who take a radio with them to a soccer match to find out the scores of other matches being played at the same time). People also like to use their time productively when traveling, as in making phone calls. Thus it is reasonable to assume they would like to read and send email on the move. The most troublesome assumption is whether people are prepared to interact with the range of services proposed using such a restricted mode of interactivity. In particular, it is questionable whether most people are prepared to give up what they have been used to (e.g., large screen estate, ability to type messages using a normal-sized keyboard) for the flexibility of having access to very restricted Internet-based information via a cell phone they can keep in their pocket.

One of the benefits of working through your assumptions for a problem space before building anything is that it can highlight problematic concerns. In so doing, it can identify ideas that need to be reworked, before it becomes too late in the design process to make changes. Having a good understanding of the problem space can also help greatly in formulating what it is you want to design. Another key aspect of conceptualizing the problem space is to think about the overall structure of what will be built and how this will be conveyed to the users. In particular, this involves developing a conceptual model.

2.3 Conceptual models

"The most important thing to design is the user's conceptual model. Everything else should be subordinated to making that model clear, obvious, and substantial. That is almost exactly the opposite of how most software is designed." (David Liddle, 1996, p. 17)
By a conceptual model is meant:

*a description of the proposed system in terms of a set of integrated ideas and concepts about what it should do, behave and look like, that will be understandable by the users in the manner intended.*

To develop a conceptual model involves envisioning the proposed product, based on the users' needs and other requirements identified. To ensure that it is designed to be understandable in the manner intended requires doing iterative testing of the product as it is developed. A key aspect of this design process is initially to decide what the users will be doing when carrying out their tasks. For example, will they be primarily searching for information, creating documents, communicating with other users, recording events, or some other activity? At this stage, the interaction mode that would best support this needs to be considered. For example, would allowing the users to browse be appropriate, or would allowing them to ask questions directly to the system in their native language be more effective? Decisions about which kind of interaction style to use (e.g., whether to use a menu-based system, speech input, commands) should be made in relation to the interaction mode. Thus, decisions about which mode of interaction to support differ from those made about which style of interaction to have; the former being at a higher level of abstraction. The former are also concerned with determining the nature of the users' activities to support, while the latter are concerned with the selection of specific kinds of interface.

Once a set of possible ways of interacting with an interactive system has been identified, the design of the conceptual model then needs to be thought through in terms of actual concrete solutions. This entails working out the behavior of the interface, the particular interaction styles that will be used, and the "look and feel" of the interface. At this stage of "fleshing out," it is always a good idea to explore a number of possible designs and to assess the merits and problems of each one.

Another way of designing an appropriate conceptual model is to select an interface metaphor. This can provide a basic structure for the conceptual model that is couched in knowledge users are familiar with. Examples of well-known interface metaphors are the desktop and search engines (which we will cover in Section 2.4). Interaction paradigms can also be used to guide the formation of an appropriate conceptual metaphor. They provide particular ways of thinking about interaction design, such as designing for desktop applications or ubiquitous computing (these will also be covered in Section 2.5).

As with any aspect of interaction design, the process of fleshing out conceptual models should be done iteratively, using a number of methods. These include sketching out ideas, *storyboarding*, describing possible scenarios, and prototyping aspects of the proposed behavior of the system. All these methods will be covered in Chapter 8, which focuses on *doing* conceptual design. Here, we describe the different kinds of conceptual models, interface metaphors, and interaction paradigms to give you a good understanding of the various types prior to thinking about how to design them.
There are a number of different kinds of conceptual models. These can be broken down into two main categories: those based on activities and those based on objects.

### 2.3.1 Conceptual models based on activities

The most common types of activities that users are likely to be engaged in when interacting with systems are:

1. instructing
2. conversing
3. manipulating and navigating
4. exploring and browsing

A first thing to note is that the various kinds of activity are not mutually exclusive, as they can be carried out together. For example, it is possible for someone to give instructions while conversing or navigate an environment while browsing. However, each has different properties and suggests different ways of being developed at the interface. The first one is based on the idea of letting the user issue instructions to the system when performing tasks. This can be done in various interaction styles: typing in commands, selecting options from menus in a windows environment or on a touch screen, speaking aloud commands, pressing buttons, or using a combination of function keys. The second one is based on the user conversing with the system as though talking to someone else. Users speak to the system or type in questions to which the system replies via text or speech output. The third type is based on allowing users to manipulate and navigate their way through an environment of virtual objects. It assumes that the virtual environment shares some of the properties of the physical world, allowing users to use their knowledge of how physical objects behave when interacting with virtual objects. The fourth kind is based on the system providing information that is structured in such a way as to allow users to find out or learn things, without having to formulate specific questions to the system.

**Activity 2.2**

A company is building a wireless information system to help tourists find their way around an unfamiliar city. What would they need to find out in order to develop a conceptual model?

**Comment**

To begin, they would need to ask: what do tourists want? Typically, they want to find out lots of things, such as how to get from A to B, where the post office is and where a good Chinese restaurant is. They then need to consider how best to support the activity of requesting information. Is it preferable to enable the tourists to ask questions of the system as if they were having a conversation with another human being? Or would it be more appropriate to allow them to ask questions as if giving instructions to a machine? Alternatively, would they prefer a system that structures information in the form of lists, maps, and recommendations that they could then explore at their leisure?
1. Instructing

This kind of conceptual model describes how users carry out their tasks through instructing the system what to do. Examples include giving instructions to a system to perform operations like tell the time, print a file, and remind the user of an appointment. A diverse range of devices has been designed based on this model, including VCRs, hi-fi systems, alarm clocks, and computers. The way in which the user issues instructions can vary from pressing buttons to typing in strings of characters. Many activities are readily supported by giving instructions.

Operating systems like Unix and DOS have been specifically designed as command-based systems, to which the user issues instructions at the prompt as a command or set of commands. In Windows and other GUI-based systems, control keys or the selection of menu options via a mouse are used. Well-known applications that are command-based include word processing, email, and CAD. Typically, a wide range of functions is provided from which users choose when they want to do something to the object they are working on. For example, a user writing a report using a word processor will want to format the document, count the numbers of words typed, and check the spelling. The user will need to instruct the system to do these operations by issuing appropriate commands. Typically, commands are carried out in a sequence, with the system responding appropriately (or not) as instructed.

One of the main benefits of an instruction-based conceptual model is that it supports quick and efficient interaction. It is particularly suited to repetitive kinds of actions performed on multiple objects. Examples include the repetitive actions of saving, deleting, and organizing email messages or files.

**ACTIVITY 2.3**

There are many different kinds of vending machines in the world. Each offers a range of goods, requiring the user initially to part with some money. Figure 2.3 shows photos of two different vending machines, one that provides soft drinks and the other a range of snacks. Both support the interaction style of issuing instructions. However, the way they do it is quite different.

What instructions must be issued to obtain a can of soft drink from the first machine and a bar of chocolate from the second? Why has it been necessary to design a more complex mode of interaction for the second vending machine? What problems can arise with this mode of interaction?

**Comment**

The first vending machine has been designed on a very simple instruction-based conceptual model. There are a small number of drinks to choose from and each is represented by a large button displaying the label of each drink. The user simply has to press one button and (hopefully) this will have the effect of returning the selected drink. The second machine is more complex, offering a wider range of snacks. The trade-off for providing more choices, however, is that the user can no longer instruct the machine by using a simple one-press action but is required to use a more complex process, involving: (i) reading off the code (e.g., C12) under the item chosen, then (ii) keying this into the number pad adjacent to the displayed items, and (iii) checking the price of the selected option and ensuring that the amount of money inserted is the same or more (depending on whether or not the machine provides change). Problems that can arise from this mode of interaction are the customer
misreading the code and or mistyping in the code, resulting in the machine not issuing the snack or providing the wrong sort.

A better way of designing an interface for a large number of choices of variable cost is to continue to use direct mapping, but use buttons that show miniature versions of the snacks placed in a large matrix (rather than showing actual versions). This would use the available space at the front of the vending machine more economically. The customer would need only to press the button of the object chosen and put in the correct amount of money.

Much research has been carried out on how to optimize command-based and other instruction-giving systems with respect to usability goals. The form of the commands (e.g., the use of abbreviations, full names, icons, and/or labels), their syntax (how best to combine different commands), and their organization (e.g., how to structure options in different menus) are examples of some of the main areas that have been investigated (Shneiderman, 1998). In addition, various cognitive issues have been investigated that we will look at in the next chapter, such as the problems people have in remembering the names of a set of commands. Less
research has been carried out, however, on the best way to design the ordering and sequencing of button pressing for physical devices like cell phones, calculators, remote controls and vending machines.

Another ubiquitous vending machine is the ticket machine. Typically, a number of instructions have to be given in a sequence when using one of these. Consider ticket machines designed to issue train tickets at railway stations—how often have you (or the person in front of you) struggled to work out how to purchase a ticket and made a mistake? How many instructions have to be given? What order are they given in? Is it logical or arbitrary? Could the interaction have been designed any differently to make it more obvious to people how to issue instructions to the machine to get the desired train ticket?

Comment

Ticketing machines vary enormously from country to country and from application to application. There seems to be little attempt to standardize. Therefore, a person's knowledge of the Eurostar ticketing machine will not be very useful when buying a ticket for the Sydney Monorail or cinema tickets for the Odeon. Sometimes the interaction has been designed to get you to specify the type of ticket first (e.g., adult, child), the kind of ticket (e.g., single, return, special saver), then the destination, and finally to insert their money. Others require that the user insert a credit card first, before selecting the destination and the type of ticket.

2. Conversing

This conceptual model is based on the idea of a person conversing with a system, where the system acts as a dialog partner. In particular, the system is designed to respond in a way another human being might when having a conversation with someone else. It differs from the previous category of instructing in being intended to reflect a more two-way communication process, where the system acts more like a partner than a machine that simply obeys orders. This kind of conceptual model has been found to be most useful for applications in which the user needs to find out specific kinds of information or wants to discuss issues. Examples include advisory systems, help facilities, and search engines. The proposed tourist application described earlier would fit into this category.

The kinds of conversation that are supported range from simple voice-recognition menu-driven systems that are interacted with via phones to more complex natural-language-based systems that involve the system parsing and responding to user queries typed in by the user. Examples of the former include banking, ticket booking, and train time inquiries, where the user talks to the system in single-word phrases (e.g., yes, no, three) in response to prompts from the system. Examples of the latter include search engines and help systems, where the user types in a specific query (e.g., how do I change the margin widths?) to which the system responds by giving various answers.

A main benefit of a conceptual model based on holding a conversation is that it allows people, especially novices, to interact with a system in a way they are already familiar with. For example, the search engine "Ask Jeeves for Kids!!" allows children to ask a question in a way they would when asking their teachers or parents—rather than making them reformulate their question in terms of key words and Boolean logic. A disadvantage of this approach, however, is the misunderstandings that can arise when the search engine is unable to answer the child's question in the
You asked: How many legs does a centipede have?

Jeeves knows these answers:

Where can I find a definition for the math term leg?

Where can I find a concise encyclopedia article on centipedes?

Where can I see an image of the human appendix?

Why does my leg or other limb fall asleep?

Where can I find advice on controlling the garden pest millipedes and centipedes?

Where can I find resources from Britannica.com on leg?

Figure 2.4 The response from "Ask Jeeves for Kids!" search engine when asked "how many legs does a centipede have?"

way the child expects. For example, a child might type in a seemingly simple question, like "How many legs does a centipede have?" which the search engine finds difficult to answer. Instead, the search engine replies by suggesting a number of possible websites that may be relevant but—as can be seen in Figure 2.4—can be off the mark.

Another problem that can arise from a conversational-based, conceptual model is that certain kinds of tasks are transformed into cumbersome and one-sided interactions. This is especially the case for automated phone-based systems that use auditory menus to advance the conversation. Users have to listen to a voice providing several options, then make a selection, and repeat through further layers of menus before accomplishing their goal (e.g., reaching a real human, paying a bill). Here is the beginning of a dialog between a user who wants to find out about car insurance and an insurance company’s reception system:

<user dials an insurance company>
"Welcome to St. Paul's Insurance Company. Press 1 if new customer, 2 if you are an existing customer".
<user presses 1>
"Thank you for calling St. Paul's Insurance Company. If you require house insurance press 1, car insurance press 2, travel insurance press 3, health insurance press 4, other press 5"
<user presses 2>
"You have reached the car insurance division. If you require information about fully comprehensive insurance press 1, 3rd-party insurance press 2..."
A recent development based on the conversing conceptual model is animated agents. Various kinds of characters, ranging from "real" people appearing at the interface (e.g., videoed personal assistants and guides) to cartoon characters (e.g., virtual and imaginary creatures), have been designed to act as the partners in the conversation with the system. In so doing, the dialog partner has become highly visible and tangible, appearing to both act and talk like a human being (or creature). The user is able to see, hear, and even touch the partner (when it is a physical toy) they are talking with, whereas with other systems based on a dialog partner (e.g., help systems) they can only hear or read what the system is saying. Many agents have also been designed to exhibit desirable human-like qualities (e.g., humorous, happy, enthusiastic, pleasant, gentle) that are conveyed through facial expressions and lifelike physical movements (head and lip movements, body movements). Others have been designed more in line with Disney-like cartoon characters, exhibiting exaggerated behaviors (funny voices, larger-than-life facial expressions).

Animated agents that exhibit human-like or creature-like physical behavior as well as "talk" can be more believable. The underlying conceptual model is conveyed much more explicitly through having the system act and talk via a visible agent. An advantage is that it can make it easier for people to work out that the interface agent (or physical toy) they are conversing with is not a human being, but a synthetic character that has been given certain human qualities. In contrast, when the dialog partner is hidden from view, it is more difficult to discern what is behind it and just how intelligent it is. The lack of visible cues can lead users into thinking it is more intelligent than it actually is. If the dialog partner then fails to understand their questions or comments, users are likely to lose patience with it. Moreover,
they are likely to be less forgiving of it (having been fooled into thinking the dialog partner is more intelligent than it really is) than of a dialog partner that is represented as a cartoon character at the interface (having only assumed it was a simple partner). The flip side of imbuing dialog partners with a physical presence at the interface, however, is that they can turn out to be rather annoying (for more on this topic see Chapter 5).

3. Manipulating and navigating

This conceptual model describes the activity of manipulating objects and navigating through virtual spaces by exploiting users' knowledge of how they do this in the physical world. For example, virtual objects can be manipulated by moving, selecting, opening, closing, and zooming in and out of them. Extensions to these actions can also be included, such as manipulating objects or navigating through virtual spaces, in ways not possible in the real world. For example, some virtual worlds have been designed to allow users to teleport from place to place or to transform one object into another.

A well known instantiation of this kind of conceptual model is direct manipulation. According to Ben Shneiderman (1983), who coined the term, direct-manipulation interfaces possess three fundamental properties:

- continuous representation of the objects and actions of interest
- rapid reversible incremental actions with immediate feedback about the object of interest
- physical actions and button pressing instead of issuing commands with complex syntax

Benefits of direct manipulation interfaces include:

- helps beginners learn basic functionality rapidly
- experienced users can work rapidly on a wide range of tasks
- infrequent users can remember how to carry out operations over time
- no need for error messages, except very rarely
  users can immediately see if their actions are furthering their goals and if not do something else
- users experience less anxiety
- users gain confidence and mastery and feel in control

Apple Computer Inc. was one of the first computer companies to design an operating environment using direct manipulation as its central mode of interaction. The highly successful Macintosh desktop demonstrates the main principles of direct manipulation (see Figure 2.5). To capitalize on people's understanding of what happens to physical objects in the real world, they used a number of visual and auditory cues at the interface that were intended to emulate them. One of
their assumptions was that people expect their physical actions to have physical results, so when a drawing tool is used, a corresponding line should appear and when a file is placed in the trash can a corresponding sound or visual cue showing it has been successfully thrown away is used (Apple Computer Inc., 1987). A number of specific visual and auditory cues were used to provide such feedback, including various animations and sounds (e.g. shrinking and expanding icons accompanied with 'shhhlicc' and 'crouik' sounds to represent opening and closing of files). Much of this interaction design was geared towards providing clues to the user to know what to do, to feel comfortable, and to enjoy exploring the interface.

Many other kinds of direct manipulation interfaces have been developed, including video games, data visualization tools and CAD systems. Virtual environments and virtual reality have similarly employed a range of interaction mechanisms that enable users to interact with and navigate through a simulated 3D physical world. For example, users can move around and explore aspects of a 3D environment (e.g., the interior of a building) while also moving objects around in the virtual environment, (e.g., rearranging the furniture in a simulated living room). Figure 2.6 on Color Plate 3 shows screen shots of some of these.

While direct manipulation and virtual environments provide a very versatile mode of interaction, they do have a number of drawbacks. At a conceptual level, some people may take the underlying conceptual model too literally and expect certain things to happen at the interface in the way they would in the physical world. A well known example of this phenomenon is of new Mac users being terri-
fied of dragging the icon of their floppy disk to the trash can icon on the desktop to eject it from the computer for fear of deleting it in the same way files are when placed in the trash can. The conceptual confusion arises because the designers opted to use the same action (dropping) on the same object (trash can) for two completely different operations, deleting and ejecting. Another problem is that not all tasks can be described by objects and not all actions can be done directly. Some tasks are better achieved through issuing instructions and having textual descriptions rather than iconic representations. Imagine if email messages were represented as small icons in your mailbox with abbreviations of who they were from and when they were sent. Moreover, you could only move them around by dragging them with a mouse. Very quickly they would take up your desk space and you would find it impossible to keep track of them all.

4. Exploring and browsing

This conceptual model is based on the idea of allowing people to explore and browse information, exploiting their knowledge of how they do this with existing media (e.g., books, magazines, TV, radio, libraries, pamphlets, brochures). When people go to a tourist office, a bookstore, or a dentist's surgery, often they scan and flick through parts of the information displayed, hoping to find something interesting to read. CD-ROMs, web pages, portals and e-commerce sites are applications based on this kind of conceptual model. Much thought needs to go into structuring the information in ways that will support effective navigation, allowing people to search, browse, and find different kinds of information.

ACTIVITY 2.5

What conceptual models are the following applications based on?

(a) a 3D video game, say a car-racing game with a steering wheel and tactile, audio, and visual feedback
(b) the Windows environment
(c) a web browser

Comment

(a) A 3D video game is based on a direct manipulation/virtual environment conceptual model.
(b) The Windows environment is based on a hybrid form of conceptual model. It combines a manipulating mode of interaction where users interact with menus, scrollbars, documents, and icons, an instructing mode of interaction where users can issue commands through selecting menu options and combining various function keys, and a conversational model of interaction where agents (e.g., Clippy) are used to guide users in their actions.
(c) A web browser is also based on a hybrid form of conceptual model, allowing users to explore and browse information via hyperlinks and also to instruct the network what to search for and what results to present and save.
BOX 2.1 Which is Best—Agents, Direct Manipulations, or Commands?

An ongoing debate in interaction design concerns the pros and cons of using direct manipulation versus ‘interface’ agents. Nicholas Negroponte (MIT Media Lab), a strong advocate of the agents approach, claims that they can be much more versatile than direct manipulation interfaces, allowing users to do what they want to do through delegating the boring and time-consuming tasks to an agent. He describes the analogy of a well-trained English butler, who answers the phone, tends to a person’s needs, fends off callers, and tells ‘white lies’ if necessary on his master’s behalf. Similarly, a digital butler is designed to read a user’s email and flag the important ones, scout the web and newsgroups for interesting information, screen unwanted electronic intrusions; and so on. His vision is based on the assumption that people like to delegate work to others rather than directly manipulating computers themselves.

In opposition, Ben Shneiderman (University of Maryland) warns of the dangers of delegating tasks to agents, pointing out how difficult it is to train an agent to do all the things users want done in the way they want them done. If the agents do the tasks incorrectly or do not understand what the user wants, frustration and anger will ensue. Moreover, he argues that users do not want to be constantly monitored and told what to do by the computer. Consider the analogy of your car deciding you should be driving more slowly because it is raining. He suggests that direct manipulation has many more advantages, allowing users to enjoy mastery and being in control. He points out how people like to know what is going on, be involved in the action and have a sense of power over the computer—all of which direct manipulation interfaces support.

Another perspective on this debate is that in fact many tasks are best carried out at an abstract level, involving neither manipulation nor conversing with an agent. Issuing abstract commands based on a carefully designed set of syntax and semantics is often a very efficient and elegant way of performing operations. This is especially the case for repetitive operations, where often the same action needs to be performed on multiple objects. Examples include sorting out files, deleting accumulated email messages, opening and closing files, and installing applications comprising multiple files—which when done by direct manipulation or through delegation can be inefficient or ambiguous.

Consider how you would edit an essay using a word processor. Suppose you had referenced work by Ben Shneiderman but had spelled his name as Schneiderman, with an extra “c” throughout the essay. How could you correct this error using a direct manipulation interface? You would need to read through your essay and manually select the “c” in every “Schneiderman,” highlighting and then deleting it. This is tedious and it would be easy to miss one or two. By contrast, this operation would be relatively effortless and also likely to be more accurate by issuing commands. You would simply need to instruct the word processor to find every “Schneiderman” and replace it with “Shneiderman.” This could be done through either speaking the commands or typing them into a dialog box.

As a compromise, several interaction designers have recognized the need to support abstract classes of action in direct manipulation interfaces, while allowing for command-based and dialog modes of interaction in direct manipulation interfaces. However, as mentioned earlier, such redundancy can result in a more complex conceptual model that the user will have to spend more time learning.
ACTIVITY 2.6 Which conceptual model or combination of models do you think is most suited to supporting the following user activities?

(a) downloading music off the web
(b) programming

Comment
(a) The activity involves selecting, saving, cataloging and retrieving large files from an external source. Users need to be able to browse and listen to samples of the music and then instruct the machine to save and catalog the files in an order that they can readily access at subsequent times. A conceptual model based on instructing and navigating would seem appropriate.

(b) Programming involves various activities including checking, debugging, copying libraries, editing, testing, and annotating. An environment that supports this range of tasks needs to be flexible. A conceptual model that allows visualization and easy manipulation of code plus efficient instructing of the system on how to check, debug, copy, etc., is essential.

2.3.2 Conceptual models based on objects

The second category of conceptual models is based on an object or artifact, such as a tool, a book, or a vehicle. These tend to be more specific than conceptual models based on activities, focusing on the way a particular object is used in a particular context. They are often based on an analogy with something in the physical world. An example of a highly successful conceptual model based on an object is the spreadsheet (Winograd, 1996). The object this is based on is the ledger sheet.

The first spreadsheet was designed by Dan Bricklin, and called VisiCalc. It enabled people to carry out a range of tasks that previously could only be done very laboriously and with much difficulty using other software packages, a calculator, or by hand (see Figure 2.7). The main reasons why the spreadsheet has become so successful are first, that Bricklin understood what kind of tool would be useful to people in the financial world (like accountants) and second, he knew how to design it so that it could be used in the way that these people would find useful. Thus, at the outset, he understood (i) the kinds of activities involved in the financial side of business, and (ii) the problems people were having with existing tools when trying to achieve these activities.

A core financial activity is forecasting. This requires projecting financial results based on assumptions about a company, such as projected and actual sales, investments, infrastructure, and costs. The amount of profit or loss is calculated for different projections. For example, a company may want to determine how much loss it will incur before it will start making a profit, based on different amounts of investment, for different periods of time. Financial analysts need to see a spread of projections for different time periods. Doing this kind of multiple projecting by hand requires much effort and is subject to errors. Using a calculator can reduce the computational load of doing numerous sums, but it still requires the person to do much key pressing and writing down of partial results — again making the process vulnerable to errors.

To tackle these problems, Bricklin exploited the interactivity provided by microcomputers and developed an application that was capable of interactive financial
modeling. Key aspects of his conceptual model were: (i) to create a spreadsheet that was analogous to a ledger sheet in the way it looked, with columns and rows, which allowed people to capitalize on their familiarity with how to use this kind of representation, (ii) to make the spreadsheet interactive, by allowing the user to input and change data in any of the cells in the columns or rows, and (iii) to get the computer to perform a range of different calculations and recalculations in response to user input. For example, the last column can be programmed to display the sum of all the cells in the columns preceding it. With the computer doing all the calculations, together with an easy-to-learn-and-use interface, users were provided with an easy-to-understand tool. Moreover, it gave them a new way of effortlessly working out any

Figure 2.7 Reference card showing annotated screen dump for VisiCalc
number of forecasts—greatly extending what they could do before with existing tools.

Another popular accounting tool intended for the home market, based on a conceptual model of an object, is Quicken. This used paper checks and registers for its basic structure. Other examples of conceptual models based on objects include most operating environments (e.g., Windows and the Mac desktop) and web portals. All provide the user with a familiar frame of reference when starting the application.

**BOX 2.2 The Star Interface (Based On Miller and Johnson, 1996 and Smith et al., 1982)**

In 1981, Xerox introduced the 8010 “Star” system, which revolutionized the way interfaces were designed for personal computing. Although not commercially successful, many of the ideas behind its design were borrowed and adapted by other companies, such as the Apple Mac and Microsoft Windows, which then became highly successful.

Star was designed as an office system, targeted at workers not interested in computing per se. An important design goal, therefore, was to make the computer as “invisible” to the users as possible and to design applications that were suitable for them. The Star developers spent several person-years at the beginning of the project working out an appropriate conceptual model for such an office system. In the end they selected a conceptual model based on a physical office. They wanted the office workers to imagine the computer to be like an office environment, by acting on electronic counterparts of physical objects in the real world. Their assumption was that this would simplify and clarify the electronic world, making it seem more familiar, less alien and easier to learn, (see Figure 2.8).

![Figure 2.8 The Star interface.](image-url)
2.3.3 A case of mix and match?

As we have pointed out, which kind of conceptual model is optimal for a given application obviously depends on the nature of the activity to be supported. Some are clearly suited to supporting a given activity (e.g., using manipulation and navigation for a flight simulator) while for others, it is less clear what might be best (e.g., writing and planning activities may be suited to both manipulation and giving instructions). In such situations, it is often the case that some form of hybrid conceptual model that combines different interaction styles is appropriate. For example, the tourist application in Activity 2.2 may end up being optimally designed based on a combination of conversing and exploring models. The user could ask specific questions by typing them in or alternatively browse through information. Shopping on the Internet is also often supported by a range of interaction modes. Sometimes the user may be browsing and navigating, other times communicating with an agent, at yet other times parting with credit card details via an instruction-based form fill-in. Hence, which mode of interaction is "active" depends on the stage of the activity that is being carried out.

**BOX 2.3 Do Users Understand The Conceptual Model In The Way Intended?**

A fundamental part of developing a conceptual model is to determine whether the ideas generated about how the system should look and behave will be understood by the users in the manner intended. Norman (1988) has provided a framework to elucidate the relationship between the design of a conceptual model and a user's understanding of it (see Figure 2.9). Essentially, there are three interacting components: the designer, the user, and the system. Behind each of these are three interlinking conceptual models:

- the design model—the model the designer has of how the system should work
- the system image—how the system actually works
- the user's model—how the user understands how the system works

In an ideal world, all three should map onto each other. Users should be able to carry out their tasks in the way intended by the designer through interacting with the system image, which makes it obvious what to do. However, if the system image does not make the design model clear to the users, it is likely that they will end up with an incorrect understanding of the system, which in turn will make them use the system ineffectively and make errors.
The down side of mixing interaction modes is that the underlying conceptual model can end up being more complex and ambiguous, making it more difficult for the user to understand and learn. For example, some operating and word-processing systems now make it possible for the user to carry out the same activity in a number of different ways (e.g., to delete a file the user can issue a command like CtrlD, speak to the computer by saying "delete file," or drag an icon of the file to the recycle bin). Users will have to learn the different styles to decide which they prefer. Inevitably, the learning curve will be steeper, but in the long run the benefits are that it enables users to decide how they want to interact with the system.

### 2.4 Interface metaphors

Another way of describing conceptual models is in terms of interface metaphors. By this is meant a conceptual model that has been developed to be similar in some way to aspects of a physical entity (or entities) but that also has its own behaviors and properties. Such models can be based on an activity or an object or both. As well as being categorized as conceptual models based on objects, the desktop and the spreadsheet are also examples of interface metaphors. Another example of an interface metaphor is a "search engine." The tool has been designed to invite comparison with a physical object—a mechanical engine with several parts working—together with an everyday action—searching by looking through numerous files in many different places to extract relevant information. The functions supported by a search engine also include other features besides those belonging to an engine that searches, such as listing and prioritizing the results of a search. It also does these actions in quite different ways from how a mechanical engine works or how a human being might search a library for books on a given topic. The similarities alluded to by the use of the term "search engine," therefore, are at a very general conceptual level. They are meant to conjure up the essence of the process of finding relevant information, enabling the user to leverage off this "anchor" further understanding of other aspects of the functionality provided.

Interface metaphors are based on conceptual models that combine familiar knowledge with new concepts. As mentioned in Box 2.2, the Star was based on a conceptual model of the familiar knowledge of an office. Paper, folders, filing cabinets, and mailboxes were represented as icons on the screen and were designed to possess some of the properties of their physical counterparts. Dragging a document icon across the desktop screen was seen as equivalent to picking up a piece of paper in the physical world and moving it (but of course is a very different action). Similarly, dragging an electronic document onto an electronic folder was seen as being analogous to placing a physical document into a physical cabinet. In addition, new concepts that were incorporated as part of the desktop metaphor were operations that couldn't be performed in the physical world. For example, electronic files could be placed onto an icon of a printer on the desktop, resulting in the computer printing them out.
BOX 2.4 Why Are Metaphors and Analogies So Popular?

People frequently use analogies and metaphors (here we use the terms interchangeably) as a source of inspiration to understand and explain to others what they are doing or trying to do, in terms that are familiar to them. They are an integral part of human language (Lackoff and Johnson, 1980). They are most commonly used to explain something that is unfamiliar or hard to grasp by way of comparison with something that is familiar and easy to grasp. For example, they are commonly employed in education, where teachers use them to introduce something new to students by comparing the new material to something they already understand. An example is the comparison of human evolution with a game. We are all familiar with the properties of a game: there are rules, each player has a goal to win (or lose), there are heuristics to deal with situations where there are no rules, there is the propensity to cheat when the other players are not looking, and so on. By conjuring up these properties, the analogy helps us begin to understand the more difficult concept of evolution—how it happens, what rules govern it, who cheats, and so.

It is not surprising, therefore, to see how widely metaphors and analogies have been applied in interaction design. Both have been used, in overlapping ways, to conceptualize abstract, hard to imagine, and difficult to articulate computer-based concepts and interactions in more concrete and familiar terms and as graphical visualizations at the interface. This use includes:

- as a way of conceptualizing a particular interaction style, e.g., using the system as a tool
- as a conceptual model that is instantiated as part of an interface, e.g., the desktop metaphor
- as a way of describing computers, e.g., the Internet highway
- names for describing specific operations, e.g., “cut” and “paste” commands for deleting and copying objects (analogy taken from the media industry)
- as part of the training material aimed at helping learning, e.g., comparing a word processor with a typewriter.

In many instances, it is hard not to use metaphorical terms, as they have become so ingrained in the language we use to express ourselves. This is increasingly the case when talking about computers. Just ask yourself or someone else to describe how the Internet works. Then try doing it without using any metaphors or analogies.

ACTIVITY 2.7 Interface metaphors are often actually composites, i.e., they combine quite different pieces of familiar knowledge with the system functionality. We already mentioned the "search engine" as one such example. Can you think of any others?

Comment Some other examples include:

- Scrollbar—combines the concept of a scroll with a bar, as in bar chart
- Toolbar—combines the idea of a set of tools with a bar
- Portal website—a gateway to a particular collection of pages of networked information

Benefits of interface metaphors

Interface metaphors have proven to be highly successful, providing users with a familiar orienting device and helping them understand and learn how to use a system. People find it easier to learn and talk about what they are doing at the com-
puter interface in terms familiar to them—whether they are computer-phobic or highly experienced programmers. Metaphorically based commands used in Unix, like "lint" and "pipe," have very concrete meanings in everyday language that, when used in the context of the Unix operating system, metaphorically represent some aspect of the operations they refer to. Although their meaning may appear obscure, especially to the novice, they make sense when understood in the context of programming. For example, Unix allows the programmer to send the output of one program to another by using the pipe (|) symbol. Once explained, it is easy to imagine the output from one container going to another via a pipe.

Can you think of any bizarre computing metaphors that have become common parlance whose original source of reference is (or always was) obscure?

**Comment**

A couple of intriguing ones are:

Java—The programming language Java originally was called Oak, but that name had already been taken. It is not clear how the developers moved from Oak to Java. Java is a name commonly associated with coffee. Other Java-based metaphors that have been spawned include Java beans (a reusable software component) and the steaming coffee-cup icon that appears in the top left-hand corner of Java applets.

Bluetooth—Bluetooth is used in a computing context to describe the wireless technology that is able to unite technology, communication, and consumer electronics. The name is taken from King Harald Blue Tooth, who was a 10th century legendary Viking king responsible for uniting Scandinavia and thus getting people to talk to each other.

**Opposition to using interface metaphors**

A mistake sometimes made by designers is to try to design an interface metaphor to look and behave literally like the physical entity it is being compared with. This misses the point about the benefit of developing interface metaphors. As stressed earlier, they are meant to be used to map familiar to unfamiliar knowledge, enabling users to understand and learn about the new domain. Designing interface metaphors only as literal models of the thing being compared with has understandably led to heavy criticism. One of the most outspoken critics is Ted Nelson (1990) who considers metaphorical interfaces as "using old half-ideas as crutches" (p. 237). Other objections to the use of metaphors in interaction design include:

Breaks the rules. Several commentators have criticized the use of interface metaphors because of the cultural and logical contradictions involved in accommodating the metaphor when instantiated as a GUI. A pet hate is the recycle bin (formerly trash can) that sits on the desktop. Logically and culturally (i.e., in the real world), it should be placed under the desk. If this same rule were followed in the virtual desktop, users would not be able to see the bin because it would be occluded by the desktop surface. A counter-argument to this objection is that it does...
not matter whether rules are contravened. Once people understand why the bin is on the desktop, they readily accept that the real-world rule had to be broken. Moreover, the unexpected juxtaposition of the bin on the desktop can draw to the user's attention the additional functionality that it provides.

**Too constraining.** Another argument against interface metaphors is that they are too constraining, restricting the kinds of computational tasks that would be useful at the interface. An example is trying to open a file that is embedded in several hundreds of files in a directory. Having to scan through hundreds of icons on a desktop or scroll through a list of files seems a very inefficient way of doing this. As discussed earlier, a better way is to allow the user to instruct the computer to open the desired file by typing in its name (assuming they can remember the name of the file).

**Conflicts with design principles.** By trying to design the interface metaphor to fit in with the constraints of the physical world, designers are forced into making bad design solutions that conflict with basic design principles. Ted Nelson sets up the trash can again as an example of such violation: "a hideous failure of consistency is the garbage can on the Macintosh, which means either "destroy this" or "eject it for safekeeping" (Nelson, 1990).

**Not being able to understand the system functionality beyond the metaphor.** It has been argued that users may get fixed in their understanding of the system based on the interface metaphor. In so doing, they may find it difficult to see what else can be done with the system beyond the actions suggested by the interface metaphor. Nelson (1990) also argues that the similarity of interface metaphors to any real objects in the world is so tenuous that it gets in the way more than it helps. We would argue the opposite: because the link is tenuous and there are only a certain number of similarities, it enables the user to see both the dissimilarities and how the metaphor has been extended.

**Overly literal translation of existing bad designs.** Sometimes designers fall into the trap of trying to create a virtual object to resemble a familiar physical object that is itself badly designed. A well-known example is the virtual calculator, which is designed to look and behave like a physical calculator. The interface of many physical calculators, however, has been poorly designed in the first place, based on poor conceptual models, with excessive use of modes, poor labeling of functions, and difficult-to-manipulate key sequences (Mullet and Sano, 1995). The design of the calculator in Figure 2.10(a) has even gone as far as replicating functions needing shift keys (e.g., deg, oct, and hex), which could have been redesigned as dedicated software buttons. Trying to use a virtual calculator that has been designed to emulate a poorly designed physical calculator is much harder than using the physical device itself. A better approach would have been for the designers to think about how to use the computational power of the computer to support the kinds of tasks people need to do when doing calculations (cf. the spreadsheet design). The calculator in Figure 2.10(b) has tried to do this to some extent, by moving the buttons closer to each other (minimizing the amount of mousing) and providing flexible display modes with one-to-one mappings with different functions.
Figure 2.10 Two virtual calculators where (a) has been designed too literally and (b) more appropriately for a computer screen.

**Limits the designer's imagination in conjuring up new paradigms and models.** Designers may fixate on "tired" ideas, based on well-known technologies, that they know people are very familiar with. Examples include travel and books for representing interaction with the web and hypermedia. One of the dangers of always looking backwards is that it restricts the designer in thinking of what new functionality to provide. For example, Gentner and Nielsen (1996) discuss how they used a book metaphor for designing the user interface to Sun Microsystems' online documentation. In hindsight they realized how it had blinkered them in organizing the online material, preventing them from introducing desirable functions such as the ability to reorder chapters according to their relevance scores after being searched.

Clearly, there are pitfalls in using interface metaphors in interaction design. Indeed, this approach has led to some badly designed conceptual models, that have resulted in confusion and frustration. However, this does not have to be the case. Provided designers are aware of the dangers and try to develop interface metaphors that effectively combine familiar knowledge with new functionality in a meaningful way, then many of the above problems can be avoided. Moreover, as we have seen with the spreadsheet example, the use of analogy as a basis for a conceptual model can be very innovative and successful, opening up the realm of computers and their applications to a greater diversity of people.
Examine a web browser interface and describe the various forms of analogy and composite interface metaphors that have been used in its design. What familiar knowledge has been combined with new functionality?

**Comment**

Many aspects of a web browser have been combined to create a composite interface metaphor:

- a range of toolbars, such as a button bar, navigation bar, favorite bar, history bar
- tabs, menus, organizers
- search engines, guides
- bookmarks, favorites
- icons for familiar objects like stop lights, home

These have been combined with other operations and functions, including saving, searching, downloading, listing, and navigating.

### 2.5 Interaction paradigms

At a more general level, another source of inspiration for informing the design of a conceptual model is an interaction paradigm. By this it is meant a particular philosophy or way of thinking about interaction design. It is intended to orient designers to the kinds of questions they need to ask. For many years the prevailing paradigm in interaction design was to develop applications for the desktop—intended to be used by single users sitting in front of a CPU, monitor, keyboard and mouse. A dominant part of this approach was to design software applications that would run using a GUI or WIMP interface (windows, icons, mouse and pull-down menus, alternatively referred to as windows, icons, menus and pointers).

As mentioned earlier, a recent trend has been to promote paradigms that move "beyond the desktop." With the advent of wireless, mobile, and handheld technologies, developers started designing applications that could be used in a diversity of ways besides running only on an individual's desktop machine. For example, in September, 2000, the clothes company Levis, with the Dutch electronics company Philips, started selling the first commercial e-jacket—incorporating wires into the lining of the jacket to create a body-area network (BAN) for hooking up various devices, e.g., mobile phone, MP3, microphone, and headphone (see Figure 1.2(iii) in Color Plate 1). If the phone rings, the MP3 player cuts out the music automatically to let the wearer listen to the call. Another innovation was handheld interactive devices, like the PalmPilot, for which a range of applications were programmed. One was to program the PalmPilot as a multipurpose identity key, allowing guests to check in to certain hotels and enter their room without having to interact with the receptionist at the front desk.

A number of alternative interaction paradigms have been proposed by researchers intended to guide future interaction design and system development (see Figure 2.11). These include:

- ubiquitous computing (technology embedded in the environment)
- pervasive computing (seamless integration of technologies)
- wearable computing (or wearables)
2.5 Interaction paradigms

Figure 2.11 Examples of new interaction paradigms: (a) Some of the original devices developed as part of the ubiquitous computing paradigm. Tabs are small hand-sized wireless computers which know where they are and who they are with. Pads are paper-sized devices connected to the system via radio. They know where they are and who they are with. Liveboards are large wall sized devices. The "Dangling String" created by artist Natalie Jeremijenko was attached directly to the ethernet that ran overhead in the ceiling. It spun around depending on the level of digital traffic.

(b) Ishii and Ulmer, MIT Lab (1997) Tangible bits: from GUIs of desktop PCs to Tangible User Interfaces. The paradigm is concerned with establishing a new type of HCI called "Tangible User Interfaces" (TUIs). TUIs augment the real physical world by coupling digital information to everyday physical objects and environments.

(c) Affective Computing: The project, called “BlueEyes,” is creating devices with embedded technology that gather information about people. This face (with movable eyebrows, eyes and mouth) tracks your movements and facial expressions and responds accordingly.
tangible bits, augmented reality, and physical/virtual integration
attentive environments (computers attend to user's needs)
the Workaday World (social aspects of technology use)

Ubiquitous computing ("ubicomp"). The late Mark Weiser (1991), an influential visionary, proposed the interaction paradigm of ubiquitous computing (Figure 2.11). His vision was for computers to disappear into the environment so that we would be no longer aware of them and would use them without thinking about them. As part of this process, they should "invisibly" enhance the world that already exists rather than create artificial ones. Existing computing technology, e.g., multimedia-based systems and virtual reality, currently do not allow us to do this. Instead, we are forced to focus our attention on the multimedia representations on the screen (e.g., buttons, menus, scrollbars) or to move around in a virtual simulated world, manipulating virtual objects.

So, how can technologies be designed to disappear into the background? Weiser did not mean ubiquity in the sense of simply making computers portable so that they can be moved from the desk into our pockets or used on trains or in bed. He meant that technology be designed to be integrated seamlessly into the physical world in ways that extend human capabilities. One of his prototypes was a "tabs, pads, and boards" setup whereby hundreds of computer devices equivalent in size to post-it notes, sheets of paper, and blackboards would be embedded in offices. Like the spreadsheet, such devices are assumed to be easy to use, because they capitalize on existing knowledge about how to interact and use everyday objects. Also like the spreadsheet, they provide much greater computational power. One of Weiser's ideas was that the tabs be connected to one another, enabling them to become multipurpose, including acting as a calendar, diary, identification card, and an interactive device to be used with a PC.

Ubiquitous computing will produce nothing fundamentally new, but by making everything faster and easier to do, with less strain and fewer mental gymnastics, it will transform what is apparently possible (Weiser, 1991, p. 940).

Pervasive computing. Pervasive computing is a direct follow-on of ideas arising from ubiquitous computing. The idea is that people should be able to access and interact with information any place and any time, using a seamless integration of technologies. Such technologies are often referred to as smart devices or information appliances—designed to perform a particular activity. Commercial products include cell phones and handheld devices, like PalmPilots. On the domestic front, other examples currently being prototyped include intelligent fridges that signal the user when stocks are low, interactive microwave ovens that allow users to access information from the web while cooking, and smart pans that beep when the food is cooked.

Wearable computing. Many of the ideas behind ubiquitous computing have since inspired other researchers to develop technologies that are part of the environment. The MIT Media Lab has created several such innovations. One example is wearable computing (Mann, 1996). The combination of multimedia and wireless
communication presented many opportunities for thinking about how to embed such technologies on people in the clothes they wear. Jewelry, head-mounted caps, glasses, shoes, and jackets have all been experimented with to provide the user with a means of interacting with digital information while on the move in the physical world. Applications that have been developed include automatic diaries that keep users up to date on what is happening and what they need to do throughout the day, and tour guides that inform users of relevant information as they walk through an exhibition and other public places (Rhodes et al., 1999).

**Tangible bits, augmented reality, and physical/virtual integration.** Another development that has evolved from ubiquitous computing is tangible user interfaces or tangible bits (Ishii and Ullmer, 1997). The focus of this paradigm is the “integration of computational augmentations into the physical environment”, in other words, finding ways to combine digital information with physical objects and surfaces (e.g., buildings) to allow people to carry out their everyday activities. Examples include physical books embedded with digital information, greeting cards that play a digital animation when opened, and physical bricks attached to virtual objects that when grasped have a similar effect on the virtual objects. Another illustration of this approach is the one described in Chapter 1 of an enjoyable interface, in which a person could use a physical hammer to hit a physical key with corresponding virtual representations of the action being displayed on a screen.

Another part of this paradigm is augmented reality, where virtual representations are superimposed on physical devices and objects (as shown in Figure 2.1 on Color Plate 2). Bridging the gulf between physical and virtual worlds is also currently undergoing much research. One of the earlier precursors of this work was the Digital Desk (Wellner, 1993). Physical office tools, like books, documents and paper, were integrated with virtual representations, using projectors and video cameras. Both virtual and real documents were seamlessly combined.

**Attentive environments and transparent computing.** This interaction paradigm proposes that the computer attend to user's needs through anticipating what the user wants to do. Instead of users being in control, deciding what they want to do and where to go, the burden should be shifted onto the computer. In this sense the mode of interaction is much more implicit: computer interfaces respond to the user's expressions and gestures. Sensor-rich environments are used to detect the user's current state and needs. For example, cameras can detect where people are looking on a screen and decide what to display accordingly. The system should be able to determine when someone wants to make a call and which websites they want to visit at particular times. IBM's BlueEyes project is developing a range of computational devices that use non-obtrusive sensing technology, including videos and microphones, to track and identify users' actions. This information is then analyzed with respect to where users are looking, what they are doing, their gestures, and their facial expressions. In turn, this is coded in terms of the users' physical, emotional or informational state and is then used to determine what information they would like. For example, a BlueEyes-enabled computer could become active when a user first walks into a room, firing up any new email messages that have arrived. If the user shakes his or her head, it would be interpreted by the computer as "I don't want to read them," and instead show a listing of their appointments for that day.
The Workaday World. In the new paradigms mentioned above, the emphasis is on exploring how technological devices can be linked with each other and digital information in novel ways that allow people to do things they could not do before. In contrast, the Workaday World paradigm is driven primarily by conceptual and mundane concerns. It was proposed by Tom Moran and Bob Anderson (1990), when working at Xerox PARC. They were particularly concerned with the need to understand the social aspects of technology use in a way that could be useful for designers. The Workaday World paradigm focuses on the essential character of the workplace in terms of people's everyday activities, relationships, knowledge, and resources. It seeks to unravel the "set of patterns that convey the richness of the settings in which technologies live—the complex, unpredictable, multiform relationships that hold among the various aspects of working life" (p. 384).

2.6 From conceptual models to physical design

As we emphasize throughout this book, interaction design is an iterative process. It involves cycling through various design processes at different levels of detail. Primarily it involves: thinking through a design problem, understanding the user's needs, coming up with possible conceptual models, prototyping them, evaluating them with respect to usability and user experience goals, thinking about the design implications of the evaluation studies, making changes to the prototypes with respect to these, evaluating the changed prototypes, thinking about whether the changes have improved the interface and interaction, and so on. Interaction design may also require going back to the original data to gather and check the requirements. Throughout the iterations, it is important to think through and understand whether the conceptual model being developed is working in the way intended and to ensure that it is supporting the user's tasks.

Throughout this book we describe the way you should go about doing interaction design. Each iteration should involve progressing through the design in more depth. A first pass through an iteration should involve essentially thinking about the problem space and identifying some initial user requirements. A second pass should involve more extensive information gathering about users' needs and the problems they experience with the way they currently carry out their activities (see Chapter 7). A third pass should continue explicating the requirements, leading to thinking through possible conceptual models that would be appropriate (see Chapter 8). A fourth pass should begin "fleshing out" some of these using a variety of user-centered methods. A number of user-centered methods can be used to create prototypes of the potential candidates. These include using storyboarding to show how the interaction between the users and the system will take place and the laying out of cards and post-it notes to show the possible structure of and navigation through a website. Throughout the process, the various prototypes of the conceptual models should be evaluated to see if they meet users' needs. Informally asking users what they think is always a good starting point (see Chapter 12). A number of other techniques can also be used at different stages of the development of the prototypes, depending on the particular information required (see Chapters 13 and 14).
Many issues will need to be addressed when developing and testing initial prototypes of conceptual models. These include:

- the way information is to be presented and interacted with at the interface
- what combinations of media to use (e.g., whether to use sound and animations)
- the kind of feedback that will be provided
- what combinations of input and output devices to use (e.g., whether to use speech, keyboard plus mouse, handwriting recognition)
- whether to provide agents and in what format
- whether to design operations to be hardwired and activated through physical buttons or to represent them on the screen as part of the software
- what kinds of help to provide and in what format

While working through these design decisions about the nature of the interaction to be supported, issues concerning the actual physical design will need to be addressed. These will often fall out of the conceptual decisions about the way information is to be represented, the kind of media to be used, and so on. For example, these would typically include:

- **information presentation**
  - which dialogs and interaction styles to use (e.g., form fill-ins, speech input, menus)
  - how to structure items in graphical objects, like windows, dialog boxes and menus (e.g., how many items, where to place them in relation to each other)

- **feedback**
  - what navigation mechanisms to provide (e.g., forward and backward buttons)

- **media combination**
  - which kinds of icons to use

Many of these physical design decisions will be specific to the interactive product being built. For example, designing a calendar application intended to be used by business people to run on a handheld computer will have quite different constraints and concerns from designing a tool for scheduling trains to run over a large network, intended to be used by a team of operators via multiple large displays. The way the information will be structured, the kinds of graphical representations that will be appropriate, and the layout of the graphics on the screens will be quite different.

These kinds of design decisions are very practical, needing user testing to ensure that they meet with the usability goals. It is likely that numerous trade-offs will surface, so it is important to recognize that there is no right or wrong way to resolve these. Each decision has to be weighed with respect to the others. For example, if you decide that a good way of providing visibility for the calendar application on the handheld device is to have a set of "soft" navigation buttons permanently as
**DILEMMA** Realism versus Abstraction?

One of the challenges facing interaction designers is whether to use realism or abstraction when designing an interface to instantiate their conceptual model. By this it is meant designing objects either (i) to give the illusion of behaving and looking like real-world counterparts or (ii) to appear as simply abstractions of the objects being represented. This concern is particularly relevant when fleshing out conceptual models that are deliberately based on an analogy to some aspect of the real world. For example, is it preferable to design a desktop to look like a real desktop, a virtual house to look like a real house, or a virtual living room to look like a real living room? Or, alternatively, is it more effective to design representations of the conceptual model as simple abstract renditions, depicting only a few salient features?

We already discussed in Chapter 1 the problems of trying to design graphical interfaces with affordances. Here, we consider more generally the dilemma of using realism at the interface. One of the main benefits of using realism is that it can enable people, especially computer phobics and novices, to feel more comfortable when first learning an application. The reason for this is that such representations can readily tap into people's understanding of the physical world. Hence, realistic interfaces can help users initially understand the underlying conceptual model. In contrast, overly schematic and abstract representations can appear to be too computer-like and off-putting to the newcomer. The advantage of these kinds of more abstract interfaces is that they are often more efficient to use. Furthermore, the more experienced users become, the more they may find comfortable interfaces no longer to their liking. A dilemma facing designers, therefore, is deciding between designing interfaces to make novice users feel comfortable (but more experienced users less comfortable) versus designing interfaces to be effective for more experienced users (but maybe harder to learn by novices).

One of the earliest attempts at using realism at the interface was General Magic's office system Magic Cap, which was rendered in 3D. To achieve this degree of realism required using various perceptual cues such as perspective, shadowing, and shading. The result of their efforts was a rather cute interface (see Figure 2.12). Although their intentions were well-grounded, the outcome was less successful. Many people commented on how childishly and gawkily it looked, having the appearance of illustrations in a children's picture book rather than a work-based application.

Mullet and Sano (1995) also point out how a 3D rendition of an object like a desk nearly always suffers from both an unnatural point of view and an awkward rendering style that ironically destroy the impression of being in a real physical space. One reason for this is that 3D depictions conflict with the effective use of display space, especially when 2D editing tasks need to be performed. As can be seen in Figure 2.12, these kinds of tasks have been represented as "flat" buttons that appear to be floating in front of the desk (e.g., mail, program manager, task manager).

In certain kinds of applications, using realism can be very effective for both novices and experienced users. Video games fall into this category, especially those where users have to react rapidly to dynamic events that happen in a virtual world in real time, say flying a plane or playing a game of virtual football. Making the characters in the game resemble humans in the way they look, move, dress, and gesture also makes them seem more convincing and lifelike, enhancing the enjoyment and fun factor (see Figure 2.13).
Figure 2.12 Magic Cap’s 3D desktop interface.

Figure 2.13 3D avatars in computer games: A screenshot from The Sims World.
part of the visual display, you then need to consider the consequences of doing this for the rest of the information that needs to be interacted with. Will it still be possible to structure the display to show the calendar as days in a week or a month, all on one screen?

This part of the design process is highly dependent on the context and essentially involves lots of juggling between design decisions. If you visit our website you can try out some of the interactivities provided, where you have to make such decisions when designing the physical layout for various interfaces. Here, we provide the background and rationale that can help you make appropriate choices when faced with a series of design decisions (primarily Chapters 3-5 and 8). For example, we explain why you shouldn't cram a screen full of information; why certain techniques are better than others for helping users remember how to carry out their tasks at the interface; and why certain kinds of agents appear more believable than others.

Assignment

The aim of this assignment is for you to think about the appropriateness of different kinds of conceptual model that have been designed for similar kinds of physical and electronic artifacts.

(a) Describe the conceptual model that underlie the design of:

- a personal pocket-sized calendar/diary (one week to a page)
- a wall calendar (one month to a page, usually with a picture/photo)
- a wall planner (displaying the whole year)

What is the main kind of activity and object they are based on? How do they differ for each of the three artifacts? What metaphors have been used in the design of their physical interface (think about the way time is conceptualized for each of them)? Do users understand the conceptual models these are based on in the ways intended (ask a few people to explain how they use them)? Do they match the different user needs?

(b) Now describe the conceptual models that underlie the design of:

- an electronic personal calendar found on a personal organizer or handheld computer
- a shared calendar found on the web

How do they differ from the equivalent physical artifacts? What new functionality has been provided? What interface metaphors have been used? Are the functions and interface metaphor well integrated? What problems do users have with these interactive kinds of calendars? Why do you think this is?

Summary

This chapter has explained the importance of conceptualizing interaction design before trying to build anything. It has stressed throughout the need always to be clear and explicit about the rationale and assumptions behind any design decision made. It described a taxonomy of conceptual models and the different properties of each. It also discussed interface metaphors and interaction paradigms as other ways of informing the design of conceptual models.
Key points

- It is important to have a good understanding of the problem space, specifying what it is you are doing, why and how it will support users in the way intended.
- A fundamental aspect of interaction design is to develop a conceptual model.
- There are various kinds of conceptual models that are categorized according to the activity or object they are based on.
- Interaction modes (e.g., conversing, instructing) provide a structure for thinking about which conceptual model to develop.
- Interaction styles (e.g., menus, form fill-ins) are specific kinds of interfaces that should be decided upon after the conceptual model has been chosen.
- Decisions about conceptual design also should be made before commencing any physical design (e.g., designing an icon).
- Interface metaphors are commonly used as part of a conceptual model.
- Many interactive systems are based on a hybrid conceptual model. Such models can provide more flexibility, but this can make them harder to learn.
- 3D realism is not necessarily better than 2D or other forms of representation when instantiating a conceptual model: what is most effective depends on the users' activities when interacting with a system.
- General interaction paradigms, like WIMP and ubiquitous computing, provide a particular way of thinking about how to design a conceptual model.

Further reading

LAUREL, B. (1990) (ed.) The Art of Human Computer Design has a number of papers on conceptual models and interface metaphors. Two that are definitely worth reading are: Tom Erickson, "Working with interface metaphors" (pp. 65–74), which is a practical hands-on guide to designing interface metaphors (covered later in this book), and Ted Nelson's polemic, "The right way to think about software design" (pp. 229–234), which is a scathing attack on the use of interface metaphors.

JOHNSON, M. AND LAKOFF, G. (1980) Metaphors We Live By, The University of Chicago Press. Those wanting to find out more about how metaphors are used in everyday conversations should take a look at this text.

There are many good articles on the topic of interface agents. A classic is:

LANIER, J. (1995) Agents of alienation, ACM Interactions, 2(3), 66–72. The Art of Human Computer Design also provides several thought-provoking articles, including one called "Interface agents: metaphors with character" by Brenda Laurel (pp. 355–366) and another called "Guides: characterizing the interface" by Tim Oren et al. (pp. 367–382).


MIT's Media Lab (www.media.mit.edu) is a good starting place to find out what is currently happening in the world of agents, wearables, and other new interaction paradigms.
Interview with Terry Winograd

Terry Winograd is a professor of computer science at Stanford University. He has done extensive research and writing on the design of human-computer interaction. His early research on natural language understanding by computers was a milestone in artificial intelligence, and he has written two books and numerous articles on that topic. His book, Bringing Design to Software, brings together the perspectives of a number of leading researchers and designers. See Color Plate 2 for an example of his latest research.

YR: Tell me about your background and how you moved into interaction design.

TW: I got into interaction design through a couple of intermediate steps. I started out doing research into artificial intelligence. I became interested in how people interact with computers, in particular, when using ordinary language. It became clear after years of working on that, however, that the computer was a long way off from matching human abilities. Moreover, using natural language with a computer when it doesn't really understand you can be very frustrating and in fact a very bad way to interact with it. So, rather than trying to get the computer to imitate the person, I became interested in other ways of taking advantage of what the computer can do well and what the person can do well. That led me into the general field of HCI. As I began to look at what was going on in that field and to study it, it became clear that it was not the same as other areas of computer science. The key issues were about how the technology fits with what people could do and what they wanted to do. In contrast, most of computer science is really dominated by how the mechanisms operate.

I was very attracted to thinking more in the style of design disciplines, like product design, urban design, architecture, and so on. I realized that there was an approach that you might call a design way, that puts the technical aspects into the background with respect to understanding the interaction. Through looking at these design disciplines, I realized that there was something unique about interaction design, which is that it has a dialogic temporal element. By this I mean a human dialog not in the sense of using ordinary language, but in the sense of thinking about the sequence and the flow of interaction. So I think interaction design is about designing a space for people, where that space has to have a temporal flow. It has to have a dialog with the person.

YR: Could you tell me a bit more about what you think is involved in interaction design?

TW: One of the biggest influences is product design. I think that interaction design overlaps with it, because they both take a very strong user-oriented view. Both are concerned with finding a user group, understanding their needs, then using that understanding to come up with new ideas. They may be ones that the users don't even realize they need. It is then a matter of trying to translate who it is, what they are doing, and why they are doing it into possible innovations. In the case of product design it is products. In the case of interaction design it is the way that the computer system interacts with the person.

YR: What do you think are important inputs into the design process?

TW: One of the characteristics of design fields as opposed to traditional engineering fields is that there is much more dependence on case studies and examples than on formulas. Whereas an engineer knows how to calculate something, an architect or a designer is working in a tradition where there is a history over time of other things people have done. People have said that the secret of great design is to know what to steal and to know when some element or some way of doing things that worked before will be appropriate to your setting and then adapt it. Of course you can't apply it directly, so I think a big part of doing good design is experience and exposure. You have to have seen a lot of things in practice and understood what is good and bad about them, to then use these to inform your design.

YR: How do you see the relationship between studying interaction design and the practice of it? Is there a good dialog between research and practice?

TW: Academic study of interaction design is a tricky area because so much of it depends on a kind of tacit knowledge that comes through experience and
exposure. It is not the kind of thing you can set down easily as, say, you can scientific formulas. A lot of design tends to be methodological. It is not about the design per se but is more about how you go about doing design, in particular, knowing what are the appropriate steps to take and how you put them together.

**YR:** How do you see the field of interaction design taking on board the current explosion in new technologies—for example mobile, ubiquitous, infrared, and so on? Is it different, say, from 20 years ago when it was just about designing software applications to sit on the desktop?

**TW:** I think a real change in people's thinking has been to move from interface design to interaction design. This has been pushed by the fact that we do have all kinds of devices nowadays. Interface design used to mean graphical interfaces, which meant designing menus and other widgets. But now when you're talking about handheld devices, gesture interfaces, telephone interfaces and so on, it is clear that you can't focus just on the widgets. The widgets may be part of any one of these devices but the design thinking as a whole has to focus on the interaction.

**YR:** What advice would you give to a student coming into the field on what they should be learning and looking for?

**TW:** I think a student who wants to learn this field should think of it as a kind of dual process, that is what Donald Schon calls "reflection in action," needing both the action and the reflection. It is important to have experience with trying to build things. That experience can be from outside work, projects, and courses where you are actually engaged in making something work. At the same time you need to be able to step back and look at it not as "What do I need to do next?" but from the perspective of what you are doing and how that fits into the larger picture.

**YR:** Are there any classic case studies that stand out as good exemplars of interaction design?

**TW:** You need to understand what has been important in the past. I still use the Xerox Star as an exemplar because so much of what we use today was there. When you go back to look at the Star you see it in the context of when it was first created. I also think some exemplars that are very interesting are ones that never actually succeeded commercially. For example, I use the PenPoint system that was developed for pen computers by Go. Again, they were thinking fresh. They set out to do something different and they were much more conscious of the design issues than somebody who was simply adapting the next version of something that already existed. PalmPilot is another good example, because they looked at the problem in a different way to make something work. Another interesting exemplar, which other people may not agree with, is Microsoft Bob—not because it was a successful program, because it wasn't, but because it was a first exploration of a certain style of interaction, using animated agents. You can see very clearly from these exemplars what design trade-offs the designers were making and why and then you can look at the consequences.

**YR:** Finally, what are the biggest challenges facing people working in this area?

**TW:** I think one of the biggest challenges is what Pelle Ehn calls the dialectic between tradition and transcendence. That is, people work and live in certain ways already, and they understand how to adapt that within a small range, but they don't have an understanding or a feel for what it would mean to make a radical change, for example, to change their way of doing business on the Internet before it was around, or to change their way of writing from pen and paper when word processors weren't around. I think what the designer is trying to do is envision things for users that the users can't yet envision. The hard part is not fixing little problems, but designing things that are both innovative and that work.
Chapter 3

Understanding users

3.1 Introduction

Imagine trying to drive a car by using just a computer keyboard. The four arrow keys are used for steering, the space bar for braking, and the return key for accelerating. To indicate left you need to press the F1 key and to indicate right the F2 key. To sound your horn you need to press the F3 key. To switch the headlights on you need to use the F4 key and, to switch the windscreen wipers on, the F5 key. Now imagine as you are driving along a road a ball is suddenly kicked in front of you. What would you do? Bash the arrow keys and the space bar madly while pressing the F4 key? How would you rate your chances of missing the ball?

Most of us would balk at the very idea of driving a car this way. Many early video games, however, were designed along these lines: the user had to press an arbitrary combination of function keys to drive or navigate through the game. There was little, if any, consideration of the user's capabilities. While some users regarded mastering an arbitrary set of keyboard controls as a challenge, many users found them very limiting, frustrating, and difficult to use. More recently, computer consoles have been designed with the user's capabilities and the demands of the activity in mind. Much better ways of controlling and interacting, such as through using joysticks and steering wheels, are provided that map much better onto the physical and cognitive aspects of driving and navigating.

In this chapter we examine some of the core cognitive aspects of interaction design. Specifically, we consider what humans are good and bad at and show how this knowledge can be used to inform the design of technologies that both extend human capabilities and compensate for their weaknesses. We also look at some of the influential cognitively based conceptual frameworks that have been developed for explaining the way humans interact with computers. (Other ways of conceptualizing
human behavior that focus on the social and affective aspects of interaction design are presented in the following two chapters.)

The main aims of this chapter are to:

- Explain what cognition is and why it is important for interaction design.
- Describe the main ways cognition has been applied to interaction design.
- Provide a number of examples in which cognitive research has led to the design of more effective interactive products.
- Explain what mental models are.
- Give examples of conceptual frameworks that are useful for interaction design.
- Enable you to try to elicit a mental model and be able to understand what it means.

### 3.2 What is cognition?

Cognition is what goes on in our heads when we carry out our everyday activities. It involves cognitive processes, like thinking, remembering, learning, daydreaming, decision making, seeing, reading, writing and talking. As Figure 3.1 indicates, there are many different kinds of cognition. Norman (1993) distinguishes between two general modes: experiential and reflective cognition. The former is a state of mind in which we perceive, act, and react to events around us effectively and effortlessly. It requires reaching a certain level of expertise and engagement. Examples include driving a car, reading a book, having a conversation, and playing a video game. In contrast, reflective cognition involves thinking, comparing, and decision-making. This kind of cognition is what leads to new ideas and creativity. Examples include designing, learning, and writing a book. Norman points out that both modes are essential for everyday life but that each requires different kinds of technological support.

![Figure 3.1 What goes on in the mind?](image-url)
Cognition has also been described in terms of specific kinds of processes. These include:

- attention
- perception and recognition
- memory
- learning
- reading, speaking, and listening
- problem solving, planning, reasoning, decision making

It is important to note that many of these cognitive processes are interdependent: several may be involved for a given activity. For example, when you try to learn material for an exam, you need to attend to the material, perceive, and recognize it, read it, think about it, and try to remember it. Thus, cognition typically involves a range of processes. It is rare for one to occur in isolation. Below we describe the various kinds in more detail, followed by a summary box highlighting core design implications for each. Most relevant (and most thoroughly researched) for interaction design is memory, which we describe in greatest detail.

Attention is the process of selecting things to concentrate on, at a point in time, from the range of possibilities available. Attention involves our auditory and/or visual senses. An example of auditory attention is waiting in the dentist’s waiting room for our name to be called out to know when it is our time to go in. An example of attention involving the visual senses is scanning the football results in a newspaper to attend to information about how our team has done. Attention allows us to focus on information that is relevant to what we are doing. The extent to which this process is easy or difficult depends on (i) whether we have clear goals and (ii) whether the information we need is salient in the environment:

(i) **Our goals** If we know exactly what we want to find out, we try to match this with the information that is available. For example, if we have just landed at an airport after a long flight and want to find out who had won the World Cup, we might scan the headlines at the newspaper stand, check the web, call a friend, or ask someone in the street.

When we are not sure exactly what we are looking for we may browse through information, allowing it to guide our attention to interesting or salient items. For example, when we go to a restaurant we may have the general goal of eating a meal but only a vague idea of what we want to eat. We peruse the menu to find things that whet our appetite, letting our attention be drawn to the imaginative descriptions of various dishes. After scanning through the possibilities and imagining what each dish might be like (plus taking into account other factors, such as cost, who we are with, what the specials are, what the waiter recommends, whether we want a two- or three-course meal, and so on), we may then make a decision.

(ii) **Information presentation** The way information is displayed can also greatly influence how easy or difficult it is to attend to appropriate pieces of information. Look at Figure 3.2 and try the activity. Here, the information-searching tasks are very precise, requiring specific answers. The information density is identical in both

3.2 What is cognition? 75
Understand users

Figure 3.2 Two different ways of structuring the same information at the interface: one makes it much easier to find information than the other. Look at the top screen and: (i) find the price for a double room at the Quality Inn in Columbia; (ii) find the phone number of the Days Inn in Charleston. Then look at the bottom screen and (i) find the price of a double room at the Holiday Inn in Bedford; (ii) find the phone number of the Quality Inn in Bedford. Which took longer to do? In an early study Tullis found that the two screens produced quite different results: it took an average of 3.2 seconds to search the top screen and 5.5 seconds to find the same kind of information in the bottom screen. Why is this so, considering that both displays have the same density of information (31%)? The primary reason is the way the characters are grouped in the display: in the top they are grouped into vertical categories of information (e.g., place, kind of accommodation, phone number, and rates) that have columns of space between them, making it much easier to select the necessary information.

Perception refers to how information is acquired from the environment, via the different sense organs (e.g., eyes, ears, fingers) and transformed into experiences of objects, events, sounds, and tastes (Roth, 1986). It is a complex process, involving other cognitive processes such as memory, attention, and language. Vision is the

displays. However, it is much harder to find the information in the bottom screen than in the top screen. The reason for this is that the information is very poorly structured in the bottom, making it difficult to find the information. In the top the information has been ordered into meaningful categories with blank spacing between them, making it easier to select the necessary information.
most dominant sense for sighted individuals, followed by hearing and touch. With respect to interaction design, it is important to present information in a way that can be readily perceived in the manner intended. For example, there are many ways to design icons. The key is to make them easily distinguishable from one another and to make it simple to recognize what they are intended to represent (not like the ones in Figure 3.4).

Combinations of different media need also to be designed to allow users to recognize the composite information represented in them in the way intended. The use of sound and animation together needs to be coordinated so they happen in a logical sequence. An example of this is the design of lip-synch applications, where the animation of an avatar’s or agent’s face to make it appear to be talking, must be carefully synchronized with the speech that is emitted. A slight delay between the two can make it difficult and disturbing to perceive what is happening—as sometimes happens when film dubbing gets out of synch. A general design principle is
that information needs to be represented in an appropriate form to facilitate the perception and recognition of its underlying meaning.

Memory involves recalling various kinds of knowledge that allow us to act appropriately. It is very versatile, enabling us to do many things. For example, it allows us to recognize someone's face, remember someone's name, recall when we last met them and know what we said to them last. Simply, without memory we would not be able to function.

It is not possible for us to remember everything that we see, hear, taste, smell, or touch, nor would we want to, as our brains would get completely overloaded. A filtering process is used to decide what information gets further processed and memorized. This filtering process, however, is not without its problems. Often we

**DESIGN IMPLICATIONS | Perception**

Representations of information need to be designed to be perceptible and recognizable across different media:

- Icons and other graphical representations should enable users to readily distinguish their meaning.
- Sounds should be audible and distinguishable so users understand what they represent.
- Speech output should enable users to distinguish between the set of spoken words and also be able to understand their meaning.
- Text should be legible and distinguishable from the background (e.g., it is OK to use yellow text on a black or blue background but not on a white or green background).
- Tactile feedback used in virtual environments should allow users to recognize the meaning of the various touch sensations being emulated. The feedback should be distinguishable so that, for example, the sensation of squeezing is represented in a tactile form that is different from the sensation of pushing.
forget things we would dearly love to remember and conversely remember things we would love to forget. For example, we may find it difficult to remember everyday things like people's names and phone numbers or academic knowledge like mathematical formulae. On the other hand, we may effortlessly remember trivia or tunes that cycle endlessly through our heads.

How does this filtering process work? Initially, encoding takes place, determining which information is attended to in the environment and how it is interpreted. The extent to which it takes place affects our ability to recall that information later. The more attention that is paid to something and the more it is processed in terms of thinking about it and comparing it with other knowledge, the more likely it is to be remembered. For example, when learning about a topic it is much better to reflect upon it, carry out exercises, have discussions with others about it, and write notes than just passively read a book or watch a video about it. Thus, how information is interpreted when it is encountered greatly affects how it is represented in memory and how it is used later.

Another factor that affects the extent to which information can be subsequently retrieved is the context in which it is encoded. One outcome is that sometimes it can be difficult for people to recall information that was encoded in a different context from the one they currently are in. Consider the following scenario:

You are on a train and someone comes up to you and says hello. You don't recognize him for a few moments but then realize it is one of your neighbors. You are only used to seeing your neighbor in the hallway of your apartment block and seeing him out of context makes him difficult to recognize initially.

Another well-known memory phenomenon is that people are much better at recognizing things than recalling things. Furthermore, certain kinds of information are easier to recognize than others. In particular, people are very good at recognizing thousands of pictures, even if they have only seen them briefly before.

Try to remember the dates of all the members of your family's and your closest friends' birthdays. How many can you remember? Then try to describe what is on the cover of the last DVD/CD or record you bought. Which is easiest and why?

It is likely that you remembered much better what was on the CD/DVD/record cover (the image, the colors, the title) than the birthdays of your family and friends. People are very good at remembering visual cues about things, for example the color of items, the location of objects (a book being on the top shelf), and marks on an object (e.g., a scratch on a watch, a chip on a cup). In contrast, people find other kinds of information persistently difficult to learn and remember, especially arbitrary material like birthdays and phone numbers.

Instead of requiring users to recall from memory a command name from a possible set of hundreds or even thousands, GUIs provide visually based options that
users can browse through until they recognize the operation they want to perform (see Figure 3.5(a) and (b)). Likewise, web browsers provide a facility of bookmarking or saving favorite URLs that have been visited, providing a visual list. This means that users need only recognize a name of a site when scanning through the saved list of URLs.

Figure 3.5(a) A DOS-based interface, requiring the user to type in commands.
3.2 What is cognition?

File Folder File Folder File Folder

Figure 3.5(b) A Windows-based interface, with menus, icons, and buttons.

Activity 3.2 What strategies do you use to help you remember things?

Comment
People often write down what they need to remember on a piece of paper. They also ask others to remind them. Another approach is to use various mental strategies, like mnemonics. A mnemonic involves taking the first letters of a set of words in a phrase or set of concepts and using them to make a more memorable phrase, often using bizarre and idiosyncratic connections. For example, some people have problems working out where east is in relation to west and vice versa (i.e., is it to the left or right). A mnemonic to help figure this out is to take the first letters of the four main points of the compass and then use them in the phrase "Never Eat Shredded Wheat" mentally recited in a clockwise sequence.

A growing problem for computer users is file management. The number of documents created, images and videoclips downloaded, emails and attachments saved, URLs bookmarked, and so on increases every day. A major problem is finding them again. Naming is the most common means of encoding them, but trying to remember a name of a file you created some time back can be very difficult, especially if there are tens of thousands of named files. How might such a process be facilitated, bearing in mind people's memory abilities? Mark Lansdale, a British psychologist, has been researching this problem of information retrieval for many
BOX 3.1 The Problem with the Magical Number 7 Plus or Minus 2

Perhaps the best known finding in psychology (certainly the one that nearly all students remember many years after they have finished their studies) is George Miller's (1956) theory that 7±2 chunks of information can be held in short-term memory at any one time. By short-term memory he meant a memory store in which information was assumed to be processed when first perceived. By chunks he meant a range of items like numbers, letters, or words. According to Miller's theory, therefore, people's immediate memory capacity is very limited. They are able to remember only a few words or numbers that they have heard or seen. If you are not familiar with this phenomenon, try out the following exercise: read the first list below (or get someone to read it to you), cover it up, and then try to recall as many of the items as possible. Repeat this for the other lists.

- 3, 12, 6, 20, 9, 4, 0, 1, 19, 8, 97, 13, 84
- cat, house, paper, laugh, people, red, yes, number, shadow, broom, rain, plant, lamp, chocolate, radio, one, coin, jet
- t, k, s, y, r, q, x, p, a, z, l, b, m, e

How many did you correctly remember for each list? Between 5 and 9, as suggested by Miller's theory?

Chunks can also be combined items that are meaningful. For example, it is possible to remember the same number of two-word phrases like hot chocolate, banana split, cream cracker, rock music, cheddar cheese, leather belt, laser printer, tree fern, fluffy duckling, cold rain. When these are all muddled up (e.g., split belt, fern crackers, banana laser, printer cream, cheddar tree, rain duckling, hot rock), however, it is much harder to remember as many chunks. This is mainly because the first set contains all meaningful two-word phrases that have been heard before and require less time to be processed in short-term memory, whereas the second set are completely novel phrases that don't exist in the real world. You need to spend time linking the two parts of the phrase together while trying to memorize them. This takes more time and effort to achieve. Of course it is possible to do if you have time to spend rehearsing them, but if you are asked to do it having heard them only once in quick succession, it is most likely you will remember only a few.

You may be thinking by now, “OK, this is interesting, but what has it got to do with interaction design?” Well not only does this 50-year-old theory have a special place in psychology, it has also made a big impression in HCI. Unfortunately, however, for the wrong reasons. Many designers have heard or read about this phenomenon and think, ah, here is a bit of psychology I can usefully apply to interface design. Would you agree with them? If so, how might people’s ability to only remember 7±2 chunks that they have just read or heard be usefully applied to interaction design?

According to a survey by Bob Bailey (2000), several designers have been led to believe the following guidelines and have even created interfaces based on them:

- Have only seven options on a menu.
- Display only seven icons on a menu bar.
- Never have more than seven bullets in a list.
- Place only seven tabs at the top of a website page.
- Place only seven items on a pull-down menu.

All of these are wrong. Why? The simple reason is that these are all items that can be scanned and recalled visually and hence do not have to be recollected from short-term memory. They don’t just flash up on the screen and disappear, requiring the user to remember them before deciding which one to select. If you were asked to find an item of food most people crave in the set of single words listed above, would you have any problem? No. you would just scan the list until you recognized the one (chocolate) that matched the task and then select it—just as people do when interacting with menus, lists, and tabs—regardless of whether they comprise three or 30 items. What the users are required to do here is not remember as many items as possible, having only heard or seen them once in a sequence, but instead scan through a set of items until they recognize the one they want. Quite a different task. Furthermore, there is much more useful psychological research that can be profitably applied to interaction design.
years. He suggests that it is profitable to view this process as involving two memory processes: recall-directed, followed by recognition-based scanning. The first refers to using memorized information about the required file to get as close to it as possible. The more exact this is, the more success the user will have in tracking down the desired file. The second happens when recall has failed to produce what a user wants and so requires reading through directories of files.

To illustrate the difference between these two processes, consider the following scenario: a user is trying to access a couple of websites visited the day before that compared the selling price of cars offered by different dealers. The user is able to recall the name of one website: “alwayscheapest.com”. She types this in and the website appears. This is an example of successful recall-directed memory. However, the user is unable to remember the name of the second one. She vaguely remembers it was something like ‘autobargains.com’; but typing this in proves unsuccessful. Instead, she switches to scanning her bookmarks/favorites, going to the list of most recent ones saved. She notices two or three URLs that could be the one desired, and on the second attempt she finds the website she is looking for. In this situation, the user initially tries recall-directed memory and when this fails, adopts the second strategy of recognition-based scanning—which takes longer but eventually results in success.

Lansdale proposes that file management systems should be designed to optimize both kinds of memory processes. In particular, systems should be developed that let users use whatever memory they have to limit the area being searched and then represent the information in this area of the interface so as to maximally assist them in finding what they need. Based on this theory, he has developed a prototype system called MEMOIRS that aims at improving users’ recall of information they had encoded so as to make it easier to recall later (Lansdale and Edmunds, 1992). The system was designed to be flexible, providing the user with a range of ways of encoding documents mnemonically, including time stamping (see Figure 3.6), flagging, and attribution (e.g., color, text, icon, sound or image).

More flexible ways of helping users track down the files they want are now beginning to be introduced as part of commercial applications. For example, various search and find tools, like Apple’s Sherlock, have been designed to enable the user to type a full or partial name or phrase that the system then tries to match by listing all the files it identifies containing the requested name/phrase. This method, however, is still quite limited, in that it allows users to encode and retrieve files using only alphanumericals.

<table>
<thead>
<tr>
<th>DESIGN IMPLICATIONS</th>
<th>Memory</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Do not overload users’ memories with complicated procedures for carrying out tasks.</td>
<td>• Provide users with a variety of ways of encoding electronic information (e.g., files, emails, images) to help them remember where they have stored them, through the use of color, flagging, time stamping, icons, etc.</td>
</tr>
<tr>
<td>• Design interfaces that promote recognition rather than recall by using menus, icons, and consistently placed objects.</td>
<td></td>
</tr>
</tbody>
</table>
Figure 3.6 Memoirs tool.
3.2 What is cognition?

**BOX 3.2 A Case of Too Much Memory Load?**

Phone banking has become increasingly popular in the last few years. It allows customers to carry out financial transactions, such as paying bills and checking the balance of their accounts, at their convenience. One of the problems confronting banks that provide this facility, however, is how to manage security concerns. Anyone can phone up a bank and pretend to be someone else. How do the banks prevent fraudulent transactions?

One solution has been to develop rigorous security measures whereby customers must provide various pieces of information before gaining access to their accounts. Typically, these include providing the answers to a combination of the following:

- their zip code or post code
- their mother’s maiden name
- their birthplace
- the last school they attended
- the first school they attended
- a password of between 5 and 10 letters
- a memorable address (not their home)
- a memorable date (not their birthday)

Many of these are relatively easy to remember and recall as they are very familiar. But consider the last two. How easy is it for someone to come up with such memorable information and then be able to recall it readily? Perhaps the customer can give the address and birthday of another member of their family as a memorable address and date. But what about the request for a password? Suppose a customer selects the word “interaction” as a password—fairly easy to remember. The problem is that the bank operators do not ask for the full password, because of the danger that someone in the vicinity might overhear and write it down. Instead they are instructed to ask the customer to provide specific letters from it, like the 7th followed by the 5th. However, such information does not spring readily to mind. Instead, it requires mentally counting each letter of the password until the desired one is reached. How long does it take you to determine the 7th letter of the password “interaction”? How did you do it?

To make things harder, banks also randomize the questions they ask. Again, this is to prevent someone who might be overhearing from memorizing the sequence of information. However, it also means that the customers themselves cannot learn the sequence of information required, meaning they have to generate different information every time they call up the bank.

This requirement to remember and recall such information puts a big memory load on customers. Some people find such a procedure quite nerve-wracking and are prone to forget certain pieces of information. As a coping strategy they write down their details on a sheet of paper. Having such an external representation at hand makes it much easier for them to read off the necessary information rather than having to recall it from memory. However, it also makes them vulnerable to the very fraud the banks were trying to prevent, should anyone else get hold of that piece of paper!

**ACTIVITY 3.3** How else might banks solve the problem of providing a secure system while making the memory load relatively easy for people wanting to use phone banking? How does phone banking compare with online banking?

**Comment**

An alternative approach is to provide the customers with a PIN number (it could be the same as that of their ATM card) and ask them to key this in on their phone keypad, followed by asking one or two questions like their zip or post code, as a backup. Online banking has similar security risks to phone banking and hence this requires a number of security measures to be enforced. These include that the user sets up a nickname and a password. For example, some banks require typing in three randomly selected letters from a password each time the user logs on. This is harder to do online than when asked over the phone, mainly
because it interferes with the normally highly automated process of typing in a password. You really have to think about what letters and numbers are in your password; for example, has it got two letter f's after the number 6, or just one?

Learning can be considered in terms of (i) how to use a computer-based application or (ii) using a computer-based application to understand a given topic. Jack Carroll (1990) and his colleagues have written extensively about how to design interfaces to help learners develop computer-based skills. A main observation is that people find it very hard to learn by following sets of instructions in a manual. Instead, they much prefer to "learn through doing." GUIs and direct manipulation interfaces are good environments for supporting this kind of learning by supporting exploratory interaction and importantly allowing users to "undo" their actions, i.e., return to a previous state if they make a mistake by clicking on the wrong option. Carroll has also suggested that another way of helping learners is by using a "training-wheels" approach. This involves restricting the possible functions that can be carried out by a novice to the basics and then extending these as the novice becomes more experienced. The underlying rationale is to make initial learning more tractable, helping the learner focus on simple operations before moving on to more complex ones.

There have also been numerous attempts to harness the capabilities of different technologies to help learners understand topics. One of the main benefits of interactive technologies, such as web-based, multimedia, and virtual reality, is that they provide alternative ways of representing and interacting with information that are not possible with traditional technologies (e.g., books, video). In so doing, they have the potential of offering learners the ability to explore ideas and concepts in different ways.

**Activity 3.4** Ask a grandparent, child, or other person who has not used a cell phone before to make and answer a call using it. What is striking about their behavior?

**Comment**

First-time users often try to apply their understanding of a land-line phone to operating a cell phone. However, there are marked differences in the way the two phones operate, even for the simplest of tasks, like making a call. First, the power has to be switched on when using a cell phone, by pressing a button (but not so with land-line phones), then the number has to be keyed in, including at all times the area code (in the UK), even if the callee is in the same area (but not so with landlines), and finally the "make a call" button must be pressed (but not so with land-line phones). First-time users may intuitively know how to switch the phone on but not know which key to hit, or that it has to be held down for a couple of seconds. They may also forget to key in the area code if they are in the same area as the person they are calling, and to press the "make a call" key. They may also forget to press the "end a call" button (this is achieved through putting the receiver down with a land-line phone). Likewise, when answering a call, the first-time user may forget to press the "accept a call" button or not know which one to press. These additional actions are quick to learn, once the user understands the need to explicitly instruct the cell phone when they want to make, accept, or end a call.

**Reading, speaking and listening:** these three forms of language processing have both similar and different properties. One similarity is that the meaning of
3.2 What is cognition? 87

sentences or phrases is the same regardless of the mode in which it is conveyed. For example, the sentence "Computers are a wonderful invention" essentially has the same meaning whether one reads it, speaks it, or hears it. However, the ease with which people can read, listen, or speak differs depending on the person, task, and context. For example, many people find listening much easier than reading. Specific differences between the three modes include:

- Written language is permanent while listening is transient. It is possible to reread information if not understood the first time round. This is not possible with spoken information that is being broadcast.

**BOX 3.3 Learning the "Difficult Stuff" through Interactive Multimedia: the Role of Dynalinking**

Children (and adults) often have problems learning the difficult stuff—by this we mean mathematical formulae, notations, laws of physics, and other abstract concepts. One of the main reasons is that they find it difficult to relate their concrete experiences of the physical world with these higher-level abstractions. Research has shown, however, that it is possible to facilitate this kind of learning through the use of interactive multimedia. In particular, different representations of the same process (e.g., a graph, a formula, a sound, a simulation) can be displayed and interacted with in ways that make their relationship with each other more explicit to the learner. This process of linking and manipulating multimedia representations at the interface is called dynalinking (Rogers and Scaife, 1998).

An example where we have found dynalinking beneficial is in helping children and students learn ecological concepts (e.g., food webs, carbon cycles, and energy). In one of our projects, we built a simple ecosystem of a pond using multimedia. The concrete simulation showed various organisms swimming and moving around and occasionally an event where one would eat another (e.g., a snail eating the weed). This was annotated and accompanied by various eating sounds (e.g., chomping) to attract the children’s attention. The children could also interact with the simulation. When an organism was clicked on, it would say what it was and what it ate (e.g., "I'm a weed. I make my own food").

The simulation was dynalinked with other abstract representations of the pond ecosystem. One of these was a food web diagram (See Figure 3.7 in Color Plate 4). The children were encouraged to interact with the interlinked diagrams in various ways and to observe what happened in the concrete simulation when something was changed in the diagram and vice versa. Our study showed that children enjoyed interacting with the simulation and diagrams and, importantly, that they understood much better the purpose of the abstract diagrams and how to use them to reason about the ecosystem.

Dynalinking is a powerful form of interaction and can be used in a range of domains to explicitly show relationships among multiple dimensions, especially when the information to be understood or learned is complex. For example, it can be useful for domains like economic forecasting, molecular modeling, and statistical analyses.

**DESIGN IMPLICATIONS Learning**

- Design interfaces that encourage exploration.
- Design interfaces that constrain and guide users to select appropriate actions.
- Dynamically link representations and abstractions that need to be learned.
• Reading can be quicker than speaking or listening, as written text can be rapidly scanned in ways not possible when listening to serially presented spoken words.

• Listening requires less cognitive effort than reading or speaking. Children, especially, often prefer to listen to narratives provided in multimedia or web-based learning material than to read the equivalent text online.

• Written language tends to be grammatical while spoken language is often ungrammatical. For example, people often start a sentence and stop in mid-sentence, letting someone else start speaking.

• There are marked differences between people in their ability to use language. Some people prefer reading to listening, while others prefer listening. Likewise, some people prefer speaking to writing and vice versa.

• Dyslexics have difficulties understanding and recognizing written words, making it hard for them to write grammatical sentences and spell correctly.

• People who are hard of hearing or hard of seeing are also restricted in the way they can process language.

Many applications have been developed either to capitalize on people's reading, writing and listening skills, or to support or replace them where they lack or have difficulty with them. These include:

• interactive books and web-based material that help people to read or learn foreign languages

• speech-recognition systems that allow users to provide instructions via spoken commands (e.g., word-processing dictation, home control devices that respond to vocalized requests)

• speech-output systems that use artificially generated speech (e.g., written-text-to-speech systems for the blind)

• natural-language systems that enable users to type in questions and give text-based responses (e.g., Ask Jeeves search engine)

• cognitive aids that help people who find it difficult to read, write, and speak. A number of special interfaces have been developed for people who have problems with reading, writing, and speaking (e.g., see Edwards, 1992).

• various input and output devices that allow people with various disabilities to have access to the web and use word processors and other software packages

Helen Petrie and her team at the Sensory Disabilities Research Lab in the UK have been developing various interaction techniques to allow blind people to access the web and other graphical representations, through the use of auditory navigation and tactile diagrams.

Problem-solving, planning, reasoning and decision-making are all cognitive processes involving reflective cognition. They include thinking about what to do, what the options are, and what the consequences might be of carrying out a given action. They often involve conscious processes (being aware of what one is thinking
3.2 What is cognition?

about), discussion with others (or oneself), and the use of various kinds of artifacts,
(e.g., maps, books, and pen and paper). For example, when planning the best route
to get somewhere, say a foreign city, we may ask others, use a map, get instructions
from the web, or a combination of these. Reasoning also involves working through
different scenarios and deciding which is the best option or solution to a given
problem. In the route-planning activity we may be aware of alternative routes and
reason through the advantages and disadvantages of each route before deciding on
the best one. Many a family argument has come about because one member thinks
he or she knows the best route while another thinks otherwise.

Comparing different sources of information is also common practice when
seeking information on the web. For example, just as people will phone around for
a range of quotes, so too, will they use different search engines to find sites that
give the best deal or best information. If people have knowledge of the pros and
cons of different search engines, they may also select different ones for different
kinds of queries. For example, a student may use a more academically oriented one
when looking for information for writing an essay, and a more commercially based
one when trying to find out what's happening in town.

The extent to which people engage in the various forms of reflective cognition
depends on their level of experience with a domain, application, or skill. Novices
tend to have limited knowledge and will often make assumptions about what to do
using other knowledge about similar situations. They tend to act by trial and error,
exploring and experimenting with ways of doing things. As a result they may start
off being slow, making errors and generally being inefficient. They may also act ir-
rationally, following their superstitions and not thinking ahead to the consequences
of their actions. In contrast, experts have much more knowledge and experience
and are able to select optimal strategies for carrying out their tasks. They are likely
to be able to think ahead more, considering what the consequences might be of
opting for a particular move or solution (as do expert chess players).
3.3 Applying knowledge from the physical world to the digital world

As well as understanding the various cognitive processes that users engage in when interacting with systems, it is also useful to understand the way people cope with the demands of everyday life. A well-known approach to applying knowledge about everyday psychology to interaction design is to emulate, in the digital world, the strategies and methods people commonly use in the physical world. An assumption is that if these work well in the physical world, why shouldn't they also work well in the digital world? In certain situations, this approach seems like a good idea. Examples of applications that have been built following this approach include electronic post-it notes in the form of "stickies," electronic "to-do" lists, and email reminders of meetings and other events about to take place. The stickies application displays different-colored notes on the desktop in which text can be inserted, deleted, annotated, and shuffled around, enabling people to use them to remind themselves of what they need to do—analogous to the kinds of externalizing they do when using paper stickies. Moreover, a benefit is that electronic stickies are more durable than paper ones—they don't get lost or fall off the objects they are stuck to, but stay on the desktop until explicitly deleted.

In other situations, however, the simple emulation approach can turn out to be counter-productive, forcing users to do things in bizarre, inefficient, or inappropriate ways. This can happen when the activity being emulated is more complex than is assumed, resulting in much of it being oversimplified and not supported effectively. Designers may notice something salient that people do in the physical world and then fall into the trap of trying to copy it in the electronic world without thinking through how and whether it will work in the new context (remember the poor design of the virtual calculator based on the physical calculator described in the previous chapter).

Consider the following classic study of real-world behavior. Ask yourself, first, whether it is useful to emulate at the interface, and second, how it could be extended as an interactive application.

Tom Malone (1983) carried out a study of the "natural history" of physical offices. He interviewed people and studied their offices, paying particular attention to their filing methods and how they organized their papers. One of his findings was that whether people have messy offices or tidy offices may be more significant than people realize. Messy offices were seen as being chaotic with piles of papers everywhere and little organization. Tidy offices, on the other hand, were seen as being well organized with good use of a filing system. In analyzing these two types of offices, Malone suggested what they reveal in terms of the underlying cognitive behaviors of the occupants. One of his observations was that messy offices may appear chaotic but in reality often reflect a coping strategy by the person: documents are left lying around in obvious places to act as reminders that something has to be done with them. This observation suggests that using piles is a fundamental strategy, regardless of whether you are a chaotic or orderly person.

Such observations about people's coping strategies in the physical world bring to mind an immediate design implication about how to support electronic file
management: to capitalize on the "pile" phenomenon by trying to emulate it in the electronic world. Why not let people arrange their electronic files into piles as they do with paper files? The danger of doing this is that it could heavily constrain the way people manage their files, when in fact there may be far more effective and flexible ways of filing in the electronic world. Mark Lansdale (1988) points out how introducing unstructured piles of electronic documents on a desktop would be counterproductive, in the same way as building planes to flap their wings in the way birds do (someone seriously thought of doing this).

But there may be benefits of emulating the pile phenomenon by using it as a kind of interface metaphor that is extended to offer other functionality. How might this be achieved? A group of interface designers at Apple Computer (Mandler et al., 1992) tackled this problem by adopting the philosophy that they were going to build an application that went beyond physical-world capabilities, providing new functionality that only the computer could provide and that enhanced the interface. To begin their design, they carried out a detailed study of office behavior and analyzed the many ways piles are created and used. They also examined how people use the default hierarchical file-management systems that computer operating systems provide. Having a detailed understanding of both enabled them to create a conceptual model for the new functionality — which was to provide various interactive organizational elements based around the notion of using piles. These included providing the user with the means of creating, ordering, and visualizing piles of files. Files could also be encoded using various external cues, including date and color. New functionality that could not be achieved with physical files included the provision of a scripting facility, enabling files in piles to be ordered in relation to these cues (see Figure 3.8).

Emulating real-world activity at the interface can be a powerful design strategy, provided that new functionality is incorporated that extends or supports the users in their tasks in ways not possible in the physical world. The key is really to understand the nature of the problem being addressed in the electronic world in relation to the various coping and externalizing strategies people have developed to deal with the physical world.
3.4 Conceptual frameworks for cognition

In the previous section we described the pros and cons of applying knowledge of people's coping strategies in the physical world to the digital world. Another approach is to apply theories and conceptual frameworks to interaction design. In this section we examine three of these approaches, which each have a different perspective on cognition:

- mental models
- information processing
- external cognition

3.4.1 Mental models

In Chapter 2 we pointed out that a successful system is one based on a conceptual model that enables users to readily learn a system and use it effectively. What happens when people are learning and using a system is that they develop knowledge of how to use the system and, to a lesser extent, how the system works. These two kinds of knowledge are often referred to as a user's mental model.

Having developed a mental model of an interactive product, it is assumed that people will use it to make inferences about how to carry out tasks when using the interactive product. Mental models are also used to fathom what to do when something unexpected happens with a system and when encountering unfamiliar systems. The more someone learns about a system and how it functions, the more their mental model develops. For example, TV engineers have a "deep" mental model of how TVs work that allows them to work out how to fix them. In contrast.
an average citizen is likely to have a reasonably good mental model of how to operate a TV but a "shallow" mental model of how it works.

Within cognitive psychology, mental models have been postulated as internal constructions of some aspect of the external world that are manipulated enabling predictions and inferences to be made (Craik, 1943). This process is thought to involve the "fleshing out" and the "running" of a mental model (Johnson-Laird, 1983). This can involve both unconscious and conscious mental processes, where images and analogies are activated.

**ACTIVITY 3.5**  
To illustrate how we use mental models in our everyday reasoning, imagine the following two scenarios:

(a) You arrive home from a holiday on a cold winter's night to a cold house. You have a small baby and you need to get the house warm as quickly as possible. Your house is centrally heated. Do you set the thermostat as high as possible or turn it to the desired temperature (e.g. 70°F)?

(b) You arrive home from being out all night, starving hungry. You look in the fridge and find all that is left is an uncooked pizza. The instructions on the packet say heat the oven to 375°F and then place the pizza in the oven for 20 minutes. Your oven is electric. How do you heat it up? Do you turn it to the specified temperature or higher?

**Comment**  
Most people when asked the first question imagine the scenario in terms of what they would do in their own house and choose the first option. When asked why, a typical explanation that is given is that setting the temperature to be as high as possible increases the rate at which the room warms up. While many people may believe this, it is incorrect. Thermostats work by switching on the heat and keeping it going at a constant speed until the desired temperature set is reached, at which point they cut out. They cannot control the rate at which heat is given out from a heating system. Left at a given setting, thermostats will turn the heat on and off as necessary to maintain the desired temperature.

When asked the second question, most people say they would turn the oven to the specified temperature and put the pizza in when they think it is at the desired temperature. Some people answer that they would turn the oven to a higher temperature in order to warm it up more quickly. Electric ovens work on the same principle as central heating and so turning the heat up higher will not warm it up any quicker. There is also the problem of the pizza burning if the oven is too hot!

Why do people use erroneous mental models? It seems that in the above scenarios, they are running a mental model based on a general valve theory of the way something works (Kempton, 1986). This assumes the underlying principle of "more is more": the more you turn or push something, the more it causes the desired effect. This principle holds for a range of physical devices, such as taps and radio controls, where the more you turn them, the more water or volume is given. However, it does not hold for thermostats, which instead function based on the principle of an on-off switch. What seems to happen is that in everyday life people develop a core set of abstractions about how things work, and apply these to a range of devices, irrespective of whether they are appropriate.
Using incorrect mental models to guide behavior is surprisingly common. Just watch people at a pedestrian crossing or waiting for an elevator (lift). How many times do they press the button? A lot of people will press it at least twice. When asked why, a common reason given is that they think it will make the lights change faster or ensure the elevator arrives. This seems to be another example of following the "more is more" philosophy: it is believed that the more times you press the button, the more likely it is to result in the desired effect.

Another common example of an erroneous mental model is what people do when the cursor freezes on their computer screen. Most people will bash away at all manner of keys in the vain hope that this will make it work again. However, ask them how this will help and their explanations are rather vague. The same is true when the TV starts acting up: a typical response is to hit the top of the box repeatedly with a bare hand or a rolled-up newspaper. Again, ask people why and their reasoning about how this behavior will help solve the problem is rather lacking.

The more one observes the way people interact with and behave towards interactive devices, the more one realizes just how strange their behavior can get—especially when the device doesn't work properly and they don't know what to do. Indeed, research has shown that people's mental models of the way interactive devices work is poor, often being incomplete, easily confusable, based on inappropriate analogies, and superstition (Norman, 1983). Not having appropriate mental models available to guide their behavior is what causes people to become very frustrated—often resulting in stereotypical "venting" behavior like those described above.

On the other hand, if people could develop better mental models of interactive systems, they would be in a better position to know how to carry out their tasks efficiently and what to do if the system started acting up. Ideally, they should be able to develop a mental model that matches the conceptual model developed by the designer. But how can you help users to accomplish this? One suggestion is to educate them better. However, many people are resistant to spending much time learning about how things work, especially if it involves reading manuals and other documentation. An alternative proposal is to design systems to be more transparent, so that they are easier to understand. This doesn't mean literally revealing the guts of the system (cf. the way some phone handsets—see Figure 3.9 on Color Plate 4—and iMacs are made of transparent plastic to reveal the colorful electronic circuitry inside), but requires developing an easy-to-understand system image (see Chapter 2 for explanation of this term in relation to conceptual models). Specifically, this involves providing:

- useful feedback in response to user input
- easy-to-understand and intuitive ways of interacting with the system
- clear and easy-to-follow instructions
- appropriate online help and tutorials
- context-sensitive guidance for users, set at their level of experience, explaining how to proceed when they are not sure what to do at a given stage of a task.
How much and what kind of transparency do you think a designer should provide in an interactive product? This is not a straightforward question to answer and depends a lot on the requirements of the targeted user groups. Some users simply want to get on with their tasks and don’t want to have to learn about how the thing they are using works. In this situation, the system should be designed to make it obvious what to do and how to use it. For example, most cell-phone users want a simple “plug-and-play” type interface, where it is straightforward how to carry out functions like saving an address, text messaging, and making a call. Functions that require too much learning can be off-putting. Users simply won’t bother to make the extra effort, meaning that many of the functions provided are never used. Other users like to understand how the device they are using works, in order to make informed decisions about how to carry out their tasks, especially if there are numerous ways of doing something. Some search engines have been designed with this in mind: they provide background information on how they work and how to improve one’s searching techniques (see Figure 3.10).

Thus, the extent to which designers should provide extensive information about how to use a system and how it works, as part of the system image, needs to be appraised in terms of what different people want to know and how much they are prepared to learn.

---

Figure 3.10 The Google search engine, which provides extensive information about how to make your searching strategy more effective.
Another approach to conceptualizing how the mind works has been to use metaphors and analogies (see also Chapter 2). A number of comparisons have been made, including conceptualizing the mind as a reservoir, a telephone network, and a digital computer. One prevalent metaphor from cognitive psychology is the idea that the mind is an information processor. Information is thought to enter and exit the mind through a series of ordered processing stages (see Figure 3.11). Within these stages, various processes are assumed to act upon mental representations. Processes include comparing and matching. Mental representations are assumed to comprise images, mental models, rules, and other forms of knowledge.

The information processing model provides a basis from which to make predictions about human performance. Hypotheses can be made about how long someone will take to perceive and respond to a stimulus (also known as reaction time) and what bottlenecks occur if a person is overloaded with too much information. The best known approach is the human processor model, which models the cognitive processes of a user interacting with a computer (Card et al., 1983). Based on the information processing model, cognition is conceptualized as a series of processing stages, where perceptual, cognitive, and motor processors are organized in relation to one another (see Figure 3.12). The model predicts which cognitive processes are involved when a user interacts with a computer, enabling calculations to be made of how long a user will take to carry out various tasks. This can be very useful when comparing different interfaces. For example, it has been used to compare how well different word processors support a range of editing tasks.

The information processing approach is based on modeling mental activities that happen exclusively inside the head. However, most cognitive activities involve people interacting with external kinds of representations, like books, documents, and computers—not to mention one another. For example, when we go home from wherever we have been we do not need to remember the details of the route because we rely on cues in the environment (e.g., we know to turn left at the red house, right when the road comes to a T-junction, and so on). Similarly, when we are at home we do not have to remember where everything is because information is "out there." We decide what to eat and drink by scanning the items in the fridge, find out whether any messages have been left by glancing at the answering machine to see if there is a flashing light, and so on. To what extent, therefore, can we say that information processing models are truly representative of everyday cognitive activities? Do they adequately account for cognition as it happens in the real world and, specifically, how people interact with computers and other interactive devices?

![Figure 3.11 Human information processing model.](image-url)
Several researchers have argued that existing information processing approaches are too impoverished:

The traditional approach to the study of cognition is to look at the pure intellect, isolated from distractions and from artificial aids. Experiments are performed in closed, isolated rooms, with a minimum of distracting lights or sounds, no other people to assist with the task, and no aids to memory or thought. The tasks are arbitrary ones, invented by the researcher. Model builders build simulations and descriptions of these isolated situations. The theoretical analyses are self-contained little structures, isolated from the world, isolated from any other knowledge or abilities of the person. (Norman, 1990, p. 5)

Instead, there has been an increasing trend to study cognitive activities in the context in which they occur, analyzing cognition as it happens "in the wild"
A central goal has been to look at how structures in the environment can both aid human cognition and reduce cognitive load. A number of alternative frameworks have been proposed, including external cognition and distributed cognition. In this chapter, we look at the ideas behind external cognition—which has focused most on how to inform interaction design (distributed cognition is described in the next chapter).

### 3.4.3 External cognition

People interact with or create information through using a variety of external representations, e.g., books, multimedia, newspapers, web pages, maps, diagrams, notes, drawings, and so on. Furthermore, an impressive range of tools has been developed throughout history to aid cognition, including pens, calculators, and computer-based technologies. The combination of external representations and physical tools have greatly extended and supported people's ability to carry out cognitive activities (Norman, 1993). Indeed, they are such an integral part that it is difficult to imagine how we would go about much of our everyday life without them.

External cognition is concerned with explaining the cognitive processes involved when we interact with different external representations (Scaife and Rogers, 1996). A main goal is to explicate the cognitive benefits of using different representations for different cognitive activities and the processes involved. The main ones include:

1. externalizing to reduce memory load
2. computational offloading
3. annotating and cognitive tracing

#### 1. Externalizing to reduce memory load

A number of strategies have been developed for transforming knowledge into external representations to reduce memory load. One such strategy is externalizing things we find difficult to remember, such as birthdays, appointments, and addresses. Diaries, personal reminders and calendars are examples of cognitive artifacts that are commonly used for this purpose, acting as external reminders of what we need to do at a given time (e.g., buy a card for a relative's birthday).

Other kinds of external representations that people frequently employ are notes, like “stickies,” shopping lists, and to-do lists. Where these are placed in the environment can also be crucial. For example, people often place post-it notes in prominent positions, such as on walls, on the side of computer monitors, by the front door and sometimes even on their hands, in a deliberate attempt to ensure they do remind them of what needs to be done or remembered. People also place things in piles in their offices and by the front door, indicating what needs to be done urgently and what can wait for a while.

Externalizing, therefore, can help reduce people's memory burden by:

- reminding them to do something (e.g., to get something for their mother's birthday)
3.4 Conceptual frameworks for cognition

- reminding them of what to do (e.g., to buy a card)
- reminding them of when to do something (send it by a certain date)

2. Computational offloading

Computational offloading occurs when we use a tool or device in conjunction with an external representation to help us carry out a computation. An example is using pen and paper to solve a math problem.

(a) Multiply 2 by 3 in your head. Easy. Now try multiplying 234 by 456 in your head. Not as easy. Try doing the sum using a pen and paper. Then try again with a calculator. Why is it easier to do the calculation with pen and paper and even easier with a calculator?

(b) Try doing the same two sums using Roman numerals.

Comment

(a) Carrying out the sum using pen and the paper is easier than doing it in your head because you "offload" some of the computation by writing down partial results and using them to continue with the calculation. Doing the same sum with a calculator is even easier, because it requires only eight simple key presses. Even more of the computation has been offloaded onto the tool. You need only follow a simple internalized procedure (key in first number, then the multiplier sign, then next number and finally the equals sign) and then read of the result from the external display.

(b) Using roman numerals to do the same sum is much harder. 2 by 3 becomes II × III, and 234 by 456 becomes CCXXXIII × CCCCCXXXXVI. The first calculation may be possible to do in your head or on a bit of paper, but the second is incredibly difficult to do in your head or even on a piece of paper (unless you are an expert in using Roman numerals or you "cheat" and transform it into Arabic numerals). Calculators do not have Roman numerals so it would be impossible to do on a calculator.

Hence, it is much harder to perform the calculations using Roman numerals than algebraic numerals—even though the problem is equivalent in both conditions. The reason for this is the two kinds of representation transform the task into one that is easy and more difficult, respectively. The kind of tool used also can change the nature of the task to being more or less easy.

3. Annotating and cognitive tracing

Another way in which we externalize our cognition is by modifying representations to reflect changes that are taking place that we wish to mark. For example, people often cross things off in a to-do list to show that they have been completed. They may also reorder objects in the environment, say by creating different piles as the nature of the work to be done changes. These two kinds of modification are called annotating and cognitive tracing:

- Annotating involves modifying external representations, such as crossing off or underlining items.
- **Cognitive tracing** involves externally manipulating items into different orders or structures.

Annotating is often used when people go shopping. People usually begin their shopping by planning what they are going to buy. This often involves looking in their cupboards and fridge to see what needs stocking up. However, many people are aware that they won’t remember all this in their heads and so often externalize it as a written shopping list. The act of writing may also remind them of other items that they need to buy that they may not have noticed when looking through the cupboards. When they actually go shopping at the store, they may cross off items on the shopping list as they are placed in the shopping basket or cart. This provides them with an annotated externalization, allowing them to see at a glance what items are still left on the list that need to be bought.

Cognitive tracing is useful in situations where the current state of play is in a state of flux and the person is trying to optimize their current position. This typically happens when playing games, such as:

- in a card game, the continued rearrangement of a hand of cards into suits, ascending order, or same numbers to help determine what cards to keep and which to play, as the game progresses and tactics change
- in Scrabble, where shuffling around letters in the tray helps a person work out the best word given the set of letters (Maglio et al., 1999)

It is also a useful strategy for letting users know what they have studied in an online learning package. An interactive diagram can be used to highlight all the nodes visited, exercises completed, and units still to study.

A general cognitive principle for interaction design based on the external cognition approach is to provide external representations at the interface that reduce memory load and facilitate computational offloading. Different kinds of information visualizations can be developed that reduce the amount of effort required to make inferences about a given topic (e.g., financial forecasting, identifying pro-

---

**BOX 3.4 Context-Sensitive Information: Shopping Reminders on the Move**

A number of researchers have begun developing wireless communication systems that use GPRS technology to provide **people on the move** with context-sensitive information. This involves providing people with information (such as reminders and to-do lists) whenever it is appropriate to their location. For example, one such system called comMotion, which is being developed at MIT (Marmasse and Schmandt, 2000) uses a speech-output system to inform people when they are driving past a store that sells the groceries they need to buy, such as milk.

How useful is this kind of externalization? Are people really that bad at remembering things? In what way will it be an improvement over other reminder techniques, like shopping lists written on paper or lists stored on PalmPilots or other pocket computers? Sure, there are certain people who have debilitating memory problems (e.g., Alzheimer's) and who may greatly benefit from having such prosthetic memory devices. But what about those who aren’t afflicted? What would happen to them if they started relying more and more on spoken reminders popping up all over the place to tell them what they should be doing when and where? They may well be reminded to buy the milk, but at what price? Losing their own ability to remember?
gramming bugs). In so doing, they can extend or amplify cognition, allowing people to perceive and do activities that they couldn't do otherwise. For example, a number of information visualizations have been developed that present masses of data in a form that makes it possible to make cross comparisons between dimensions at a glance (see Figure 3.13). GUIs can also be designed to reduce memory load significantly, enabling users to rely more on external representations to guide them through their interactions.

3.5 Informing design: from theory to practice

Theories, models, and conceptual frameworks provide abstractions for thinking about phenomena. In particular, they enable generalizations to be made about cognition across different situations. For example, the concept of mental models provides a means of explaining why and how people interact with interactive products in the way they do across a range of situations. The information processing model has been used to predict the usability of a range of different interfaces.

Theory in its pure form, however, can be difficult to digest. The arcane terminology and jargon used can be quite off-putting to those not familiar with it. It also requires much time to become familiar with it—something that designers and engineers can't afford when working to meet deadlines. Researchers have tried to help out by making theory more accessible and practical. This has included translating it into:

- design principles and concepts
- design rules
- analytic methods
- design and evaluation methods
A main emphasis has been on transforming theoretical knowledge into tools that can be used by designers. For example, Card et al's (1983) psychological model of the human processor, mentioned earlier, was simplified into another model called GOMS (an acronym standing for goals, operators, methods, and selection rules). The four components of the GOMS model describe how a user performs a computer-based task in terms of goals (e.g., save a file) and the selection of methods and operations from memory that are needed to achieve them. This model has also been transformed into the keystroke level method that essentially provides a formula for determining the amount of time each of the methods and operations takes. One of the main attractions of the GOMS approach is that it allows quantitative predictions to be made (see Chapter 14 for more on this).

Another approach has been to produce various kinds of design principles, such as the ones we discussed in Chapter 1. More specific ones have also been proposed for designing multimedia and virtual reality applications (Rogers and Scaife, 1998). Thomas Green (1990) has also proposed a framework of cognitive dimensions. His overarching goal is to develop a set of high-level concepts that are both valuable and easy to use for evaluating the designs of informational artifacts, such as software applications. An example dimension from the framework is "viscosity," which simply refers to resistance to local change. The analogy of stirring a spoon in syrup (high viscosity) versus milk (low viscosity) quickly gives the idea. Having understood the concept in a familiar context, Green then shows how the dimension can be further explored to describe the various aspects of interacting with the information structure of a software application. In a nutshell, the concept is used to examine "how much extra work you have to do if you change your mind." Different kinds of viscosity are described, such as knock-on viscosity, where performing one goal-related action makes necessary the performance of a whole train of extraneous actions. The reason for this is constraint density: the new structure that results from performing the first action violates some constraint that must be rectified by the second action, which in turn leads to a different violation, and so on. An example is editing a document using a word processor without widow control. The action of inserting a sentence at the beginning of the document means that the user must then go through the rest of the document to check that all the headers and bodies of text still lie on the same page.
Assignment

The aim of this assignment is for you to elicit mental models from people. In particular, the goal is for you to understand the nature of people's knowledge about an interactive product in terms of how to use it and how it works.

(a) First, elicit your own mental model. Write down how you think a cash machine (ATM) works. Then answer the following questions (abbreviated from Payne, 1991):

- How much money are you allowed to take out?
- If you took this out and then went to another machine and tried to withdraw the same amount, what would happen?
- What is on your card?
- How is the information used?
- What happens if you enter the wrong number?
- Why are there pauses between the steps of a transaction?
- How long are they? What happens if you type ahead during the pauses?
- What happens to the card in the machine?
- Why does it stay inside the machine?
- Do you count the money? Why?

Next, ask two other people the same set of questions.

(b) Now analyze your answers. Do you get the same or different explanations? What do the findings indicate? How accurate are people's mental models of the way ATMs work? How transparent are the ATM systems they are talking about?

(c) Next, try to interpret your findings with respect to the design of the system. Are any interface features revealed as being particularly problematic? What design recommendations do these suggest?

(d) Finally, how might you design a better conceptual model that would allow users to develop a better mental model of ATMs (assuming this is a desirable goal)?

This exercise is based on an extensive study carried out by Steve Payne on people's mental models of ATMs. He found that people do have mental models of ATMs, frequently resorting to analogies to explain how they work. Moreover, he found that people's explanations were highly variable and based on ad hoc reasoning.

Summary

This chapter has explained the importance of understanding users, especially their cognitive aspects. It has described relevant findings and theories about how people carry out their everyday activities and how to learn from these when designing interactive products. It has provided illustrations of what happens when you design systems with the user in mind and what happens when you don't. It has also presented a number of conceptual frameworks that allow ideas about cognition to be generalized across different situations.

Key points

- Cognition comprises many processes, including thinking, attention, learning, memory, perception, decision-making, planning, reading, speaking, and listening.
The way an interface is designed can greatly affect how well people can perceive, attend, learn, and remember how to carry out their tasks.

The main benefits of conceptual frameworks and cognitive theories are that they can explain user interaction and predict user performance.

The conceptual framework of mental models provides a way of conceptualizing the user's understanding of the system.

Research findings and theories from cognitive psychology need to be carefully reinterpreted in the context of interaction design to avoid oversimplification and misapplication.

Further reading

MULLET, K., AND SANO, D. (1995) Designing Visual Interfaces. New Jersey: SunSoft Press. This is an excellent book on the do's and don'ts of interactive graphical design. It includes many concrete examples that have followed (or not) design principles based on cognitive issues.


NORMAN, D. (1993) Things that Make Us Smart. Reading, MA: Addison-Wesley. These two early books by Don Norman provide many key findings and observations about people's behavior and their use of artifacts. They are written in a stimulating and thought-provoking way, using many examples from everyday life to illustrate conceptual issues. He also presents a number of psychological theories, including external cognition, in an easily digestible form.


For more on dynalinking and interactivity see www.cogs.susx.ac.uk/ECOi
4.1 Introduction

Imagine going into school or work each day and sitting in a room all by yourself with no distractions. At first, it might seem blissful. You'd be able to get on with your work. But what if you discovered you had no access to email, phones, the Internet and other people? On top of that there is nowhere to get coffee. How long would you last? Probably not very long. Humans are inherently social: they live together, work together, learn together, play together, interact and talk with each other, and socialize. It seems only natural, therefore, to develop interactive systems that support and extend these different kinds of sociality.

There are many kinds of sociality and many ways of studying it. In this chapter our focus is on how people communicate and collaborate in their working and everyday lives. We examine how collaborative technologies (also called groupware) have been designed to support and extend communication and collaboration. We also look at the social factors that influence the success or failure of user adoption of such technologies. Finally, we examine the role played by ethnographic studies and theoretical frameworks for informing system design.
The main aims of this chapter are to:

- Explain what is meant by communication and collaboration.
- Describe the main kinds of social mechanisms that are used by people to communicate and collaborate.
- Outline the range of collaborative systems that have been developed to support this kind of social behavior.
- Consider how field studies and socially-based theories can inform the design of collaborative systems.

4.2 Social mechanisms in communication and collaboration

A fundamental aspect of everyday life is talking, during which we pass on knowledge to each other. We continuously update each other about news, changes, and developments on a given project, activity, person, or event. For example, friends and families keep each other posted on what's happening at work, school, at the pub, at the club, next door, in soap operas, and in the news. Similarly, people who work together keep each other informed about their social lives and everyday happenings—as well as what is happening at work, for instance when a project is about to be completed, plans for a new project, problems with meeting deadlines, rumors about closures, and so on.

The kinds of knowledge that are circulated in different social circles are diverse, varying among social groups and across cultures. The frequency with which knowledge is disseminated is also highly variable. It can happen continuously throughout the day, once a day, weekly or infrequently. The means by which communication happens is also flexible—it can take place via face to face conversations, telephone, videophone, messaging, email, fax, and letters. Non-verbal communication also plays an important role in augmenting face to face conversation, involving the use of facial expressions, back channeling (the "aha's" and "umms"), voice intonation, gesturing, and other kinds of body language.

All this may appear self-evident, especially when one reflects on how we interact with one another. Less obvious is the range of social mechanisms and practices that have evolved in society to enable us to be social and maintain social order. Various rules, procedures, and etiquette have been established whose function is to let people know how they should behave in social groups. Below we describe three main categories of social mechanisms and explore how technological systems have been and can be designed to facilitate these:

- the use of conversational mechanisms to facilitate the flow of talk and help overcome breakdowns during it
- the use of coordination mechanisms to allow people to work and interact together
- the use of awareness mechanisms to find out what is happening, what others are doing and, conversely, to let others know what is happening
4.2 Social mechanisms in communication and collaboration

4.2.1 Conversational mechanisms

Talking is something that is effortless and comes naturally to most people. And yet holding a conversation is a highly skilled collaborative achievement, having many of the qualities of a musical ensemble. Below we examine what makes up a conversation. We begin by examining what happens at the beginning:

A: Hi there.
B: Hi!
C: Hi.
A: All right?
C: Good. How's it going?
A: Fine, how are you?
C: Good.
B: OK. How's life treating you?

Such mutual greetings are typical. A dialog may then ensue in which the participants take turns asking questions, giving replies, and making statements. Then when one or more of the participants wants to draw the conversation to a close, they do so by using either implicit or explicit cues. An example of an implicit cue is when a participant looks at his watch, signaling indirectly to the other participants that he wants the conversation to draw to a close. The other participants may choose to acknowledge this cue or carry on and ignore it. Either way, the first participant may then offer an explicit signal, by saying, "Well, I must be off now. Got work to do," or, "Oh dear, look at the time. Must dash. Have to meet someone." Following the acknowledgment by the other participants of such implicit and explicit signals, the conversation draws to a close, with a farewell ritual. The different participants take turns saying, "Bye," "Bye then," "See you," repeating themselves several times, until they finally separate.

Such conversational mechanisms enable people to coordinate their "talk" with one another, allowing them to know how to start and stop. Throughout a conversation further "turn-taking" rules are followed, enabling people to know when to listen, when it is their cue to speak, and when it is time for them to stop again to allow the others to speak. Sacks, Schegloff and Jefferson (1978)—who are famous for their work on conversation analysis—describe these in terms of three basic rules:

- rule 1—the current speaker chooses the next speaker by asking an opinion, question, or request
- rule 2—another person decides to start speaking
- rule 3—the current speaker continues talking

The rules are assumed to be applied in the above order, so that whenever there is an opportunity for a change of speaker to occur (e.g., someone comes to the end of a sentence), rule 1 is applied. If the listener to whom the question or opinion is addressed does not accept the offer to take the floor, the second rule is applied and
someone else taking part in the conversation may take up the opportunity and offer a view on the matter. If this does not happen then the third rule is applied and the current speaker continues talking. The rules are cycled through recursively until someone speaks again.

To facilitate rule following, people use various ways of indicating how long they are going to talk and on what topic. For example, a speaker might say right at the beginning of their turn in the conversation that he has three things to say. A speaker may also explicitly request a change in speaker by saying, "OK, that's all I want to say on that matter. So, what do you think?" to a listener. More subtle cues to let others know that their turn in the conversation is coming to an end include the lowering or raising of the voice to indicate the end of a question or the use of phrases like, "You know what I mean?" or simply, "OK?" Back channeling (uh-huh, mmm), body orientation (e.g., moving away from or closer to someone), gaze (staring straight at someone or glancing away), and gesture (e.g., raising of arms) are also used in different combinations when talking, to signal to others when someone wants to hand over or take up a turn in the conversation.

Another way in which conversations are coordinated and given coherence is through the use of adjacency pairs (Shegloff and Sacks, 1973). Utterances are assumed to come in pairs in which the first part sets up an expectation of what is to come next and directs the way in which what does come next is heard. For example, A may ask a question to which B responds appropriately:

A: So shall we meet at 8:00?
B: Um, can we make it a bit later, say 8:30?

Sometimes adjacency pairs get embedded in each other, so it may take some time for a person to get a reply to their initial request or statement:

A: So shall we meet at 8:00?
B: Wow, look at him.
A: Yes, what a funny hairdo!
B: Um, can we make it a bit later, say 8:30?

For the most part people are not aware of following conversational mechanisms, and would be hard pressed to articulate how they can carry on a conversation. Furthermore, people don't necessarily abide by the rules all the time. They may interrupt each other or talk over each other, even when the current speaker has clearly indicated a desire to hold the floor for the next two minutes to finish an argument. Alternatively, a listener may not take up a cue from a speaker to answer a question or take over the conversation, but instead continue to say nothing even though the speaker may be making it glaringly obvious it is the listener's turn to say something. Many a time a teacher will try to hand over the conversation to a student in a seminar, by staring at her and asking a specific question, only to see the student look at the floor, and say nothing. The outcome is an embarrassing silence, followed by either the teacher or another student picking up the conversation again.

Other kinds of breakdowns in conversation arise when someone says something that is ambiguous and the other person misinterprets it to mean something else. In
such situations the participants will collaborate to overcome the misunderstanding by using repair mechanisms. Consider the following snippet of conversation between two people:

A: Can you tell me the way to get to the Multiplex Ranger cinema?

B: Yes, you go down here for two blocks and then take a right (pointing to the right), go on till you get to the lights and then it is on the left.

A: Oh, so I go along here for a couple of blocks and then take a right and the cinema is at the lights (pointing ahead of him)?

A: No, you go on this street for a couple of blocks (gesturing more vigorously than before to the street to the right of him while emphasizing the word "this").

B: Ahhhh! I thought you meant that one: so it's this one (pointing in same direction as the other person).

A: Uh-hum, yes that's right, this one.

Detecting breakdowns in conversation requires the speaker and listener to be attending to what the other says (or does not say). Once they have understood the nature of the failure, they can then go about repairing it. As shown in the above example, when the listener misunderstands what has been communicated, the speaker repeats what she said earlier, using a stronger voice intonation and more exaggerated gestures. This allows the speaker to repair the mistake and be more explicit to the listener, allowing her to understand and follow better what they are saying. Listeners may also signal when they don't understand something or want further clarification by using various tokens, like "Huh?", "Quoi?" or "What?" (Scheffleff, 1982) together with giving a puzzled look (usually frowning). This is especially the case when the speaker says something that is vague. For example, they might say "I want it" to their partner, without saying what it is they want. The partner may reply using a token or, alternatively, explicitly ask, "What do you mean by it?"

Taking turns also provides opportunities for the listener to initiate repair or request clarification, or for the speaker to detect that there is a problem and to initiate repair. The listener will usually wait for the next turn in the conversation before interrupting the speaker, to give the speaker the chance to clarify what is being said by completing the utterance (Suchman, 1987).

How do people repair breakdowns in conversations when using the phone or email?

Comment

In these settings people cannot see each other and so have to rely on other means of repairing their conversations. Furthermore, there are more opportunities for breakdowns to occur and fewer mechanisms available for repair. When a breakdown occurs over the phone, people will often shout louder, repeating what they said several times, and use stronger intonation. When a breakdown occurs via email, people may literally spell out what they meant, making things much more explicit in a subsequent email. If the message is beyond repair they may resort to another mode of communication that allows greater flexibility of expression, either telephoning or speaking to the recipient face to face.
Kinds of conversations

Conversations can take a variety of forms, such as an argument, a discussion, a heated debate, a chat, a tête-à-tête, or giving someone a "telling off." A well-known distinction in conversation types is between formal and informal communication. Formal communication involves assigning certain roles to people and prescribing *a priori* the types of turns that people are allowed to take in a conversation. For example, at a board meeting, it is decided who is allowed to speak, who speaks when, who manages the turn-taking, and what the participants are allowed to talk about.

In contrast, informal communication is the chat that goes on when people socialize. It also commonly happens when people bump into each other and talk briefly. This can occur in corridors, at the coffee machine, when waiting in line, and walking down the street. Informal conversations include talking about impersonal things like the weather (a favorite) and the price of living, or more personal things, like how someone is getting on with a new roommate. It also provides an opportunity to pass on gossip, such as who is going out to dinner with whom. In office settings, such chance conversations have been found to serve a number of functions, including coordinating group work, transmitting knowledge about office culture, establishing trust, and general team building (Kraut et al, 1990). It is also the case that people who are in physical proximity, such as those whose offices or desks are close to one another, engage much more frequently in these kinds of informal chats than those who are in different corridors or buildings. Most companies and organizations are well aware of this and often try to design their office space so that people who need to work closely together are placed close to one another in the same physical space.

4.2.2 Designing collaborative technologies to support conversation

As we have seen, "talk" and the way it is managed is integral to coordinating social activities. One of the challenges confronting designers is to consider how the different kinds of communication can be facilitated and supported in settings where there may be obstacles preventing it from happening "naturally." A central concern has been to develop systems that allow people to communicate with each other when they are in physically different locations and thus not able to communicate in the usual face to face manner. In particular, a key issue has been to determine how to allow people to carry on communicating as if they were in the same place, even though they are geographically separated—sometimes many thousands of miles apart.

*Email*, videoconferencing, videophones, computer conferencing, chatrooms and messaging are well-known examples of some of the collaborative technologies that have been developed to enable this to happen. Other less familiar systems are collaborative virtual environments (CVEs) and media spaces. CVEs are virtual worlds where people meet and chat. These can be 3D graphical worlds where users explore rooms and other spaces by teleporting themselves around in the guise of avatars (See Figure 4.1 on Color Plate 5), or text and graphical "spaces" (often called MUDs and MOOs) where users communicate with each other via some
form of messaging. Media spaces are distributed systems comprising audio, video, and computer systems that "extend the world of desks, chairs, walls and ceilings" (Harrison et al., 1997), enabling people distributed over space and time to communicate and interact with one another as if they were physically present. The various collaborative technologies have been designed to support different kinds of communication, from informal to formal and from one-to-one to many-to-many conversations. Collectively, such technologies are often referred to as computer-mediated communication (CMC).

**ACTIVITY 4.2**

Do you think it is better to develop technologies that will allow people to talk at a distance as if they were face to face, or to develop technologies that will support new ways of conversing?

**Comment**

On the one hand, it seems a good idea to develop technologies supporting people communicating at a distance in the way they hold conversations in face to face situations. After all, this means of communicating is so well established and second nature to people. Phones and videoconferencing have been developed to essentially support face to face conversations. It is important to note, however, that conversations held in this way are not the same as when face to face. People have adapted the way they hold conversations to fit in with the constraints of the respective technologies. As noted earlier, they tend to shout more when misunderstood over the phone. They also tend to speak more loudly when talking on the phone, since they can't monitor how well the person can hear them at the other end of the phone. Likewise, people tend to project themselves more when videoconferencing. Turn-taking appears to be much more explicit, and greetings and farewells more ritualized.

On the other hand, it is interesting to look at how the new communication technologies have been extending the way people talk and socialize. For example, SMS text messaging has provided people with quite different ways of having a conversation at a distance. People (especially teenagers) have evolved a new form of fragmentary conversation (called "texting") that they continue over long periods. The conversation comprises short phrases that are typed in, using the key pad, commenting on what each is doing or thinking, allowing the other to keep posted on current developments. These kinds of "streamlined" conversations are coordinated simply by taking turns sending and receiving messages. Online chatting has also enabled effectively hundreds and even thousands of people to take part in the same conversations, which is not possible in face to face settings.

The range of systems that support computer-mediated communication is quite diverse. A summary table of the different types is shown in Table 4.1, highlighting how they support, extend and differ from face to face communication. A conventionally accepted classification system of CMC is to categorize them in terms of either synchronous or asynchronous communication. We have also included a third category: systems that support CMC in combination with other collaborative activities, such as meetings, decision-making, learning, and collaborative authoring of documents. Although some communication technologies are not strictly speaking computer-based (e.g., phones, video-conferencing) we have included these in the classification of CMC, as most now are display-based and interacted with or controlled via an interface. (For more detailed overviews of CMC, see Dix et al. (Chapter 13, 1998) and Baecker et al. (Part III and IV, 1993).
Table 4.1  Classification of computer-mediated communication (CMC) into three types: (I) Synchronous communication, (ii) Asynchronous communication and (iii) CMC combined with other activity

i. Synchronous communication
Where conversations in real time are supported by letting people talk with each other either using their voices or through typing. Both modes seek to support non-verbal communication to varying degrees.

**Examples:**
- Talking with voice: video phones, video conferencing (desktop or wall), media spaces.
- Talking via typing: text messaging (typing in messages using cell phones), instant messaging (real-time interaction via PCs) chatrooms, collaborative virtual environments (CVEs).

**New kinds of functionality:**
- CVEs allow communication to take place via a combination of graphical representations of self (in the form of avatars) with a separate chatbox or overlaying speech bubbles.
- CVEs allow people to represent themselves as virtual characters, taking on new personas (e.g., opposite gender), and expressing themselves in ways not possible in face-to-face settings.
- CVEs, MUDs and chatrooms have enabled new forms of conversation mechanisms, such as multi-turn-taking, where a number of people can contribute and keep track of a multi-streaming text-based conversation.
- Instant messaging allows users to multitask by holding numerous conversations at once.

**Benefits:**
- Not having to physically face people may increase shy people's confidence and self-esteem to converse more in "virtual" public.
- It allows people to keep abreast of the goings-on in an organization without having to move from their office.
- It enables users to send text and images instantly between people using instant messaging.
- In offices, instant messaging allows users to fire off quick questions and answers without the time lag of email or phone-tag.

**Problems:**
- Lack of adequate bandwidth has plagued video communication, resulting in poor-quality images that frequently break up, judder, have shadows, and appear as unnatural images.
- It is difficult to establish eye contact (normally an integral and subconscious part of face-to-face conversations) in CVEs, video conferencing, and videophones.
- Having the possibility of hiding behind a persona, a name, or an avatar in a chatroom gives people the opportunity to behave differently. Sometimes this can result in people becoming aggressive or intrusive.

ii. Asynchronous communication
Where communication between participants takes place remotely and at different times. It relies not on time-dependent turn-taking but on participants initiating communication and responding to others when they want or are able to do so.

**Examples:**
- *email*, bulletin boards, newsgroups, computer conferencing

**New kinds of functionality:**
- Attachments of different sorts (including annotations, images, music) for email and computer conferencing can be sent.
- Messages can be archived and accessed using various search facilities.

**Benefits:**
- Ubiquity: Can read any place, any time.
- Flexibility: Greater autonomy and control of when and how to respond, so can attend to it in own time rather than having to take a turn in a conversation at a particular cue.
- Powerful: Can send the same message to many people.
- Makes some things easier to say: Do not have to interact with person so can be easier to say things than when face to face (e.g., announcing sudden death of colleague, providing feedback on someone's performance).

(Continued)
Table 4.1 (Continued)

Problems:
- Flaming: When a user writes incensed angry email expressed in uninhibited language that is much stronger than normally used when interacting with the same person face to face. This includes the use of impolite statements, capitalized sentences or words, swearing, and superlatives. Such "charged" communication can lead to misunderstandings and bad feelings among the recipients.
- Overload: Many people experience message overload, receiving over 30 emails or other messages a day. They find it difficult to cope and may overlook an important message while working through their ever increasing pile of email—especially if they have not read it for a few days. Various interface mechanisms have been designed to help people manage their email better, including filtering, threading, and the use of signaling to indicate the level of importance of a message (via the sender or recipient), through color coding, bold font, or exclamation marks placed beside a message.
- False expectations: An assumption has evolved that people will read their messages several times a day and reply to them there and then. However, many people have now reverted to treating email more like postal mail, replying when they have the time to do so.

iii. CMC combined with other activity
People often talk with each other while carrying out other activities. For example, designing requires people to brainstorm together in meetings, drawing on whiteboards, making notes, and using existing designs. Teaching involves talking with students as well as writing on the board and getting students to solve problems collaboratively. Various meeting- and decision-support systems have been developed to help people work or learn while talking together.

Examples:
- Customized electronic meeting rooms have been built that support people in face-to-face meetings, via the use of networked workstations, large public displays, and shared software tools, together with various techniques to help decision-making. One of the earliest systems was the University of Arizona's GroupSystem (see Figure 4.2).

![Figure 4.2](image_url)
Networked classrooms: Recently schools and universities have realized the potential of using combinations of technologies to support learning. For example, wireless communication, portable devices and interactive whiteboards are being integrated in classroom settings to allow the teacher and students to learn and communicate with one another in novel interactive ways (see Figure 4.3).

- Argumentation tools which record the design rationale and other arguments used in a discussion that lead to decisions in a design (e.g., gIBIS, Conklin and Begeman, 1989). These are mainly designed for people working in the same physical location.
- Shared authoring and drawing tools that allow people to work on the same document at the same time. This can be remotely over the web (e.g., shared authoring tools like Shredit) or on the same drawing surface in the same room using multiple mouse cursors (e.g., KidPad, Benford et al., 2000).

**New kinds of functionality:**
- Allows new ways of collaboratively creating and editing documents.
- Supports new forms of collaborative learning.
- Integrates different kinds of tools.

**Benefits:**
- Supports talking while carrying out other activities at the same time, allowing multi-tasking — which is what happens in face-to-face settings.
- Speed and efficiency: allows multiple people to be working an same document at same time.
- Greater awareness: allows users to see how one another are progressing in real time.

**Problems:**
- WYSIWIS (what you see is what I see): It can be difficult to see what other people are referring to when in remote locations, especially if the document is large and different users have different parts of the document on their screens.
- Floor control: Users may want to work on the same piece of text or design, potentially resulting in file conflicts. These can be overcome by developing various social and technological floor-control policies.
One of the earliest technological innovations (besides the telephone and telegraph) developed for supporting conversations at a distance was the videophone. Despite numerous attempts by the various phone companies to introduce them over the last 50 years (see Figure 4.4), they have failed each time. Why do you think this is so?

Comment

One of the biggest problems with commercial videophones is that the bandwidth is too low, resulting in poor resolution and slow refresh rate. The net effect is the display of unacceptable images: the person in the picture appears to move in sudden jerks; shadows are left behind when a speaker moves, and it is difficult to read lips or establish eye contact. There is also the social acceptability issue of whether people want to look at pocket-sized images of each other when talking. Sometimes you don't want people to see what state you are in or where you are.

Another innovation has been to develop systems that allow people to communicate and interact with each other in ways not possible in the physical world. Rather than try to imitate or facilitate face to face communication (like the above systems), designers have tried to develop new kinds of interactions. For example, ClearBoard was developed to enable facial expressions of participants to be made visible to others by using a transparent board that showed their face to the others (Ishii et al., 1993). HyperMirror was designed to provide an environment in which the participants could feel they were in the same virtual place even

Figure 4.4 (a) One of British Telecom's early videophones and (b) a recent mobile "visualphone" developed in Japan.
A number of researchers have tried to capitalize on the social phenomenon of informal communication and the important role it plays at work. In particular, they have been interested in finding ways of using audio-video links to mimic physical settings that are conducive to informal communication for people who are geographically separated. One of the first systems to be built, at Bellcore in 1989, was the VideoWindow System (see Figure 4.5). The goal was to design a shared space that would allow people in different locations to carry on a conversation as they would do if sitting in the same room drinking coffee together. Two lounge areas that were 50 miles apart were connected with high-bandwidth video channels and full-duplex four-channel audio. Connecting them was a 3 x 8 foot “picture-window” onto which video images were projected. The large size was meant to allow viewers to see a room of people roughly the same size as themselves. The system was designed to be active 24 hours a day, so that anyone entering one room could speak to whoever happened to be in the other room.

A study by Kraut et al. (1990) of how effective the system was showed that in general, many of the interactions that took place between the remote conversants were indeed indistinguishable from similar face-to-face interactions—with the exception that they spoke a bit louder and constantly talked about the video system. However, they also found that people who were in the same room tended to talk more with each other than with those in the video-linked room. Various usability problems were identified as contributing to this reluctance to talk with video-images of other people. One of these was the tendency for people to move closer to the picture window to strike up a conversation with someone (which is what one would normally do in a face-to-face setting); this had the opposite effect to what the person intended, as it moved his or her head out of the picture and also out of microphone range, meaning he or she could not be seen or heard. Thus rather than getting nearer to the other person, this behavior had the counter-intuitive effect of removing him or her from the “picture.” Moreover, there was no way for participants to know whether they were being seen and heard by the others in the other room. This inability to monitor how others are or are not “receiving” you caused numerous problems. Another problem was that the system allowed only public conversations, meaning that

Figure 4.5 Diagram of VideoWindow system in use.
they could be heard by everyone in the rooms. Such public broadcasting contrasts with how people normally engage in informal face-to-face conversations, where they will often whisper and conspire with each other when a topic becomes more private or secret. Such private conversations clearly could not be supported by the VideoWindow system.

A number of other synchronous audio-video systems have since been developed that have tried to incorporate different kinds of conversational mechanisms to facilitate informal communication in video-linked locations. For example, Cruiser was designed to support informal communication by placing separate audio and video equipment on the desktop of each person who was connected to the system (Fish, 1989). This set-up differed from the VideoWindow system in that it enabled both public and private interactions to take place. It also provided additional functionality that allowed people to initiate conversations by typing in a cruise command followed by a question like, "I'm bored. Anyone want a chat?" or "Can someone help me?"—the aim here being explicitly to encourage people to engage in the kind of talk that they normally do when they bump into each other, but this time over the computer network. A further conversation mechanism built into Cruiser was a "glance" feature that allowed users to check whether the person they wanted to talk to was in fact available before trying to initiate a conversation.

Commercial systems are now available that support multiple connections among sites. These can be very useful for virtual centers that have multiple groups working at a number of different sites. For example, the Distributed Systems Technology Center (a research collaboration between Australian universities and industries) has been using a commercial videoconferencing system (Polycom Viewstation 128) that enables people at its main sites (Brisbane, Sydney, and Melbourne) to keep in contact via regular informal and formal meetings. Formal meetings among the sites involve all the teams (over a hundred people) getting together and reporting on their projects. Each site has a camera which it controls, projecting different images to the other sites of what is happening at the site. This can be an image of the person who is doing the talking, a person being talked about, or a person who is being funny. The images from the different sites are displayed side by side on a large screen at each site. Weekly informal project meetings are also held among small groups. Informal chatting is also supported by having a continuous virtual presence via a video wall of the kitchen of one Queensland University site in another, and vice versa (see Figure 4.6).

Figure 4.6. A commercial videoconferencing system being used to support informal chatting among researchers across sites at Queensland University. In contrast to the VideoWindow system, a window of each site is shown in the top left-hand corner of the display to let the participants monitor their own behavior.
Figure 4.7 Hypermirror in action, showing perception of virtual personal space. (a) A woman is in one room (indicated by arrow on screen), (b) while a man and another woman in the other room chat to each other. They move apart when they notice they are "overlapping" her and (c) virtual personal space is established.

though they were physically in different places (Morikawa and Maesako, 1998). Mirror reflections of people in different places were synthesized and projected onto a single screen, so that they appeared side by side in the same virtual space. In this way, the participants could see both themselves and others in the same seamless virtual space. Observations of people using the system showed how quickly they adapted to perceiving themselves and others in this way. For example, participants quickly became sensitized to the importance of virtual personal space, moving out of the way if they perceived they were overlapping someone else on the screen (see Figure 4.7).

4.2.3 Coordination mechanisms

Coordination takes place when a group of people act or interact together to achieve something. For example, consider what is involved in playing a game of basketball. Teams have to work out how to play with each other and to plan a set of tactics that they think will outwit the other team. For the game to proceed both teams need to follow (and sometimes contravene) the rules of the game. An incredible amount of coordination is required within a team and between the competing teams in order to play.

In general, collaborative activities require us to coordinate with each other, whether playing a team game, moving a piano, navigating a ship, working on a large software project, taking orders and serving meals in a restaurant, constructing a bridge or playing tennis. In particular, we need to figure out how to interact with one another to progress with our various activities. To help us we use a number of coordinating mechanisms. Primarily, these include:

- verbal and non-verbal communication
- schedules, rules and conventions
- shared external representations
Verbal and non-verbal communication

When people are working closely together they talk to each other, issuing commands and letting others know how they are progressing with their part. For example, when two or more people are collaborating together, as in moving a piano, they shout to each other commands like "Down a bit, left a bit, now straight forward" to coordinate their actions with each other. As in a conversation, nods, shakes, winks, glances, and hand-raising are also used in combination with such coordination "talk" to emphasize and sometimes replace it.

In formal settings, like meetings, explicit structures such as agendas, memos, and minutes are employed to coordinate the activity. Meetings are chaired, with secretaries taking minutes to record what is said and plans of actions agreed upon. Such minutes are subsequently distributed to members to remind them of what was agreed in the meeting and for those responsible to act upon what was agreed.

For time-critical and routinized collaborative activities, especially where it is difficult to hear others because of the physical conditions, gestures are frequently used (radio-controlled communication systems may also be used). Various kinds of hand signals have evolved, with their own set of standardized syntax and semantics. For example, the arm and baton movements of a conductor coordinate the different players in an orchestra, while the arm and baton movements of a ground marshal at an airport signal to a pilot how to bring the plane into its allocated gate.

ACTIVITY 4.4

How much communication is non-verbal? Watch a soap opera on the TV and turn down the volume and look at the kinds and frequency of gestures that are used. Are you able to understand what is going on? How do radio soaps compensate for not being able to use non-verbal gestures? How do people compensate when chatting online?

Comment

Soaps are good to watch for observing non-verbal behavior as they tend to be overcharged, with actors exaggerating their gestures and facial expressions to convey their emotions. It is often easy to work out what kind of scene is happening from their posture, body movement, gestures, and facial expressions. In contrast, actors on the radio use their voice a lot more, relying on intonation and surrounding sound effects to help convey emotions. When chatting online, people use emoticons and other specially evolved verbal codes.

Schedules, rules, and conventions

A common practice in organizations is to use various kinds of schedules to organize the people who are part of it. For example, consider how a university manages to coordinate the people within it with its available resources. A core task is allocating the thousands of lectures and seminars that need to be run each week with the substantially smaller number of rooms available. A schedule has to be devised
that allows students to attend the lectures and seminars for their given courses, taking into account numerous rules and constraints. These include:

- A student cannot attend more than one lecture at a given time.
- A professor cannot give more than one lecture or seminar at a given time.
- A room cannot be allocated to more than one seminar or lecture at a given time.
- Only a certain number of students can be placed in a room, depending on its size.

**BOX 4.2 Shared Calendars—My Time Or Yours?**

Trying to schedule meetings for different people in an organization to attend can be a nightmare. Typically, a secretary will send out email or try to call those who need to be assembled for a given meeting. Some of those people may not be there at the time they are contacted and so the secretary must wait for them to reply before being able to set up a meeting. In the meantime, the others who have replied (but have not heard back about the meeting) may start to fill up the times they said they would be free with other engagements. When the secretary eventually gets back to them with a proposed date, it is often too late. The consequence is that the secretary will have to start again, coming up with a new set of times. The more people to be scheduled, the more difficult it becomes to keep track of people’s calendars in order to find mutually free time slots to arrange a meeting. All in all, it is a very time-consuming and laborious activity, one in which it would seem a computer-based scheduling tool would be very useful.

Indeed, a number of shared calendars have been developed. Some of the most recent ones have been developed as web applications, allowing individuals to use them both as personal calendars, reminding them of events they have to attend, and as public calendars, broadcasting to anyone accessing their web pages, when they are busy and when they are free.

A number of studies of the implementation of shared calendars in various organizations have found them to be successful computer-supported coordination tools. For example, a study of one system, called MeetingMaker, showed that the users found it had greatly simplified meeting scheduling and was much faster than manual scheduling (Mosier and Tammaro, 1997). The shared tool provided a range of facilities, including supporting group scheduling, resource and room scheduling, to-do lists and various permissions to permit others to schedule meetings. The system could also inform individual users of upcoming events and meetings via a pop-up dialog box.

Other shared calendars, however, have not been so successful. These are usually ones that have been designed to allow people to look at a person’s schedule, and upon finding a free slot, book a meeting with them without any form of negotiation. The normal etiquette when arranging a meeting is to ask someone when they are free to meet and suggest a number of dates and times. When a meeting is simply imposed, people can feel that their privacy has been invaded, especially if they had planned to use the allocated slot to work on something else. A typical response is to simply stop using the shared calendar. The problem with one person doing this, however, is that the rest of the group cannot continue using the shared calendar as a coordinating tool, since that person is now excluded.

The more successful scheduling tools, like MeetingMaker, have overcome this sensitive privacy issue by providing users with a proxy function. This allows users to mark off parts of their calendars as “private” while giving permissions to others to read and/or write to their calendars. Providing this more flexible control allows users to decide a priori which times of the week they will make themselves available for meetings and which times they want to keep private to get on with their own work, without revealing to others what they are up to.
Other coordinating mechanisms that are employed by groups working together are rules and conventions. These can be formal or informal. Formal rules, like the compulsory attendance of seminars, writing monthly reports, and filling in of timesheets, enable organizations to maintain order and keep track of what its members are doing. Conventions, like keeping quiet in a library or removing meal trays after finishing eating in a cafeteria, are a form of courtesy to others.

**Shared external representations**

Shared external representations are commonly used to coordinate people. We have already mentioned one example, that of shared calendars that appear on user’s monitors as graphical charts, email reminders, and dialog boxes. Other kinds that are commonly used include forms, checklists, and tables. These are presented on public noticeboards or as part of other shared spaces. They can also be attached to documents and folders. They function by providing external information of who is working on what, when, where, when a piece of work is supposed to be finished, and who it goes to next. For example, a shared table of who has completed the checking of files for a design project (see Figure 4.8), provides the necessary information from which other members of the group can at a glance update their model of the current progress of that project. Importantly, such external representations can be readily updated by annotating. If a project is going to take longer than planned, this can be indicated on a chart or table by extending the line representing it, allowing others to see the change when they pass by and glance up at the whiteboard.

Shared externalizations allow people to make various inferences about the changes or delays with respect to their effect on their current activities. Accordingly,

<table>
<thead>
<tr>
<th>Sheet no</th>
<th>Gary copied in</th>
<th>Kate &amp; Gary plot file created</th>
<th>Mark checked by Phil</th>
<th>Kate plot sent</th>
<th>Mark plot file created</th>
<th>Mark plot sent mylar</th>
</tr>
</thead>
<tbody>
<tr>
<td>59656</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>S7</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>S8</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>S9</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>S10</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

*Figure 4.8* An external representation used to coordinate collaborative work in the form of a print-out table showing who has completed the checking of files and who is down to do what.
they may need to reschedule their work and annotate the shared workplan. In so doing, these kinds of coordination mechanisms are considered to be tangible, providing important representations of work and responsibility that can be changed and updated as and when needed.

4.2.4 Designing collaborative technologies to support coordination

Shared calendars, electronic schedulers, project management tools, and workflow tools that provide interactive forms of scheduling and planning are some of the main kinds of collaborative technologies that have been developed to support coordination. A specific mechanism that has been implemented is the use of conventions. For example, a shared workspace system (called POLITeam) that supported email and document sharing to allow politicians to work together at different sites introduced a range of conventions. These included how folders and files should be organized in the shared workspace. Interestingly, when the system was used in practice, it was found that the conventions were often violated (Mark, et al., 1997). For example, one convention that was set up was that users should always type in the code of a file when they were using it. In practice, very few people did this, as pointed out by an administrator: "They don't type in the right code. I must correct them. I must sort the documents into the right archive. And that's annoying".

The tendency of people not to follow conventions can be due to a number of reasons. If following conventions requires additional work that is extraneous to the users' ongoing work, they may find it gets in the way. They may also perceive the convention as an unnecessary burden and "forget" to follow it all the time. Such "productive laziness" (Rogers, 1993) is quite common. A simple analogy to everyday life is forgetting to put the top back on the toothpaste tube: it is a very simple convention to follow and yet we are all guilty sometimes (or even all the time) of not doing this. While such actions may only take a tiny bit of effort, people often don't do them because they perceive them as tedious and unnecessary. However, the consequence of not doing them can cause grief to others.

When designing coordination mechanisms it is important to consider how socially acceptable they are to people. Failure to do so can result in the users not using the system in the way intended or simply abandoning it. A key part is getting the right balance between human coordination and system coordination. Too much system control and the users will rebel. Too little control and the system breaks down. Consider the example of file locking, which is a form of concurrency control. This is used by most shared applications (e.g., shared authoring tools, file-sharing systems) to prevent users from clashing when trying to work on the same part of a shared document or file at the same time. With file locking, whenever someone is working on a file or part of it, it becomes inaccessible to others. Information about who is using the file and for how long may be made available to the other users, to show why they can't work on a particular file. When file-locking mechanisms are used in this way, however, they are often considered too rigid as a form of coordination, primarily because they don't let other users negotiate with the first user about when they can have access to the locked file.
4.2 Social mechanisms in communication and collaboration

A more flexible form of coordination is to include a social policy of floor control. Whenever a user wants to work on a shared document or file, he must initially request "the floor." If no one else is using the specified section or file at that time, then he is given the floor. That part of the document or file then becomes locked, preventing others from having access to it. If other users want access to the file, they likewise make a request for the floor. The current user is then notified and can then let the requester know how long the file will be in use. If not acceptable, the requester can try to negotiate a time for access to the file. This kind of coordination mechanism, therefore, provides more scope for negotiation between users on how to collaborate, rather than simply receiving a point-blank "permission denied" response from the system when a file is being used by someone else.

**BOX 4.3 Turning Technology Inside Out: Online versus Physical Coordination Mechanisms**

Many software applications now exist to support coordination, notably project management systems. From the project manager's perspective, they provide a flexible means of scheduling, distributing, and monitoring collaborative work and enabling them continuously to remind people of deadlines and milestones via the use of email and other kinds of representations. From the perspective of the individuals working in the company, they give them a means of letting others know when they are available for meetings and where they will be.

In practice, however, project management systems that rely exclusively on computer-mediated coordination mechanisms have not been found to be as effective as hoped. This tends to happen when the system is used to coordinate a large number of events or projects. People begin not to take notice of the numerous internal reminders and messages that are sent to them by the system, finding them to be too-intrusive, overwhelming, or annoying. This can then lead to missing important meetings and deadlines. A work-around in some organizations has been to print out the schedules and events that have been entered into the project management online database and display them as paper-based external representations (see Whittaker and Schwartz, 1995). A study that looked at the creation and use of shared external representations in collaborative work (Bellotti and Rogers, 1997) found that in several cases, information that is represented online becomes re-represented as a physical entity because the online version often gets lost, forgotten, or overlooked. This was particularly prevalent at new media companies producing web content that needed to be updated regularly. The various groups had to be coordinated across a number of parallel-running, time-critical projects.

At one site, a project coordinator would write up on a physical whiteboard every morning the main projects, schedules, and deadlines relevant for that day extracted from the online project management software. When asked why she laboriously wrote down by hand information that could be readily accessed by everyone over the computer network, she replied that, owing to the multiplication of projects and people working on them, it had become very difficult to keep track of everything that was going on. Moreover, people had become de-sensitized to the many email reminders that the software application provided, so they often forgot their significance immediately after having acknowledged them. Consequently, everyone (including herself) needed to be reminded of what was urgent and what needed dealing with that day. Placing this critical information on a physical whiteboard in a prominent public place that was clearly distinct from the continuous stream of other online information and messages provided a more effective public reminder of what was urgent and needed doing that day. In essence, the company had resorted to "turning the technology inside out."
Why are whiteboards so useful for coordinating projects? How might electronic whiteboards be designed to extend this practice?

**Comment**

Physical whiteboards are very good as coordinating tools as they display information that is external and public, making it highly visible for everyone to see. Furthermore, the information can be easily annotated to show up-to-date modifications to a schedule. Whiteboards also have a gravitational force, drawing people to them. They provide a meeting place for people to discuss and catch up with latest developments.

Electronic whiteboards have the added advantage that important information can be animated to make it stand out. Important information can also be displayed on multiple displays throughout a building and can be extracted from existing databases and software, thereby making the project coordinator's work much easier. The boards could also be used to support on-the-fly meetings in which individuals could use electronic pens to sketch out ideas that could then be stored electronically. In such settings they could also be interacted with via wireless handheld computers, allowing information to be "scraped" off or "squirted" onto the whiteboard.

### 4.2.5 Awareness mechanisms

Awareness involves knowing who is around, what is happening, and who is talking with whom (Dourish and Bly, 1992). For example, when we are at a party, we move around the physical space, observing what is going on and who is talking to whom, eavesdropping on others’ conversations and passing on gossip to others. A specific kind of awareness is peripheral awareness. This refers to a person's ability to maintain and constantly update a sense of what is going on in the physical and social context, through keeping an eye on what is happening in the periphery of their vision. This might include noting whether people are in a good or bad mood by the way they are talking, how fast the drink and food is being consumed, who has entered or left the room, how long someone has been absent, and whether the lonely guy in the corner is finally talking to someone—all while we are having a conversation with someone else. The combination of direct observations and peripheral monitoring keeps people informed and updated of what is happening in the world.

Similar ways of becoming aware and keeping aware take place in other contexts, such as a place of study or work. Importantly, this requires fathoming when is an appropriate time to interact with others to get and pass information on. Seeing a professor slam the office door signals to students that this is definitely not a good time to ask for an extension on an assignment deadline. Conversely, seeing teachers with beaming faces, chatting openly to other students suggests they are in a good mood and therefore this would be a good time to ask them if it would be all right to miss next week's seminar because of an important family engagement. The knowledge that someone is amenable or not rapidly spreads through a company, school, or other institution. People are very eager to pass on both good and bad news to others and will go out of their way to gossip, loitering in corridors, hanging around at the photocopier and coffee machine "spreading the word."
In addition to monitoring the behaviors of others, people will organize their work and physical environment to enable it to be successfully monitored by others. This ranges from the use of subtle cues to more blatant ones. An example of a subtle cue is when someone leaves their dorm or office door slightly ajar to indicate that they can be approached. A more blatant one is the complete closing of their door together with a "do not disturb" notice prominently on it, signaling to everyone that under no circumstances should they be disturbed (see Figure 4.9).

Overhearing and overseeing

People who work closely together also develop various strategies for coordinating their work, based on an up-to-date awareness of what the others are doing. This is especially so for interdependent tasks, where the outcome of one person's activity is needed for others to be able to carry out their tasks. For example, when putting on a show, the performers will constantly monitor what one another is doing in order to coordinate their performance efficiently.

The metaphorical expression "closely-knit teams" exemplifies this way of collaborating. People become highly skilled in reading and tracking what others are doing and the information they are attending to. A well-known study of this phenomenon is described by Christian Heath and Paul Luff (1992), who looked at how two controllers worked together in a control room in the London Underground. An overriding observation was that the actions of one controller were tied very closely to what the other was doing. One of the controllers was responsible for the movement of trains on the line (controller A), while the other was responsible for providing information to passengers about the current service (controller B). In many instances, it was found that controller B overheard what controller A was doing and saying, and acted accordingly—even though controller A had not said anything explicitly to him. For example, on overhearing controller A discussing a problem with a train driver over the in-cab intercom system, controller B inferred from the ensuing conversation that there was going to be a disruption to the service
and so started announcing this to the passengers on the platform before controller A had even finished talking with the train driver. At other times, the two controllers keep a lookout for each other, monitoring the environment for actions and events which they might have not noticed but may be important for them to know about so that they can act appropriately.

**Activity 4.6** What do you think happens when one person of a closely knit team does not see or hear something or misunderstands what has been said, while the others in the group assume they have seen, heard, or understood what has been said?

**Comment** In such circumstances, the person is likely to carry on as normal. In some cases this will result in inappropriate behavior. Repair mechanisms will then need to be set in motion. The knowledgeable participants may notice that the other person has not acted in the manner expected. They may then use one of a number of subtle repair mechanisms, say coughing or glancing at something that needs attending to. If this doesn't work, they may then resort to explicitly stating aloud what had previously been signaled implicitly. Conversely, the unaware participant may wonder why the event hasn't happened and, likewise, look over at the other people, cough to get their attention or explicitly ask them a question. The kind of repair mechanism employed at a given moment will depend on a number of factors, including the relationship among the participants (e.g., whether one is more senior than the others—this determines who can ask what), perceived fault or responsibility for the breakdown and the severity of the outcome of not acting there and then on the new information.

### 4.2.6 Designing collaborative technologies to support awareness

The various observations about awareness have led system developers to consider how best to provide awareness information for people who need to work together but who are not in the same physical space. Various technologies have been employed along with the design of specific applications to convey information about what people are doing and the progress of their ongoing work. As mentioned previously, audio-video links have been developed to enable remote colleagues to keep in touch with one another. Some of these systems have also been developed to provide awareness information about remote partners, allowing them to find out what one another is doing. One of the earliest systems was Portholes, developed at Xerox PARC research labs (Dourish and Bly, 1992). The system presented regularly-updated digitized video images of people in their offices from a number of different locations (in the US and UK). These were shown in a matrix display on people's workstations. Clicking on one of the images had the effect of bringing up a dialog box providing further information about that individual (e.g., name, phone number) together with a set of lightweight action buttons (e.g., email the person, listen to a pre-recorded audio snippet). The system provided changing images of people throughout the day and night in their offices, letting others see at a glance whether they were in their offices, what they were working on, and who was around (see Figure 4.10). Informal evaluation of the
set-up suggested that having access to such information led to a shared sense of community.

The emphasis in the design of these early awareness systems was largely on supporting peripheral monitoring, allowing people to see each other and their progress. Dourish and Bellotti (1992) refer to this as shared feedback. More recent distributed awareness systems provide a different kind of awareness information. Rather than place the onus on participants to find out about each other, they have been designed to allow users to notify each other about specific kinds of events. Thus, there is less emphasis on monitoring and being monitored and more on explicitly letting others know about things. Notification mechanisms are also used to provide information about the status of shared objects and the progress of collaborative tasks.

Hence, there has been a shift towards supporting a collective "stream of consciousness" that people can attend to when they want to, and likewise provide information for when they want to. An example of a distributed awareness system is Elvin, developed at the University of Queensland (Segall and Arnold, 1997), which provides a range of client services. A highly successful client is Tickertape, which is a lightweight instant messaging system, showing small color-coded messages that scroll from right to left across the screen (Fitzpatrick et al., 1999). It has been most useful as a "chat" and local organizing tool, allowing people in different locations to effortlessly send brief messages and requests to the public tickertape display (see Figure 4.11). It has been used for a range of functions, including organizing shared
events (e.g. lunch dates), making announcements, and as an "always-on" communication tool for people working together on projects but who are not physically colocated. It is also often used as a means of mediating help between people. For example, when I was visiting the University of Queensland, I asked for help over Tickertape. Within minutes, I was inundated with replies from people logged onto the system who did not even know me. At the time, I was having problems working out the key mappings between the PC that I was using in Australia and a Unix editor I couldn't find a way of quitting from on a remote machine in the UK. The suggestions that appeared on Tickertape quickly led to a discussion among the participants, and within five minutes someone had come over to my desk and sorted the problem out for me!

In addition to presenting awareness information as streaming text messages, more abstract forms of representation have been used. For example, a communication tool called Babble, developed at IBM (Erickson et al., 1999), provides a dynamic visualization of the participants in an ongoing chat-like conversation. A large 2D circle is depicted with colored marbles on each user's monitor. Marbles inside the circle convey those individuals active in the current conversation. Marbles outside the circle convey users involved in other conversations. The more active a participant is in the conversation, the more the corresponding marble is moved towards the center of the circle. Conversely, the less engaged a person is in the ongoing conversation, the more the marble moves towards the periphery of the circle (see Figure 4.12).
4.3 Ethnographic studies of collaboration and communication

One of the main approaches to informing the design of collaborative technologies that takes into account social concerns is carrying out an ethnographic study (a type of field study). Observations of the setting, be it home, work, school, public place, or other setting, are made, examining the current work and other collaborative practices people engage in. The way existing technologies and everyday artifacts are used is also analyzed. The outcome of such studies can be very illuminating, revealing how people currently manage in their work and everyday environments. They also provide a basis from which to consider how such existing settings might be improved or enhanced through the introduction of new technologies, and can also expose problematic assumptions about how collaborative technologies will or should be used in a setting (for more on how to use ethnography to inform design, see Chapter 9; how to do ethnography is covered in Chapter 12).

Many studies have analyzed in detail how people carry out their work in different settings (Plowman et al., 1995). The findings of these studies are used both to inform the design of a specific system, intended for a particular workplace, and more generally, to provide input into the design of new technologies. They can also highlight problems with existing system design methods. For example, an early study by Lucy Suchman (1983) looked at the way existing office technologies were being designed in relation to how people actually worked. She observed what really happened in a number of offices and found that there was a big mismatch between the way work was actually accomplished and the way people were supposed to work using the office technology provided. She argued that designers would be much better positioned to develop systems that could match the way people behave and use technology, if they began by considering the actual details of work practice.

In her later, much-cited study of how pairs of users interacted with an interactive help system—intended as a facility for using with a photocopier—Suchman (1987) again stressed the point that the design of interactive systems would greatly benefit from analyses that focused on the unique details of the user's particular situation—rather than being based on preconceived models of how people ought to (and will) follow instructions and procedures. Her detailed analysis of how the help system was unable to help users in many situations, highlighted the inadequacy of basing the design of an interactive system purely on an abstract user model.

Since Suchman's seminal work, a large number of ethnographic studies have examined how work gets done in a range of companies (e.g., fashion, design, multimedia, newspapers) and local government. Other settings have also recently come under scrutiny to see how technologies are used and what people do at home, in public places, in schools, and even cyberspace. Here, the objective has been to understand better the social aspects of each setting and then to come up with implications for the design of future technologies that will support and extend these. For more on the way user studies can inform future technologies, see the interview at the end of this chapter with Abigail Sellen.
4.4 Conceptual frameworks

A number of conceptual frameworks of the "social" have been adapted from other disciplines, like sociology and anthropology. As with the conceptual frameworks derived from cognitive approaches, the aim has been to provide analytic frameworks and concepts that are more amenable to design concerns. Below, we briefly describe two well known approaches, that have quite distinct origins and ways of informing interaction design. These are:

- **Language/action framework**
- **Distributed cognition**

The first describes how a model of the way people communicate was used to inform the design of a collaborative technology. The second describes a theory that is used primarily to analyze how people carry out their work, using a variety of technologies.

4.4.1 The language/action framework

The basic premise of the language/action framework is that people act through language (Winograd and Flores, 1986). It was developed to inform the design of systems to help people work more effectively through improving the way they communicate with one another. It is based on various theories of how people use language in their everyday activities, most notably speech act theory.

Speech act theory is concerned with the functions utterances have in conversations (Austin, 1962; Searle, 1969). A common function is a request that is asked indirectly (known as an indirect speech act). For example, when someone says, "It's hot in here" they may really be asking if it would be OK to open the window because they need some fresh air. Speech acts range from formalized statements (e.g., I hereby declare you man and wife) to everyday utterances (e.g., how about dinner?).

There are five categories of speech acts:

- **Assertives**—commit the speaker to something being the case
- **Commissives**—commit the speaker to some future action
- **Declarations**—pronounce something has happened
- **Directives**—get the listener to do something
- **Expressives**—express a state of affairs, such as apologizing or praising someone

Each utterance can vary in its force. For example, a command to do something has quite a different force from a polite comment about the state of affairs.

The language/action approach was developed further into a framework called conversations for action (CfA). Essentially, this framework describes the sequence of actions that can follow from a speaker making a request of someone else. It depicts a conversation as a kind of "dance" (see Figure 4.13) involving a series of steps that are seen as following the various speech acts. Different dance steps ensue depending on the speech acts followed. The most straightforward kind of dance involves progressing from state 1 through to state 5 of the conversation,
in a linear order. For example, A (state 1) may request B to do homework (state 2), B may promise to do it after she has watched a TV program (state 3), B may then report back to A that the homework is done (state 4) and A, having looked at it, declares that this is the case (state 5). In reality, conversation dances tend to be more complex. For example, A may look at the homework and see that it is very shoddy and request that B complete it properly. The conversation is thus moved back a step. B may promise to do the homework but may in fact not do it at all, thereby canceling their promise (state 7), or A may say that B doesn't need to do it any more (state 9). B may also suggest an alternative, like cooking dinner (moving to state 6).

The CfA framework was used as the basis of a conceptual model for a commercial software product called the Coordinator. The goal was to develop a system to facilitate communication in a variety of work settings, like sales, finance, general management, and planning. The Coordinator was designed to enable electronic messages to be sent between people in the form of explicit speech acts. When sending someone a request, say "Could you get the report to me?", the sender was also required to select the menu option "request." This was placed in the subject header of the message, thereby explicitly specifying the nature of the speech act. Other speech-act options included offer, promise, inform, and question (see Figure 4.14). The system also asked the user to fill in the dates by which the request should be completed. Another user receiving such a message had the option of responding with another labeled speech act. These included:

- acknowledge
- promise
- counter-offer
- decline
- free form

Figure 4.13 Conversation for action (CfA) diagram (from Winograd and Flores, 1986, p. 65).
Table A: Menu items for initiating a new conversation.

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Request</td>
<td>Sender wants receiver to do something.</td>
</tr>
<tr>
<td>Offer</td>
<td>Sender offers to do something, pending acceptance.</td>
</tr>
<tr>
<td>Promise</td>
<td>Sender promises to do something (request is implicit).</td>
</tr>
<tr>
<td>What if</td>
<td>Opens a joint exploration of a space of possibilities.</td>
</tr>
<tr>
<td>Inform</td>
<td>Sender provides information.</td>
</tr>
<tr>
<td>Question</td>
<td>A request for information.</td>
</tr>
<tr>
<td>Note</td>
<td>A simple exchange of messages (as in ordinary E-mail).</td>
</tr>
</tbody>
</table>

Figure 4.14 Menu items for initiating a conversation.

Thus, the Coordinator was designed to provide a straightforward conversational structure, allowing users to make clear the status of their work and, likewise, to be clear about the status of others' work in terms of various commitments. To reiterate, a core rationale for developing this system was to try to improve people's ability to communicate more effectively. Earlier research had shown how communication could be improved if participants were able to distinguish among the kinds of commitments people make in conversation and also the time scales for achieving them. These findings suggested to Winograd and Flores that they might achieve their goal by designing a communication system that enabled users to develop a better awareness of the value of using "speech acts." Users would do this by being explicit about their intentions in their email messages to one another.

Normally, the application of a theory backed up with empirical research is regarded as a fairly innocuous and systematic way of informing system design. However, in this instance it opened up a very large can of worms. Much of the research community at the time was incensed by the assumptions made by Winograd and Flores in applying speech act theory to the design of the Coordinator System. Many heated debates ensued, often politically charged. A major concern was the extent to which the system prescribed how people should communicate. It was pointed out that asking users to specify explicitly the nature of their implicit speech acts was contrary to what they normally do in conversations. Forcing people to communicate in such an artificial way was regarded as highly undesirable. While some people may be very blatant about what they want doing, when they want it done by, and what they are prepared to do, most people tend to use more subtle and indirect forms of communication to advance their collaborations with others. The problem that Winograd and Flores came up against was people's resistance to radically change their way of communicating.

Indeed, many of the people who tried using the Coordinator System in their work organizations either abandoned it or resorted to using only the free-form message facility, which had no explicit demands associated with it. In these con-
4.4 Conceptual frameworks

In the previous chapter we described how traditional approaches to modeling cognition have focused on what goes on inside one person's head. We also mentioned that there has been considerable dissatisfaction with this approach, as it ignores how people interact with one another and their use of artifacts and external representations in their everyday and working activities. To redress this situation, Ed Hutchins and his colleagues developed the distributed cognition approach as a new paradigm for conceptualizing human work activities (e.g., Hutchins, 1995) (see Figure 4.15).

The distributed cognition approach describes what happens in a cognitive system. Typically, this involves explaining the interactions among people, the artifacts...
they use, and the environment they are working in. An example of a cognitive system is an airline cockpit, where a top-level goal is to fly the plane. This involves:

- the pilot, co-pilot and air traffic controller interacting with one another
- the pilot and co-pilot interacting with the instruments in the cockpit
- the pilot and co-pilot interacting with the environment in which the plane is flying (e.g., sky, runway).

A primary objective of the distributed cognition approach is to describe these interactions in terms of how information is propagated through different media. By this is meant how information is represented and re-represented as it moves across individuals and through the array of artifacts that are used (e.g., maps, instrument readings, scribbles, spoken word) during activities. These transformations of information are referred to as changes in representational state.

This way of describing and analyzing a cognitive activity contrasts with other cognitive approaches (e.g., the information processing model) in that it focuses not on what is happening inside the heads of each individual but on what is happening across individuals and artifacts. For example, in the cognitive system of the cockpit, a number of people and artifacts are involved in the activity of "flying to a higher altitude." The air traffic controller initially tells the co-pilot when it is safe to fly to a higher altitude. The co-pilot then alerts the pilot, who is flying the plane, by moving a knob on the instrument panel in front of them, indicating that it is now safe to fly (see Figure 4.16). Hence, the information concerning this activity is transformed
through different media (over the radio, through the co-pilot, and via a change in the position of an instrument).

A distributed cognition analysis typically involves examining:

- the distributed problem solving that takes place (including the way people work together to solve a problem)
- the role of verbal and non-verbal behavior (including what is said, what is implied by glances, winks, etc., and what is not said)
- the various coordinating mechanisms that are used (e.g., rules, procedures)
- the various communicative pathways that take place as a collaborative activity progresses
- how knowledge is shared and accessed

In addition, an important part of a distributed cognition analysis is to identify the problems, breakdowns, and concomitant problem-solving processes that emerge to deal with them. The analysis can be used to predict what would happen to the way information is propagated through a cognitive system, using a different arrangement of technologies and artifacts and what the consequences of this would be for the current work setting. This is especially useful when designing and evaluating new collaborative technologies.

**DILEMMA Who Should Have Control?**

A core design dilemma facing those involved in developing collaborative technologies is how much control to implement and how much to leave to the users to configure for themselves. Should coordinating mechanisms like rules, procedures, and conventions be designed as part of the system architecture, or should the system be designed to be more free and open, allowing all users to do the same things? For example, when designing a file-sharing system is it more desirable to allow all users free access to all files, or is it preferable to implement some kind of social protocol that allows different user privileges and permissions? Likewise, when designing shared applications such as shared workspaces and collaborative authoring tools, to what extent should system-mediated control mechanisms be implemented that prescribe (and make quite clear) to users how they should share and collaborate? What happens when it is left up to the users to decide upon their own social protocols on how they should coordinate and collaborate with one another? Does it result in anarchy, or do they succeed in creating a shared environment that supports a harmonious way of working?

When working through these design concerns, it is important to consider what happens if too much or too little control is implemented in the collaborative technology. If there is too much "social engineering," then it could result in the users refusing to use it in the way intended. For example, the Coordinator system was found to be unusable in a number of organizations because it required people to radically change their way of communicating in ways considered unacceptable. Similarly, many of the conventions implemented in the POLITeam workspace system (e.g., always typing in the code of a file when using it) were not followed because they required the users to do extra work that was seen by them as being tedious and unnecessary.

Conversely, if not enough consideration is given to the way control is handled, the resulting system could end up being unusable and unacceptable. For example, some of the early shared calendar systems that had a free-for-all policy (anyone could look at anyone else's calendar and arrange a meeting in a free slot) were found to be too much of an invasion of people's privacy.
There are several other well known conceptual frameworks that are used to analyze how people collaborate and communicate, including activity theory, ethnomethodology, situated action and common ground theory.

Assignment

The aim of this design activity is for you to analyze the design of a collaborative virtual environment (CVE) with respect to how it is designed to support collaboration and communication.

Visit an existing CVE (many are freely downloadable) such as V-Chat (vchat.microsoft.com), one of the many Worlds Away environments (www.worlds.net), or the Palace (www.communities.com). Try to work out how they have been designed to take into account the following:

(a) *General social issues*

What is the purpose of the CVE?
What kinds of conversation mechanisms are supported?
What kinds of coordination mechanisms are provided?
What kinds of social protocols and conventions are used?
What kinds of awareness information is provided?
Does the mode of communication and interaction seem natural or awkward?

(b) *Specific interaction design issues*

What form of interaction and communication is supported (e.g., text, audio, video)?
What other visualizations are included? What information do they convey?
How do users switch between different modes of interaction (e.g., exploring and chatting)? Is the switch seamless?
Are there any social phenomena that occur specific to the context of the CVE that wouldn't happen in face to face settings (e.g., flaming)?

(c) *Design issues*

What other features might you include in the CVE to improve communication and collaboration?
Summary

In this chapter we have looked at some core aspects of sociality, namely communication and collaboration. We examined the main social mechanisms that people use in different settings in order to collaborate. A number of collaborative technologies have been designed to support and extend these mechanisms. We looked at representative examples of these, highlighting core interaction design concerns. A particular concern is social acceptability that is critical for the success or failure of technologies intended to be used by groups of people working or communicating together. We also discussed how ethnographic studies and theoretical frameworks can play a valuable role when designing new technologies for work and other settings.

Key points

- Social aspects are the actions and interactions that people engage in at home, work, school, and in public.
- The three main kinds of social mechanism used to coordinate and facilitate social aspects are conversation, coordination, and awareness.
- Talk and the way it is managed is integral to coordinating social activities.
- Many kinds of computer-mediated communication systems have been developed to enable people to communicate with one another when in physically different locations.
- External representations, rules, conventions, verbal and non-verbal communication are all used to coordinate activities among people.
- It is important to take into account the social protocols people use in face to face collaboration when designing collaborative technologies.
- Keeping aware of what others are doing and letting others know what you are doing are important aspects of collaborative working and socializing.
- Ethnographic studies and conceptual frameworks play an important role in understanding the social issues to be taken into account in designing collaborative systems.
- Getting the right level of control between users and system is critical when designing collaborative systems.

Further reading


BAECKER, R. M. (ed.) (1993) Readings in Groupware and Computer-Supported Cooperative Work: Assisting Human-Human Collaboration, San Mateo, Ca.: Morgan Kaufmann. These two collections of readings include a number of representative papers from the field of CSCW, ranging from social to system architecture issues.

Abigail Sellen is a senior researcher at Hewlett Packard Labs in Bristol, UK. Her work involves carrying out user studies to inform the development of future products, including appliances and web-based services. She has a background in cognitive science and human factors engineering, having obtained her doctorate at the University of California, San Diego. Prior to this Abigail worked at Xerox Research Labs in Cambridge, UK, and Apple Computer Inc. She has also worked as an academic researcher at the Computer Systems Research Institute at the University of Toronto, Canada and the Applied Psychology Unit in Cambridge, UK. She has written widely on the social and cognitive aspects of paper use, video conferencing, input devices, human memory, and human error, all with an eye to the design of new technologies.

YR: Could you tell me what you do at Hewlett Packard Research Labs?

AS: Sure, I've been at HP Labs for a number of years now as a member of its User Studies and Design Group. This is a smallish group consisting of five social scientists and three designers. Our work can best be described as doing three things: we do projects that are group-led around particular themes, like for example, how people use digital music or how people capture documents using scanning technology. We do consulting work for development teams at HP, and thirdly, we do a little bit of our own individual work, like writing papers and books, and giving talks.

YR: Right. Could you tell me about user studies, what they are and why you consider them important?

AS: OK. User studies essentially involve looking at how people behave either in their natural habitats or in the laboratory, both with old technologies and with new ones. I think there are many different questions that these kinds of studies can help you answer. Let me name a few. One question is: who is going to be the potential user for a particular device or service that you are thinking of developing? A second question—which I think is key—is, what is the potential value of a particular product for a user? Once we know this, we can then ask, for a particular situation or task, what features do we want to deliver and how best should we deliver those features? This includes, for example, what would the interface look like? Finally, I think user studies are important to understand how users' lives may change and how they will be affected by introducing a new technology. This has to take into account the social, physical, and technological context into which it will be introduced.

YR: So it sounds like you have a set of general questions you have in mind when you do a user study. Could you now describe how you would do a user study and what kinds of things you would be looking for?

AS: Well, I think there are two different classes of user studies and both are quite different in the ways you go about them. There are evaluation studies, where we take a concept, a prototype or even a developed technology and look at how it is used and then try to modify or improve it based on what we find. The second class of user studies is more about discovering what people's unmet needs may be. This means trying to develop new concepts and ideas for things that people may never have thought of before. This is difficult because you can't necessarily just ask people what they would like or what they would use. Instead, you have to make inferences from studying people in different situations and try to understand from this what they might need or value.

YR: In the book we mention the importance of taking into account social aspects, such as awareness of others, how people communicate with each other and so on. Do you think these issues are important when you are doing these two kinds of user studies?

AS: Well, yes, and in particular I think social aspects really are playing to that second class of user study I mentioned where you are trying to discover what people's unmet needs or requirements may be. Here you are trying to get rich descriptions about what people do in the context of their everyday lives—whether this is in their working lives, their home lives, or lives on the move. I'd say getting the social aspects understood is often very important in trying to understand what value new products and services might
bring to people's day-to-day activities, and also how they would fit into those existing activities.

YR: And what about cognitive aspects, such as how people carry out their tasks, what they remember, what they are bad at remembering? Is that also important to look into when you are doing these kinds of studies?

AS: Yes, if you think about evaluation studies, then cognitive aspects are extremely important. Looking at cognitive aspects can help you understand the nature of the user interaction, in particular what processes are going on in their heads. This includes issues like learning how users perceive a device and how they form a mental model of how something works. Cognitive issues are especially important to consider when we want to contrast one device with another or think about new and better ways in which we might design an interface.

YR: I wonder if you could describe to me briefly one of your recent studies where you have looked at cognitive and social aspects.

AS: How about a recent study we did to do with building devices for reading digital documents? When we first set out on this study, before we could begin to think about how to build such devices, we had to begin by asking, "What do we mean by reading?" It turned out there was not a lot written about the different ways people read in their day-to-day lives. So the first thing we did was a very broad study looking at how people read in work situations. The technique we used here was a combination of asking people to fill out a diary about their reading activities during the course of a day and interviewing them at the end of each day. The interviews were based around what was written in the diaries, which turned out to be a good way of unpacking more details about what people had been doing.

That initial study allowed us to categorize all the different ways people were reading. What we found out is that actually you can't talk about reading in a generic sense but that it falls into at least 10 different categories. For example, sometimes people skim read, sometimes they read for the purpose of writing something, and sometimes they read very reflectively and deeply, marking up their documents as they go. What quickly emerged from this first study was that if you're designing a device for reading it might look very different depending on the kind of reading the users are doing. So, for example, if they're reading by themselves, the screen size and viewing angle may not be as important as if they're reading with others. If they're skim reading, the ability to quickly flick through pages is important. And if they're reading and writing, then this points to the need for a pen-based interface. All of these issues become important design considerations.

This study then led to the development of some design concepts and ideas for new kinds of reading devices. At this stage we involved designers to develop different "props" to get feedback and reactions from potential users. A prop could be anything from a quick sketch to an animation to a styrofoam 3D mockup. Once you have this initial design work, you can then begin to develop working prototypes and test them with realistic tasks in both laboratory and natural settings. Some of this work we have already completed, but the project has had an impact on several different research and development efforts.

YR: Would you say that user studies are going to become an increasingly important part of the interaction design process, especially as new technologies like ubiquitous computing and handheld devices come into being—and where no one really knows what applications to develop?

AS: Yes. I think the main contribution of user studies, say, 15 years ago was in the area of evaluation and usability testing. I think that role is changing now in that user studies researchers are not only those who evaluate devices and interfaces but also those who develop new concepts. Also, another important development is a change in the way the research is carried out. More and more I am finding that teams are drawing together people from other disciplines, such as sociologists, marketing people, designers, and people from business and technology development.

YR: So they are essentially working as a multidisciplinary team. Finally, what is it like to work in a large organization like HP, with so many different departments?

AS: One thing about working for a large organization is that you get a lot of variety in what you can do. You can pick and choose to some extent and, depending on the organization, don't have to be tied to a particular product. If, on the other hand, you work
for a smaller organization such as a start-up company, inevitably there is lots of pressure to get things out the door quickly. Things are often very focused. Whether large or small, however, I think one of the hardest things I have found in working for corporate research is learning to work with the development teams. They put huge pressures on you because they have huge pressures on them. You really have to work at effectively incorporating user studies findings into the development process. This can be incredibly challenging, but it's also satisfying to have an impact on real products.
5.1 Introduction

An overarching goal of interaction design is to develop interactive systems that elicit positive responses from users, such as feeling at ease, being comfortable, and enjoying the experience of using them. More recently, designers have become interested in how to design interactive products that elicit specific kinds of emotional responses in users, motivating them to learn, play, be creative, and be social. There is also a growing concern with how to design websites that people can trust, that make them feel comfortable about divulging personal information or making a purchase.

We refer to this newly emerging area of interaction design as affective aspects. In this chapter we look at how and why the design of computer systems cause certain kinds of emotional responses in users. We begin by looking in general at expressive interfaces, examining the role of an interface's appearance on users and how it affects usability. We then examine how computer systems elicit negative responses, e.g., user frustration. Following this, we present a debate on the controversial topic of anthropomorphism and its implications for designing applications to have human-like qualities. Finally, we examine the range of virtual characters designed to motivate people to learn, buy, listen, etc., and consider how useful and appropriate they are.
The main aims of this chapter are to:

- Explain what expressive interfaces are and the affects they can have on people.
- Outline the problems of user frustration and how to reduce them.
- Debate the pros and cons of applying anthropomorphism in interaction design.
- Assess the believability of different kinds of agents and virtual characters.
- Enable you to critique the persuasive impact of e-commerce agents on customers.

### 5.2 What are affective aspects?

In general, the term "affective" refers to producing an emotional response. For example, when people are happy they smile. Affective behavior can also cause an emotional response in others. So, for example, when someone smiles it can cause others to feel good and smile back. Emotional skills, especially the ability to express and recognize emotions, are central to human communication. Most of us are highly skilled at detecting when someone is angry, happy, sad, or bored by recognizing their facial expressions, way of speaking, and other body signals. We are also very good at knowing what emotions to express in given situations. For example, when someone has just heard they have failed an exam we know it is not a good time to smile and be happy. Instead we try to empathize.

It has been suggested that computers be designed to recognize and express emotions in the same way humans do (Picard, 1998). The term coined for this approach is "affective computing". A long-standing area of research in artificial intelligence and artificial life has been to create intelligent robots and other computer-based systems that behave like humans and other creatures. One well-known project is MIT’s COG, in which a number of researchers are attempting to build an artificial two-year-old. One of the offsprings of COG is Kismet (Breazeal, 1999), which has been designed to engage in meaningful social interactions with humans (see Figure 5.1). Our concern in this chapter takes a different approach: how can interactive systems be designed (both deliberately and inadvertently) to make people respond in certain ways?

Figure 5.1 Kismet the robot expressing surprise, anger, and happiness.
5.3 Expressive interfaces

A well-known approach to designing affective interfaces is to use expressive icons and other graphical elements to convey emotional states. These are typically used to indicate the current state of a computer. For example, a hallmark of the Apple computer is the icon of a smiling Mac that appears on the screen when the machine is first started (see Figure 5.2(a)). The smiling icon conveys a sense of friendliness, inviting the user to feel at ease and even smile back. The appearance of the icon on the screen can also be very reassuring to users, indicating that their computer is working fine. This is especially useful when they have just rebooted the computer after it has crashed and where previous attempts to reboot have failed (usually indicated by a sad icon face—see Figure 5.2(b)). Other ways of conveying the status of a system are through the use of:

- dynamic icons, e.g., a recycle bin expanding when a file is placed into it
- animations, e.g., a bee flying across the screen indicating that the computer is doing something, like checking files
- spoken messages, using various kinds of voices, telling the user what needs to be done
- various sounds indicating actions and events (e.g., window closing, files being dragged, new email arriving)

One of the benefits of these kinds of expressive embellishments is that they provide reassuring feedback to the user that can be both informative and fun.

The style of an interface, in terms of the shapes, fonts, colors, and graphical elements that are used and the way they are combined, influences how pleasurable it is to interact with. The more effective the use of imagery at the interface, the more engaging and enjoyable it can be (Mullet and Sano, 1995). Conversely, if little thought is given to the appearance of an interface, it can turn out looking like a dog’s dinner. Until recently, HCI has focused primarily on getting the usability right, with little attention being paid to how to design aesthetically pleasing interfaces. Interestingly, recent research suggests that the aesthetics of an interface can

Figure 5.2 (a) Smiling and (b) sad Apple Macs.
have a positive effect on people's perception of the system's usability (Tractinsky, 1997). Moreover, when the "look and feel" of an interface is pleasing (e.g., beautiful graphics, nice feel to the way the elements have been put together, well-designed fonts, elegant use of images and color) users are likely to be more tolerant of its usability (e.g., they may be prepared to wait a few more seconds for a website to download). As we have argued before, interaction design should not just be about usability per se, but should also include aesthetic design, such as how pleasurable an interface is to look at (or listen to). The key is to get the right balance between usability and other design concerns, like aesthetics (See Figure 5.3 on Color Plate 6).

**ACTIVITY 5.1**

A question of style or stereotype? Figure 5.4 shows two differently designed dialog boxes. Describe how they differ in terms of style. Of the two, which one do you prefer? Why? Which one do you think (i) Europeans would like the most and (ii) Americans would like the most?

**Comment**

Aaron Marcus, a graphic designer, created the two designs in an attempt to provide appealing interfaces. Dialog box A was designed for white American females while dialog box B was designed for European adult male intellectuals. The rationale behind Marcus's ideas was that European adult male intellectuals like "suave prose, a restrained treatment of information density, and a classical approach to font selection (e.g., the use of serif type in axial symmetric layouts similar to those found in elegant bronze European building identification signs)." In contrast, white American females "prefer a more detailed presentation, curvilinear shapes and the absence of some of the more-brutal terms... favored by male software engineers."

When the different interfaces were empirically tested by Teasley et al. (1994), their results did not concur with Marcus's assumptions. In particular, they found that the European dialog box was liked the best by all people and was considered most appropriate for all users. Moreover, the round dialog box designed for women was strongly disliked by everyone. The assumption that women like curvilinear features clearly was not true in this context. At the very least, displaying the font labels in a circular plane makes them more difficult to read than when presented in the conventionally accepted horizontal plane.

Another popular kind of expressive interface is the friendly interface agent. A general assumption is that novices will feel more at ease with this kind of "companion" and will be encouraged to try things out, after listening, watching, following, and interacting with them. For example, Microsoft pioneered a new class of agent-based software, called Bob, aimed at new computer users (many of whom were seen as computer-phobic). The agents were presented as friendly characters, including a friendly dog and a cute bunny. An underlying assumption was that having these kinds of agents on the screen would make the users feel more comfortable and at ease with using the software. An interface metaphor of a warm, cozy living room, replete with fire, furnishings, and furniture was provided (see Figure 5.5)—again intended to convey a comfortable feeling.

Since the creation of Bob, Microsoft has developed other kinds of agents, including the infamous "Clippy" (a paper clip that has human-like qualities), as part
Figure 5.4 Square and round dialog boxes designed by Aaron Marcus (1993): (a) dialog box designed for white American women, and (b) dialog box designed for European adult male intellectuals.
of their Windows '98 operating environment.' The agents typically appear at the bottom of the screen whenever the system "thinks" the user needs help carrying out a particular task. They, too, are depicted as cartoon characters, with friendly warm personalities. As mentioned in Chapter 2, one of the problems of using agents in this more general context is that some users do not like them. More experienced users who have developed a reasonably good mental model of the system often find such agent helpers very trying and quickly find them annoying intrusions, especially when they distract them from their work. (We return to anthropomorphism and the design of interface agents later in Section 5.5).

Users themselves have also been inventive in expressing their emotions at the computer interface. One well-known method is the use of emoticons. These are keyboard symbols that are combined in various ways to convey feelings and emotions by simulating facial expressions like smiling, winking, and frowning on the screen. The meaning of an emoticon depends on the content of the message and where it is placed in the message. For example, a smiley face placed at the end of a message can mean that the sender is happy about a piece of news she has just written about. Alternatively, if it is placed at the end of a comment in the body of the message, it usually indicates that this comment is not intended to be taken seriously. Most emoticons are designed to be interpreted with the viewer's head tilted over to the left (a result of the way the symbols are represented on the screen). Some of the best known ones are presented in Table 5.1.

A recently created shorthand language, used primarily by teenagers when online chatting or texting is the use of abbreviated words. These are formed by keying in various numbers and let-

---

1On the Mac version of Microsoft's Office 2001, Clippy was replaced by an anthropomorphized Mac computer with big feet and a hand that conveys various gestures and moods.
Table 5.1  Some commonly used emoticons.

<table>
<thead>
<tr>
<th>Emotion</th>
<th>Expression</th>
<th>Emoticon</th>
<th>Possible meanings</th>
</tr>
</thead>
<tbody>
<tr>
<td>Happy</td>
<td>Smile</td>
<td>:) or :D</td>
<td>(i) Happiness, or (ii) previous comment not to be taken seriously</td>
</tr>
<tr>
<td>Sad</td>
<td>Mouth down</td>
<td>:( or :&lt;-</td>
<td>Disappointed, unhappy</td>
</tr>
<tr>
<td>Cheeky</td>
<td>Wink</td>
<td>;) or ;-)</td>
<td>Previous comment meant as tongue-in-cheek</td>
</tr>
<tr>
<td>Mad</td>
<td>Brows raised</td>
<td>&gt;:</td>
<td>Mad about something</td>
</tr>
<tr>
<td>Very angry</td>
<td>Angry face</td>
<td>&gt;:-(</td>
<td>Very angry, cross</td>
</tr>
<tr>
<td>Embarrassed</td>
<td>Mouth open</td>
<td>:O</td>
<td>Embarrassed, shocked</td>
</tr>
<tr>
<td>Sick</td>
<td>Looking sick</td>
<td>:x</td>
<td>Feeling ill</td>
</tr>
<tr>
<td>Naïve</td>
<td>Schoolboyish look</td>
<td>&lt;:-)</td>
<td>Smiley wearing a dunce's cap to convey that the sender is about to ask a stupid question.</td>
</tr>
</tbody>
</table>

ters in place of words, e.g., "I 1 2 CU 2nite". As well as being creative, the shorthand can convey emotional connotations.

Expressive forms like emoticons, sounds, icons, and interface agents have been primarily used to (i) convey emotional states and/or (ii) elicit certain kinds of emotional responses in users, such as feeling at ease, comfort, and happiness. However, in many situations computer interfaces inadvertently elicit negative emotional responses. By far the most common is user frustration, to which we now turn our attention.

5.4  User frustration

Everyone at some time or other gets frustrated when using a computer. The effect ranges from feeling mildly amused to extremely angry. There are myriads of reasons why such emotional responses occur:

- when an application doesn't work properly or crashes
- when a system doesn't do what the user wants it to do
- when a user's expectations are not met
- when a system does not provide sufficient information to let the user know what to do
- when error messages pop up that are vague, obtuse, or condemning
- when the appearance of an interface is too noisy, garish, gimmicky, or patronizing
- when a system requires users to carry out many steps to perform a task, only to discover a mistake was made somewhere along the line and they need to start all over again
Provide specific examples for each of the above categories from your own experience, when you have become frustrated with an interactive device (e.g., telephone, VCR, vending machine, PDA, computer). In doing this, write down any further types of frustration that come to mind. Then prioritize them in terms of how annoying they are. What are the worst types?

Comment

In the text below we provide examples of common frustrations experienced when using computer systems. The worst include unhelpful error messages and excessive housekeeping tasks. You no doubt came up with many more.

Often user frustration is caused by bad design, no design, inadvertent design, or ill-thought-out design. It is rarely caused deliberately. However, its impact on users can be quite drastic and make them abandon the application or tool. Here, we present some examples of classic user-frustration provokers that could be avoided or reduced by putting more thought into the design of the conceptual model.

1. Gimmicks

Cause: When a users' expectations are not met and they are instead presented with a gimmicky display.

Level of frustration: Mild

This can happen when clicking on a link to a website only to discover that it is still "under construction." It can be still more annoying when the website displays a road-sign icon of "men at work" (see Figure 5.6). Although the website owner may think such signs amusing, it serves to underscore the viewer's frustration at having made the effort to go to the website only to be told that it is incomplete (or not even started in some cases). Clicking on links that don't work is also frustrating.

How to avoid or help reduce the frustration:

By far the best strategy is to avoid using gimmicks to cover up the real crime. In this example it is much better to put material live on the web only when it is complete and working properly. People very rarely return to sites when they see icons like the one in Figure 5.6.

2. Error Messages

Cause: When a system or application crashes and provides an "unexpected" error message.

Level of frustration: High

Error messages have a long history in computer interface design, and are notorious for their incomprehensibility. For example, Nielsen (1993) describes an early system that was developed that allowed only for one line of error messages. Whenever the

Figure 5.6 Men at work icon sign indicating "website under construction." According to AltaVista, there were over 12 million websites containing the phrase "under construction" in January 2001.
5.4 User frustration

error message was too long, the system truncated it to fit on the line, which the users would spend ages trying to decipher. The full message was available only by pressing the PF1 (help key) function key. While this may have seemed like a natural design solution to the developers, it was not at all obvious to the users. A much better design solution would have been to use the one line of the screen to indicate how to find more information about the current error ("press the PF1 key for explanation").

The use of cryptic language and developer's jargon in error messages is a major contributing factor in user frustration. It is one thing to have to cope when something goes wrong but it is another to have to try to understand an obscure message that pops up by way of explanation. One of my favorites, which sometimes appears on the screen when I'm trying to do something perfectly reasonable like paste some text into a document, using a word processor, is: "The application Word Wonder has unexpectedly quit due to a Type 2 error."

It is very clear from what the system has just done (closed the application very rapidly) that it has just crashed, so such feedback is not very helpful. Letting the user know that the error is of a Type 2 kind is also not very useful. How is the average user meant to understand this? Is there a list of error types ready at hand to tell the user how to solve the problem for each error? Moreover, such a reference invites the user to worry about how many more error types there might be. The tone of the message is also annoying. The adjective "unexpectedly" seems condescending, implying almost that it is the fault of the user rather than the computer. Why include such a word at all? After all, how else could the application have quit? One could never imagine the opposite situation: an error message pops up saying, "The application has expectedly quit, due to poor coding in the operating system."

**How to avoid or help reduce the frustration:**

Ideally, error messages should be treated as how-to-fix-it messages. Instead of explicating what has happened, they should state the cause of the problem and what the user needs to do to fix it. Shneiderman (1998) has developed a detailed set of guidelines on how to develop helpful messages that are easy to read and understand. Box 5.1 summarizes the main recommendations.

**BOX 5.1 Main Guidelines on How to Design Good Error Messages (Adapted from Shneiderman, 1998)**

- Rather than condemn users, messages should be courteous, indicating what users need to do to set things right.
- Avoid using terms like FATAL, ERROR, INVALID, BAD, and ILLEGAL.
- Avoid long code numbers and uppercase letters.
- Audio warnings should be under the user's control, since they can cause much embarrassment.
- Messages should be precise rather than vague.
- Messages should provide a help icon or command to allow users to get context-sensitive help.
- Messages should be provided at multiple levels, so that short messages can be supplemented with longer explanations.
Below are some common error messages expressed in harsh computer jargon that can be quite threatening and offensive. Rewrite them in more usable, useful, and friendly language that would help users to understand the cause of the problem and how to fix it. For each message, imagine a specific context where such a problem might occur.

SYNTAX ERROR
INVALID FILENAME
INVALID DATA
APPLICATION ZETA HAS UNEXPECTEDLY QUIT DUE TO A TYPE 4 ERROR
DRIVE ERROR: ABORT, RETRY OR FAIL?

Comment

How specific the given advice can be will depend on the kind of system it is. Here are suggestions for hypothetical systems.

SYNTAX ERROR—There is a problem with the way you have typed the command. Check for typos.
INVALID FILENAME—Choose another file name that uses only 20 characters or less and is lower case without any spaces.
INVALID DATA—There is a problem with the data you have entered. Try again, checking that no decimal points are used.
APPLICATION ZETA HAS UNEXPECTEDLY QUIT DUE TO A TYPE 4 ERROR—The application you were working on crashed because of an internal memory problem. Try rebooting and increasing the amount of allocated memory to the application.
DRIVE ERROR: ABORT, RETRY OR FAIL?—There is a problem with reading your disk. Try inserting it again.

3. **Overburdening the user**

*Cause:* Upgrading software so that users are required to carry out excessive housekeeping tasks

*Level of frustration:* Medium to high

Another pervasive frustrating user experience is upgrading a piece of software. It is now common for users to have to go through this housekeeping task on a regular basis, especially if they run a number of applications. More often than not it tends to be a real chore, being very time-consuming and requiring the user to do a whole range of things, like resetting preferences, sorting out extensions, checking other configurations, and learning new ways of doing things. Often, problems can develop that are not detected till some time later, when a user tries an operation that worked fine before but mysteriously now fails. A common problem is that settings get lost or do not copy over properly during the upgrade. As the number of options for customizing an application or operating system increases for each new upgrade, so, too, does the headache of having to reset all the relevant preferences. Wading through myriads of dialog boxes and menus and figuring out which checkbox to
Figure 5.7a Typical message in dialog box that appears when trying to run an applet on a website that needs a plug-in the user does not have.

click on, can be a very arduous task. To add to the frustration, users may also discover that several of their well-learned procedures for carrying out tasks have been substantially changed in the upgrade.

A pet frustration of mine over the years has been trying to run various websites that require me to install a new plug-in. Achieving this is never straightforward. I have spent huge amounts of time trying to install what I assume to be the correct plug-in—only to discover that it is not yet available or incompatible with the operating system or machine I am using.

What typically happens is I’ll visit a tempting new website, only to discover that my browser is not suitably equipped to view it. When my browser fails to run the applet, a helpful dialog box will pop up saying that a plug-in of X type is required. It also usually directs me to another website from where the plug-in can be downloaded (see Figure 5.7a). Websites that offer such plug-ins, however, are not organized around my specific needs but are designed more like hardware stores (a bad conceptual model), offering hundreds (maybe even thousands) of plug-ins covering all manner of applications and systems. Getting the right kind of plug-in from the vast array available requires knowing a number of things about your machine and the kind of network you are using. In going through the various options

WEB PLUG-IN DIRECTORY
Here is where you find the links to all of the plug-ins available on the net. Simply find a plug-in you’re interested in, view what platforms it currently (or will ‘soon’) support and click on its link. If you know of a plug-in not listed on this page please take a moment and tell us about it with our all new reporting system!

Plug-ins by Category
The Full List This is the whole list, but I gotta warn ya its getting big
MultiMedia Multi-Media Plug-Ins, AVI, QuickTime, ShockWave...
Graphics Graphic Plug-Ins, PNG, CMX, DWG...
Sound Sound & MIDI Plug-Ins, MIDI, ReadAudio, TrueSpeech...
Document Document Viewer Plug-Ins, Acrobat, Envoy, MS Word...
Productivity Productivity Plug-Ins, Map Viewers, Spell Checkers...
VRML/3-D VRML & QD3D Plug-Ins

Plug-ins by platform
Macintosh Macintosh Plug-Ins
OS/2 IBM OS/2 Plug-Ins
Unix Unix Plug-Ins
Windows Windows Plug-Ins

Figure 5.7b Directory of plug-ins available on a plug-in site directed to from Netscape.
to narrow down which plug-in is required, it is easy to overlook something and end up with an inappropriate plug-in. Even when the right plug-in has been downloaded and placed in the appropriate system folder, it may not work. A number of other things usually need to be done, like specifying mime-type and suffix. The whole process can end up taking huge amounts of time, rather than the couple of minutes most users would assume.

How to avoid or help reduce the frustration:

Users should not have to spend large amounts of time on housekeeping tasks. Upgrading should be an effortless and largely automatic process. Designers need to think carefully about the trade-offs incurred when introducing upgrades, especially the amount of relearning required. Plug-ins that users have to search for, download, and set up themselves should be phased out and replaced with more powerful browsers that automatically download the right plug-ins and place them in the appropriate desktop folder reliably, or, better still, interpret the different file types themselves.

4. Appearance

Cause: When the appearance of an interface is unpleasant

Level of frustration: Medium

As mentioned earlier, the appearance of an interface can affect its usability. Users get annoyed by:

- websites that are overloaded with text and graphics, making it difficult to find the information desired and slow to access
- flashing animations, especially banner ads, which are very distracting
- the copious use of sound effects and Muzak, especially when selecting options, carrying out actions, starting up CD-ROMS, running tutorials, or watching website demos
- featuritis—an excessive number of operations, represented at the interface as banks of icons or cascading menus
- childish designs that keep popping up on the screen, such as certain kinds of helper agents
- poorly laid out keyboards, pads, control panels, and other input devices that cause the user to press the wrong keys or buttons when trying to do something else

How to avoid or help reduce the frustration:

Interfaces should be designed to be simple, perceptually salient, and elegant and to adhere to usability design principles, well-thought-out graphic design principles, and ergonomic guidelines (e.g. Mullet and Sano, 1996).

5.3.1 Dealing with user frustration

One way of coping with computer-induced frustration is to vent and take it out on the computer or other users. As mentioned in Chapter 3, a typical response to seeing the cursor freeze on the screen is repeatedly to bash every key on the keyboard.
Another way of venting anger is through flaming. When upset or annoyed by a piece of news or something in an email message, people may overreact and respond by writing things in email that they wouldn't dream of saying face to face. They often use keyboard symbols to emphasize their anger or frustration, e.g., exclamation marks (!!!), capital letters (WHY DID YOU DO THAT?) and repeated question marks (??????) that can be quite offensive to those on the receiving end. While such venting behavior can make the user feel temporarily less frustrated, it can be very unproductive and can annoy the recipients. Anyone who has received a flame knows just how unpleasant it is.

In the previous section, we provided some suggestions on how systems could be improved to help reduce commonly caused frustrations. Many of the ideas discussed throughout the book are also concerned with designing technologies and interfaces that are usable, useful, and enjoyable. There will always be situations, however, in which systems do not function in the way users expect them to, or in which the user misunderstands something and makes a mistake. In these circumstances, error messages (phrased as "how-to-fix-it"advice) should be provided that explain what the user needs to do.

Another way of providing information is through online help, such as tips, handy hints, and contextualized advice. Like error messages, these need to be designed to guide users on what to do next when they get stuck and it is not obvious from the interface what to do. The signaling used at the interface to indicate that such online help is available needs careful consideration. A cartoon-based agent with a catchy tune may seem friendly and helpful the first time round but can quickly become annoying. A help icon or command that is activated by the users themselves when they want help is often preferable.

**BOX 5.2 Should Computers Say They're Sorry?**

A provocative idea is that computers should apologize when they make a mistake. Reeves and Naas (1996), for example, argue that they should be polite and courteous in the same way as people are to one another. While apologizing is normal social etiquette in human behavior, especially when someone makes a mistake, would you agree that computers should be made to behave in the same way? Would users be as forgiving of computers as they are of one another? For example, what would most users think if, after a system had crashed, it came up with a spoken or written apology such as, "I’m really sorry I crashed. I’ll try not to do it again"? Would they think that the computer was being sincere? Would the apology make them forgive the computer in the way they forgive other people, after receiving such an apology? Or would it have no effect at all? Worse still, would users perceive such messages as vacuous statements and regard them simply as condescending, thereby increasing their level of frustration? How else might systems communicate with users when they have committed an error?

**5.5 A debate: the application of anthropomorphism to interaction design**

In this section we present a debate. Read through the arguments for and against the motion and then the evidence provided. Afterwards decide for yourself whether you support the motion.
The motion

The use of anthropomorphism in interaction design is an effective technique and should be exploited further.

Background

A controversial debate in interaction design is whether to exploit the phenomenon of anthropomorphism (the propensity people have to attribute human qualities to objects). It is something that people do naturally in their everyday lives and is commonly exploited in the design of technologies (e.g., the creation of humanlike animals and plants in cartoon films, the design of toys that have human qualities). The approach is also becoming more widespread in interaction design, through the introduction of agents in a range of domains.

What is anthropomorphism? It is well known that people readily attribute human qualities to their pets and their cars, and, conversely, are willing to accept human attributes that have been assigned by others to cartoon characters, robots, toys, and other inanimate objects. Advertisers are well aware of this phenomenon and often create humanlike characters out of inanimate objects to promote their products. For example, breakfast cereals, butter, and fruit drinks have all been transmogrified into characters with human qualities (they move, talk, have personalities, and show emotions), enticing the viewer to buy them. Children are especially susceptible to this kind of "magic," as witnessed in their love of cartoons, where all manner of inanimate objects are brought to life with humanlike qualities.

Examples of its application to system design

The finding that people, especially children, have a propensity to accepting and enjoying objects that have been given humanlike qualities has led many designers into capitalizing on it, most prevalently in the design of human-computer dialogs modeled on how humans talk to each other. A range of animated screen characters, such as agents, friends, advisors and virtual pets, have also been developed.

Anthropomorphism has also been used in the development of cuddly toys that are embedded with computer systems. Commercial products like ActiMates™ have been designed to try to encourage children to learn through playing with the cuddly toys. For example, Barney attempts to motivate play in children by using human-based speech and movement (Strommen, 1998). The toys are programmed to react to the child and make comments while watching TV together or working together on a computer-based task (see Figure 1.2 in Color Plate 1). In particular, Barney is programmed to congratulate the child whenever he or she gets a right answer and also to react to the content on screen with appropriate emotions (e.g., cheering at good news and expressing concern at bad news).

Arguments for exploiting this behavior

An underlying argument in favor of the anthropomorphic approach is that furnishing interactive systems with personalities and other humanlike attributes makes them more enjoyable and fun to interact with. It is also assumed that they can moti-
vate people to carry out the tasks suggested (e.g., learning material, purchasing
goods) more strongly than if they are presented in cold, abstract computer lan-
guage. Being addressed in first person (e.g., "Hello Chris! Nice to see you again. 
Welcome back. Now what were we doing last time? Oh yes, exercise 5. Let's start 
again.") is much more endearing than being addressed in the impersonal third per-
son ("User 24, commence exercise 5"), especially for children. It can make them 
feel more at ease and reduce their anxiety. Similarly, interacting with screen char-
acters like tutors and wizards can be much pleasanter than interacting with a cold 
dialog box or blinking cursor on a blank screen. Typing a question in plain English, 
using a search engine like Ask Jeeves (which impersonates the well-known ficti-
tious butler), is more natural and personable than thinking up a set of keywords, as 
required by other search engines. At the very least, anthropomorphic interfaces are 
a harmless bit of fun.

Arguments against exploiting this behavior

There have been many criticisms of the anthropomorphic approach. Shneiderman 
(1998), one of the best known critics, has written at length about the problems of 
attributing human qualities to computer systems. His central argument is that an-
thropomorphic interfaces, especially those that use first-person dialog and screen 
characters, are downright deceptive. An unpleasant side effect is that they can 
make people feel anxious, resulting in them feeling inferior or stupid. A screen 
tutor that wags its finger at the user and says, "Now, Chris, that's not right! Try 
again. You can do better." is likely to feel more humiliating than a system dialog 
box saying, "Incorrect. Try again."

Anthropomorphism can also lead people into a false sense of belief, enticing 
them to confide in agents called "software bots" that reside in chatrooms and other 
electronic spaces, pretending to be conversant human beings. By far the most com-
mon complaint against computers pretending to have human qualities, however, is 
that people find them very annoying and frustrating. Once users discover that the 
system cannot really converse like a human or does not possess real human quali-
ties (like having a personality or being sincere), they become quickly disillusioned 
and subsequently distrust it. E-commerce sites that pretend to be caring by present-
ing an assortment of virtual assistants, receptionists, and other such helpers are 
seen for what they really are—artificial and flaky. Children and adults alike also are 
quickly bored and annoyed with applications that are fronted by artificial screen 
characters (e.g., tutor wizards) and simply ignore whatever they might suggest.

Evidence for the motion

A number of studies have investigated people's reactions and responses to comput-
ers that have been designed to be more humanlike. A body of work reported by 
Reeves and Nass (1996) has identified several benefits of the anthropomorphic ap-
proach. They found that computers that were designed to flatter and praise users 
when they did something right had a positive impact on how they felt about them-
seves. For example, an educational program was designed to say, "Your question 
makes an interesting and useful distinction. Great job!" after a user had contributed
a new question to it. Students enjoyed the experience and were more willing to continue working with the computer than were other students who were not praised by the computer for doing the same things. In another study, Walker et al. (1994) compared people's responses to a talking-face display and an equivalent text-only one and found that people spent more time with the talking-face display than the text-only one. When given a questionnaire to fill in, the face-display group made fewer mistakes and wrote down more comments. In a follow-up study, Sproull et al. (1996) again found that users reacted quite differently to the two interfaces, with users presenting themselves in a more positive light to the talking-face display and generally interacting with it more.

Evidence against the motion

Sproull et al.'s studies also revealed, however, that the talking-face display made some users feel somewhat disconcerted and displeased. The choice of a stern talking face may have been a large contributing factor. Perhaps a different kind of response would have been elicited if a friendlier smiling face had been used. Nevertheless, a number of other studies have shown that increasing the "humanness" of an interface is counterproductive. People can be misled into believing that a computer is like a human, with human levels of intelligence. For example, one study investigating user's responses to interacting with agents at the interface represented as human guides found that the users expected the agents to be more humanlike than they actually were. In particular, they expected the agents to have personality, emotion, and motivation—even though the guides were portrayed on the screen as simple black and white static icons (see Figure 5.8). Furthermore, the users became disappointed when they discovered the agents did not have any of these characteristics (Oren et al., 1990). In another study comparing an anthropomorphic interface that spoke in the first person and was highly personable (HI THERE, JOHN! IT'S NICE TO MEET YOU, I SEE YOU ARE READY NOW) with a mechanistic one that spoke in third person (PRESS THE ENTER KEY TO

![Guides of historical characters.](image-url)
BEGIN SESSION), the former was rated by college students as less honest and it made them feel less responsible for their actions (Quintanar et al., 1982).

**Casting your vote:** On the basis of this debate and any other articles on the topic (see Section 5.6 and the recommended readings at the end of this chapter) together with your experiences with anthropomorphic interfaces, make up your mind whether you are for or against the motion.

### 5.6 Virtual characters: agents

As mentioned in the debate above, a whole new genre of cartoon and life-like characters has begun appearing on our computer screens—as agents to help us search the web, as e-commerce assistants that give us information about products, as characters in video games, as learning companions or instructors in educational programs, and many more. The best known are videogame stars like Lara Croft and Super Mario. Other kinds include virtual pop stars (See Figure 5.9 on Color Plate 6), virtual talk-show hosts, virtual bartenders, virtual shop assistants, and virtual newscasters. Interactive pets (e.g., Aibo) and other artificial anthropomorphized characters (e.g., Pokemon, Creatures) that are intended to be cared for and played with by their owners have also proved highly popular.

#### 5.6.1 Kinds of agents

Below we categorize the different kinds of agents in terms of the degree to which they anthropomorphize and the kind of human or animal qualities they emulate. These are (1) synthetic characters, (2) animated agents, (3) emotional agents, and (4) embodied conversational interface agents.

1. **Synthetic characters**

These are commonly designed as 3D characters in video games or other forms of entertainment, and can appear as a first-person avatar or a third-person agent. Much effort goes into designing them to be lifelike, exhibiting realistic human movements, like walking and running, and having distinct personalities and traits. The design of the characters' appearance, their facial expressions, and how their lips move when talking are also considered important interface design concerns.

Bruce Blumberg and his group at MIT are developing autonomous animated creatures that live in virtual 3D environments. The creatures are autonomous in that they decide what to do, based on what they can sense of the 3D world, and how they feel, based on their internal states. One of the earliest creatures to be developed was Silas T. Dog (Blumberg, 1996). The 3D dog looks like a cartoon creature (colored bright yellow) but is designed to behave like a real dog (see Figure 5.10). For example, he can walk, run, sit, wag his tail, bark, cock his leg, chase sticks, and rub his head on people when he is happy. He navigates through his world by using his "nose" and synthetic vision. He also has been programmed with various internal goals and needs that he tries to satisfy, including wanting to play
and have company. He responds to events in the environment; for example, he becomes aggressive if a hamster enters his patch.

A person can interact with Silas by making various gestures that are detected by a computer-vision system. For example, the person can pretend to throw a stick, which is recognized as an action that Silas responds to. An image of the person is also projected onto a large screen so that he can be seen in relation to Silas (see Figure 5.10). Depending on his mood, Silas will run after the stick and return it (e.g., when he is happy and playful) or cower and refuse to fetch it (e.g., when he is hungry or sad).

2. **Animated agents**

These are similar to synthetic characters except they tend to be designed to play a collaborating role at the interface. Typically, they appear at the side of the screen as tutors, wizards and helpers intended to help users perform a task. This might be designing a presentation, writing an essay or learning about a topic. Most of the characters are designed to be cartoon-like rather than resemble human beings.

An example of an animated agent is Herman the Bug, who was developed by Intellimedia at North Carolina State University to teach children from kindergarten to high school about biology (Lester et al., 1997). Herman is a talkative, quirky insect that flies around the screen and dives into plant structures as it provides problem-solving advice to students (See Figure 5.11 on Color Plate 7). When providing its explanations it performs a range of activities including walking, flying, shrinking, expanding, swimming, bungee jumping, acrobatics, and teleporting. Its behavior includes 30 animated segments, 160 canned audio clips, and a number of songs. Herman offers advice on how to perform tasks and also tries to motivate students to do them.

3. **Emotional agents**

These are designed with a predefined personality and set of emotions that are manipulated by users. The aim is to allow people to change the moods or emotions of agents and see what effect it has on their behavior. Various mood changers are pro-
vided at the interface in the form of sliders and icons. The effect of requesting an animated agent to become very happy, sad, or grumpy is seen through changes to their behavior. For example, if a user moves a slider to a "scared" position on an emotional scale, the agent starts behaving scared, hiding behind objects and making frightened facial expressions.

The Woggles are one of the earliest forms of emotional agents (Bates, 1994). A group of agents was designed to appear on the screen that played games with one another, such as hide and seek. They were designed as different colored bouncy balls with cute facial expressions. Users could change their moods (e.g., from happy to sad) by moving various sliders, which in turn changed their movement (e.g., they bounced less), facial expression (e.g., they no longer smiled), and how willing they were to play with the other Woggles (See Figure 5.12 on Color Plate 7).

4. **Embodied conversational interface agents**

Much of the research on embodied conversational interface agents has been concerned with how to emulate human conversation. This has included modeling various conversational mechanisms such as:

- recognizing and responding to verbal and non-verbal input
- generating verbal and non-verbal output
- coping with breakdowns, turn-taking and other conversational mechanisms
- giving signals that indicate the state of the conversation as well as contributing new suggestions for the dialog (Cassell, 2000, p.72)

In many ways, this approach is the most anthropomorphic in its aims of all the agent research and development.

Rea is an embodied real-estate agent with a humanlike body that she uses in humanlike ways during a conversation (Cassell, 2000). In particular, she uses eye gaze, body posture, hand gestures, and facial expressions while talking (See Figure 5.13 on Color Plate 8). Although the dialog appears relatively simple, it involves a sophisticated underlying set of conversational mechanisms and gesture-recognition techniques. An example of an actual interaction with Rea is:

Mike approaches the screen and Rea turns to face him and says:

"Hello. How can I help you?"

Mike: "I'm looking to buy a place near MIT."

Rea nods, indicating she is following.

Rea: "I have a house to show you" (picture of a house appears on the screen).

"It is in Somerville."

Mike: "Tell me about it."

Rea looks up and away while she plans what to say.

Rea: "It's big."

Rea makes an expansive gesture with her hands.
Mike brings his hands up as if to speak, so Rea does not continue, waiting for him to speak.
Mike: "Tell me more about it."
Rea: "Sure thing. It has a nice garden . . ."

Which of the various kinds of agents described above do you think are the most convincing? Is it those that try to be as humanlike as possible or those that are designed to be simple, cartoon-based animated characters?

Comment
We argue that the agents that are the most successful are ironically those that are least like humans. The reasons for this include that they appear less phony and are not trying to pretend they are more intelligent or human than they really are. However, others would argue that the more humanlike they are, the more believable they are and hence the more convincing.

5.6.2 General design concerns

Believability of virtual characters
One of the major concerns when designing agents and virtual characters is how to make them believable. By believability is meant "the extent to which users interacting with an agent come to believe that it has its own beliefs, desires and personality" (Lester and Stone, 1997, p 17). In other words, a virtual character that a person can believe in is taken as one that allows users to suspend their disbelief. A key aspect is to match the personality and mood of the character to its actions. This requires deciding what are appropriate behaviors (e.g., jumping, smiling, sitting, raising arms) for different kinds of emotions and moods. How should the emotion "very happy" be expressed? Through a character jumping up and down with a big grin on its face? What about moderately happy — through a character jumping up and down with a small grin on its face? How easy is it for the user to distinguish between these two and other emotions that are expressed by the agents? How many emotions are optimal for an agent to express?

Appearance
The appearance of an agent is very important in making it believable. Parsimony and simplicity are key. Research findings suggest that people tend to prefer simple cartoon-based screen characters to detailed images that try to resemble the human form as much as possible (Scaife and Rogers, 2001). Other research has also found that simple cartoon-like figures are preferable to real people pretending to be artificial agents. A project carried out by researchers at Apple Computer Inc. in the 80s found that people reacted quite differently to different representations of the same interface agent. The agent in question, called Phil, was created as part of a promotional
video called "The Knowledge Navigator." He was designed to respond and behave just like a well-trained human assistant. In one version, he was played by a real actor that appeared on a university professor's computer screen. Thus, he was portrayed as an artificial agent but was played by a real human. The actor was a smartly dressed assistant wearing a white shirt and bow tie. He was also extremely polite. He performed a number of simple tasks at the computer interface, such as reminding the professor of his appointments for that day and alerting him to phone calls waiting. Many people found this version of Phil unrealistic. After viewing the promotional video, people complained about him, saying that he seemed too stupid. In another version, Phil was designed as a simple line-drawn cartoon with limited animation (see Figure 5.14) and was found to be much more likeable (see Laurel, 1993).

**Behavior**

Another important consideration in making virtual characters believable is how convincing their behavior is when performing actions. In particular, how good are they at pointing out relevant objects on the screen to the user, so that the user knows what they are referring to? One way of achieving this is for the virtual character to "lead" with its eyes. For example, Silas the dog turns to look at an object or a person before he actually walks over to it (e.g., to pick the object up or to invite the person to play). A character that does not lead with its eyes looks very mechanical and as such not very life-like (Maes, 1995).

As mentioned previously, an agent's actions need also to match their underlying emotional state. If the agent is meant to be angry, then its body posture, movements, and facial expression all need to be integrated to show this. How this can be achieved effectively can be learned from animators, who have a long tradition in this field. For example, one of their techniques is to greatly exaggerate expressions.
and movements as a way of conveying and drawing attention to an emotional state of a character.

**Mode of interaction**

The way the character communicates with the user is also important. One approach has been towards emulating human conversations as much as possible to make the character's way of talking more convincing. However, as mentioned in the debate above, a drawback of this kind of masquerading is that people can get annoyed easily and feel cheated. Paradoxically, a more believable and acceptable dialog with a virtual character may prove to be one that is based on a simple artificial mode of interaction, in which prerecorded speech is played at certain choice points in the interaction and the user's responses are limited to selecting menu options. The reason why this mode of interaction may ultimately prove more effective is because the user is in a better position to understand what the agent is capable of doing. There is no pretence of a stupid agent pretending to be a smart human.

**Assignment**

*This assignment requires you to write a critique of the persuasive impact of virtual sales agents on customers. Consider what it would take for a virtual sales agent to be believable, trustworthy, and convincing, so that customers would be reassured and happy to buy something based on its recommendations.*

(a) Look at some e-commerce sites that use virtual sales agents (use a search engine to find sites or start with Miss Boo at boo.com, which was working at time of printing) and answer the following:

- What do the virtual agents do?
- What type of agent are they?
- Do they elicit an emotional response from you? If so, what is it?
- What kind of personality do they have?
- How is this expressed?
- What kinds of behavior do they exhibit?
- What are their facial expressions like?
- What is their appearance like? Is it realistic or cartoon-like?
- Where do they appear on the screen?
- How do they communicate with the user (text or speech)?
- Is the level of discourse patronizing or at the right level?
- Are the agents helpful in guiding the customer towards making a purchase?
- Are they too pushy?
- What gender are they? Do you think this makes a difference?
- Would you trust the agents to the extent that you would be happy to buy a product from them? If not, why not?
- What else would it take to make the agents persuasive?
(b) Next, look at an e-commerce website that does not include virtual sales agents but is based on a conceptual model of browsing (e.g., Amazon.com). How does it compare with the agent-based sites you have just looked at?

- Is it easy to find information about products?
- What kind of mechanism does the site use to make recommendations and guide the user in making a purchase?
- Is any kind of personalization used at the interface to make the user feel welcome or special?
- Would the site be improved by having an agent? Explain your reasons either way.

(c) Finally, discuss which site you would trust most and give your reasons for this.

Summary

This chapter has described the different ways interactive products can be designed (both deliberately and inadvertently) to make people respond in certain ways. The extent to which users will learn, buy a product online, chat with others, and so on depends on how comfortable they feel when using a product and how well they can trust it. If the interactive product is frustrating to use, annoying, or patronizing, users easily get angry and despondent, and often stop using it. If, on the other hand, the system is a pleasure, enjoyable to use, and makes the users feel comfortable and at ease, then they are likely to continue to use it, make a purchase, return to the website, continue to learn, etc. This chapter has described various interface mechanisms that can be used to elicit positive emotional responses in users and ways of avoiding negative ones.

Key points

- Affective aspects of interaction design are concerned with the way interactive systems make people respond in emotional ways.
- Well-designed interfaces can elicit good feelings in people.
- Aesthetically pleasing interfaces can be a pleasure to use.
- Expressive interfaces can provide reassuring feedback to users as well as be informative and fun.
- Badly designed interfaces often make people frustrated and angry.
- Anthropomorphism is the attribution of human qualities to objects.
- An increasingly popular form of anthropomorphism is to create agents and other virtual characters as part of an interface.
  People are more accepting of believable interface agents.
- People often prefer simple cartoon-like agents to those that attempt to be humanlike.

Further reading

Turkle, S. (1995) Life on the Screen. New York: Simon and Schuster. This classic covers a range of social impact and affective aspects of how users interact with a variety of computer-based applications. Sherry Turkle discusses at length how computers, the Internet, software, and the design of interfaces affect our identities.
Two very good papers on interface agents can be found in Brenda Laurel's (ed.) *The Art of Human-Computer Interface Design* (1990) Reading, MA.: Addison Wesley:


Excerpts from a lively debate between Pattie Maes and Ben Shneiderman on "Direct manipulation vs. interface agents" can be found *ACM Interactions Magazine*, 4 (6) (1997), 42–61.
Chapter 6

The process of interaction design

6.1 Introduction

Design is a practical and creative activity, the ultimate intent of which is to develop a product that helps its users achieve their goals. In previous chapters, we looked at different kinds of interactive products, issues you need to take into account when doing interaction design and some of the theoretical basis for the field. This chapter is the first of four that will explore how we can design and build interactive products.

Chapter 1 defined interaction design as being concerned with "designing interactive products to support people in their everyday and working lives." But how do you go about doing this?

Developing a product must begin with gaining some understanding of what is required of it, but where do these requirements come from? Whom do you ask about them? Underlying good interaction design is the philosophy of user-centered design, i.e., involving users throughout development, but who are the users? Will they know what they want or need even if we can find them to ask? For an innovative product, users are unlikely to be able to envision what is possible, so where do these ideas come from?

In this chapter, we raise and answer these kinds of questions and discuss the four basic activities and key characteristics of the interaction design process that
were introduced in Chapter 1. We also introduce a lifecycle model of interaction design that captures these activities and characteristics.

The main aims of this chapter are to:

- Consider what 'doing' interaction design involves.
- Ask and provide answers for some important questions about the interaction design process.
- Introduce the idea of a lifecycle model to represent a set of activities and how they are related.
- Describe some lifecycle models from software engineering and HCI and discuss how they relate to the process of interaction design.
- Present a lifecycle model of interaction design.

### 6.2 What is interaction design about?

There are many fields of design, for example graphic design, architectural design, industrial and software design. Each discipline has its own interpretation of "designing." We are not going to debate these different interpretations here, as we are focusing on interaction design, but a general definition of "design" is informative in beginning to understand what it's about. The definition of design from the *Oxford English Dictionary* captures the essence of design very well: "(design is) a plan or scheme conceived in the mind and intended for subsequent execution." The act of designing therefore involves the development of such a plan or scheme. For the plan or scheme to have a hope of ultimate execution, it has to be informed with knowledge about its use and the target domain, together with practical constraints such as materials, cost, and feasibility. For example, if we conceived of a plan for building multi-level roads in order to overcome traffic congestion, before the plan could be executed we would have to consider drivers' attitudes to using such a construction, the viability of the structure, engineering constraints affecting its feasibility, and cost concerns.

In interaction design, we investigate the artifact's use and target domain by taking a user-centered approach to development. This means that users' concerns direct the development rather than technical concerns.

Design is also about trade-offs, about balancing conflicting requirements. If we take the roads plan again, there may be very strong environmental arguments for stacking roads higher (less countryside would be destroyed), but these must be balanced against engineering and financial limitations that make the proposition less attractive. Getting the balance right requires experience, but it also requires the development and evaluation of alternative solutions. Generating alternatives is a key principle in most design disciplines, and one that should be encouraged in interaction design. As Marc Rettig suggested: "To get a good idea, get lots of ideas" (Rettig, 1994). However, this is not necessarily easy, and unlike many design disciplines, interaction designers are not generally trained to generate alternative designs. However, the ability to brainstorm and contribute alternative ideas can be learned, and techniques from other design disciplines can be successfully used in interaction
design. For example, Danis and Boies (2000) found that using techniques from graphic design that encouraged the generation of alternative designs stimulated innovative interactive systems design. See also the interview with Gillian Crampton Smith at the end of this chapter for her views on how other aspects of traditional design can help produce good interaction design.

Although possible, it is unlikely that just one person will be involved in developing and using a system and therefore the plan must be communicated. This requires it to be captured and expressed in some suitable form that allows review, revision, and improvement. There are many ways of doing this, one of the simplest being to produce a series of sketches. Other common approaches are to write a description in natural language, to draw a series of diagrams, and to build prototypes. A combination of these techniques is likely to be the most effective. When users are involved, capturing and expressing a design in a suitable format is especially important since they are unlikely to understand jargon or specialist notations. In fact, a form that users can interact with is most effective, and building prototypes of one form or another (see Chapter 8) is an extremely powerful approach.

So interaction design involves developing a plan which is informed by the product's intended use, target domain, and relevant practical considerations. Alternative designs need to be generated, captured, and evaluated by users. For the evaluation to be successful, the design must be expressed in a form suitable for users to interact with.

### Activity 6.1

Imagine that you want to design an electronic calendar or diary for yourself. You might use this system to plan your time, record meetings and appointments, mark down people's birthdays, and so on, basically the kinds of things you might do with a paper-based calendar. Draw a sketch of the system outlining its functionality and its general look and feel. Spend about five minutes on this.

Having produced an outline, now spend five minutes reflecting on how you went about tackling this activity. What did you do first? Did you have any particular artifacts or experience to base your design upon? What process did you go through?

**Comment**

The sketch I produced is shown in Figure 6.1. As you can see, I was quite heavily influenced by the paper-based books I currently use! I had in mind that this calendar should allow me to record meetings and appointments, so I need a section representing the days and months. But I also need a section to take notes. I am a prolific note-taker, and so for me this was a key requirement. Then I began to wonder about how I could best use hyperlinks. I certainly want to keep addresses and telephone numbers in my calendar, so maybe there could be a link between, say, someone's name in the calendar and their entry in my address book that will give me their contact details when I need them? But I still want the ability to be able to turn page by page, for when I'm scanning or thinking about how to organize my time. A search facility would be useful too.

The first thing that came into my head when I started doing this was my own paper-based book where I keep appointments, maps, telephone numbers, and other small notes. I also thought about my notebook and how convenient it would be to have the two combined. Then I sat and sketched different ideas about how it might look (although I'm not very good at sketching). The sketch in Figure 6.1 is the version I'm happiest with. Note that my sketch
has a strong resemblance to a paper-based book, yet I’ve also tried to incorporate electronic capabilities. Maybe once I have evaluated this design and ensured that the tasks I want to perform are supported, then I will be more receptive to changing the look away from a paper-based "look and feel."

The exact steps taken to produce a product will vary from designer to designer, from product to product, and from organization to organization. In this activity, you may have started by thinking about what you’d like such a system to do for you, or you may have been thinking about an existing paper calendar. You may have mixed together features of different systems or other record-keeping support. Having got or arrived at an idea of what you wanted, maybe you then imagined what it might look like, either through sketching with paper and pencil or in your mind.

### 6.2.1 Four basic activities of interaction design

Four basic activities for interaction design were introduced in Chapter 1, some of which you will have engaged in when doing Activity 6.1. These are: identifying needs and establishing requirements, developing alternative designs that meet those requirements, building interactive versions so that they can be communicated and assessed, and evaluating them, i.e., measuring their acceptability. They are fairly generic activities and can be found in other designs disciplines too. For example, in architectural design (RIBA, 1988) basic requirements are established in a work stage called "inception", alternative design options are considered in a "feasibility" stage and "the brief" is developed through outline proposals and scheme de-
sign. During this time, prototypes may be built or perspectives may be drawn to
give clients a better indication of the design being developed. Detail design speci-
fies all components, and working drawings are produced. Finally, the job arrives on
site and building commences.

We will be expanding on each of the basic activities of interaction design in the
next two chapters. Here we give only a brief introduction to each.

Identifying needs and establishing requirements
In order to design something to support people, we must know who our target
users are and what kind of support an interactive product could usefully provide.
These needs form the basis of the product's requirements and underpin subsequent
design and development. This activity is fundamental to a user-centered approach,
and is very important in interaction design; it is discussed further in Chapter 7.

Developing alternative designs
This is the core activity of designing: actually suggesting ideas for meeting the re-
quirements. This activity can be broken up into two sub-activities: conceptual design
and physical design. Conceptual design involves producing the conceptual model for
the product, and a conceptual model describes what the product should do, behave
and look like. Physical design considers the detail of the product including the col-
ors, sounds, and images to use, menu design, and icon design. Alternatives are con-
sidered at every point. You met some of the ideas for conceptual design in Chapter
2; we go into more detail about conceptual and physical design in Chapter 8.

Building interactive versions of the designs
Interaction design involves designing interactive products. The most sensible way
for users to evaluate such designs, then, is to interact with them. This requires an
interactive version of the designs to be built, but that does not mean that a software
version is required. There are different techniques for achieving "interaction," not
all of which require a working piece of software. For example, paper-based proto-
types are very quick and cheap to build and are very effective for identifying prob-
lems in the early stages of design, and through role-playing users can get a real
sense of what it will be like to interact with the product. This aspect is also covered
in Chapter 8.

Evaluating designs
Evaluation is the process of determining the usability and acceptability of the prod-
uct or design that is measured in terms of a variety of criteria including the number of
errors users make using it, how appealing it is, how well it matches the requirements,
and so on. Interaction design requires a high level of user involvement throughout
development, and this enhances the chances of an acceptable product being deliv-
ered. In most design situations you will find a number of activities concerned with
quality assurance and testing to make sure that the final product is "fit-for-purpose." Evaluation does not replace these activities, but complements and enhances them. We devote Chapters 10 through 14 to the important subject of evaluation.

The activities of developing alternative designs, building interactive versions of the design, and evaluation are intertwined: alternatives are evaluated through the interactive versions of the designs and the results are fed back into further design. This iteration is one of the key characteristics of the interaction design process, which we introduced in Chapter 1.

6.2.2 Three key characteristics of the interaction design process

There are three characteristics that we believe should form a key part of the interaction design process. These are: a user focus, specific usability criteria, and iteration.

The need to focus on users has been emphasized throughout this book, so you will not be surprised to see that it forms a central plank of our view on the interaction design process. While a process cannot, in itself, guarantee that a development will involve users, it can encourage focus on such issues and provide opportunities for evaluation and user feedback.

Specific usability and user experience goals should be identified, clearly documented, and agreed upon at the beginning of the project. They help designers to choose between different alternative designs and to check on progress as the product is developed.

Iteration allows designs to be refined based on feedback. As users and designers engage with the domain and start to discuss requirements, needs, hopes and aspirations, then different insights into what is needed, what will help, and what is feasible will emerge. This leads to a need for iteration, for the activities to inform each other and to be repeated. However good the designers are and however clear the users may think their vision is of the required artifact, it will be necessary to revise ideas in light of feedback, several times. This is particularly true if you are trying to innovate. Innovation rarely emerges whole and ready to go. It takes time, evolution, trial and error, and a great deal of patience. Iteration is inevitable because designers never get the solution right the first time (Gould and Lewis, 1985).

We shall return to these issues and expand upon them in Chapter 9.

6.3 Some practical issues

Before we consider how the activities and key characteristics of interaction design can be pulled together into a coherent process, we want to consider some questions highlighted by the discussion so far. These questions must be answered if we are going to be able to "do" interaction design in practice. These are:

- Who are the users?
- What do we mean by needs?
- How do you generate alternative designs?
- How do you choose among alternatives?
6.3 Some practical issues

6.3.1 Who are the users?

In Chapter 1, we said that an overarching objective of interaction design is to optimize the interactions people have with computer-based products, and that this requires us to support needs, match wants, and extend capabilities. We also stated above that the activity of identifying these needs and establishing requirements was fundamental to interaction design. However, we can't hope to get very far with this intent until we know who the users are and what they want to achieve. As a starting point, therefore, we need to know who we consult to find out the users' requirements and needs.

Identifying the users may seem like a straightforward activity, but in fact there are many interpretations of "user." The most obvious definition is those people who interact directly with the product to achieve a task. Most people would agree with this definition; however, there are others who can also be thought of as users. For example, Holtzblatt and Jones (1993) include in their definition of "users" those who manage direct users, those who receive products from the system, those who test the system, those who make the purchasing decision, and those who use competitive products. Eason (1987) identifies three categories of user: primary, secondary and tertiary. Primary users are those likely to be frequent hands-on users of the system; secondary users are occasional users or those who use the system through an intermediary; and tertiary users are those affected by the introduction of the system or who will influence its purchase.

The trouble is that there is a surprisingly wide collection of people who all have a stake in the development of a successful product. These people are called stakeholders. Stakeholders are "people or organizations who will be affected by the system and who have a direct or indirect influence on the system requirements" (Kotonya and Sommerville, 1998). Dix et al. (1993) make an observation that is very pertinent to a user-centered view of development, that "It will frequently be the case that the formal 'client' who orders the system falls very low on the list of those affected. Be very wary of changes which take power, influence or control from some stakeholders without returning something tangible in its place."

Generally speaking, the group of stakeholders for a particular product is going to be larger than the group of people you'd normally think of as users, although it will of course include users. Based on the definition above, we can see that the group of stakeholders includes the development team itself as well as its managers, the direct users and their managers, recipients of the product's output, people who may lose their jobs because of the introduction of the new product, and so on.

For example, consider again the calendar system in Activity 6.1. According to the description we gave you, the user group for the system has just one member: you. However, the stakeholders for the system would also include people you make appointments with, people whose birthdays you remember, and even companies that produce paper-based calendars, since the introduction of an electronic calendar may increase competition and force them to operate differently.
This last point may seem a little exaggerated for just one system, but if you think of others also migrating to an electronic version, and abandoning their paper calendars, then you can see how the companies may be affected by the introduction of the system.

The net of stakeholders is really quite wide! We do not suggest that you need to involve all of the stakeholders in your user-centered approach, but it is important to be aware of the wider impact of any product you are developing. Identifying the stakeholders for your project means that you can make an informed decision about who should be involved and to what degree.

**ACTIVITY 6.2** Who do you think are the stakeholders for the check-out system of a large supermarket?

**Comment**

First, there are the check-out operators. These are the people who sit in front of the machine and pass the customers’ purchases over the bar code reader, receive payment, hand over receipts, etc. Their stake in the success and usability of the system is fairly clear and direct. Then you have the customers, who want the system to work properly so that they are charged the right amount for the goods, receive the correct receipt, are served quickly and efficiently. Also, the customers want the check-out operators to be satisfied and happy in their work so that they don’t have to deal with a grumpy assistant. Outside of this group, you then have supermarket managers and supermarket owners, who also want the assistants to be happy and efficient and the customers to be satisfied and not complaining. They also don’t want to lose money because the system can’t handle the payments correctly. Other people who will be affected by the success of the system include other supermarket employees such as warehouse staff, supermarket suppliers, supermarket owners’ families, and local shop owners whose business would be affected by the success or failure of the system. We wouldn’t suggest that you should ask the local shop owner about requirements for the supermarket check-out system. However, you might want to talk to warehouse staff, especially if the system links in with stock control or other functions.

6.3.2 What do we mean by "needs"?

If you had asked someone in the street in the late 1990s what she 'needed', I doubt that the answer would have included interactive television, or a jacket which was wired for communication, or a smart fridge. If you presented the same person with these possibilities and asked whether she would buy them if they were available, then the answer would have been different. When we talk about identifying needs, therefore, it's not simply a question of asking people, "What do you need?" and then supplying it, because people don't necessarily know what is possible (see Suzanne Robertson's interview at the end of Chapter 7 for "un-dreamed-of" requirements). Instead, we have to approach it by understanding the characteristics and capabilities of the users, what they are trying to achieve, how they achieve it currently, and whether they would achieve their goals more effectively if they were supported differently.

There are many dimensions along which a user's capabilities and characteristics may vary, and that will have an impact on the product's design. You have met
6.3 Some practical issues

some of these in Chapter 3. For example, a person's physical characteristics may affect the design: size of hands may affect the size and positioning of input buttons, and motor abilities may affect the suitability of certain input and output devices; height is relevant in designing a physical kiosk, for example; and strength in designing a child's toy—a toy should not require too much strength to operate, but may require strength greater than expected for the target age group to change batteries or perform other operations suitable only for an adult. Cultural diversity and experience may affect the terminology the intended user group is used to, or how nervous about technology a set of users may be.

If a product is a new invention, then it can be difficult to identify the users and representative tasks for them; e.g., before microwave ovens were invented, there were no users to consult about requirements and there were no representative tasks to identify. Those developing the oven had to imagine who might want to use such an oven and what they might want to do with it.

It may be tempting for designers simply to design what they would like, but their ideas would not necessarily coincide with those of the target user group. It is imperative that representative users from the real target group be consulted. For example, a company called Netpliance was developing a new "Internet appliance," i.e., a product that would seamlessly integrate all the services necessary for the user to achieve a specific task on the Internet (Isensee et al., 2000). They took a user-centered approach and employed focus group studies and surveys to understand their customers' needs. The marketing department led these efforts, but developers observed the focus groups to learn more about their intended user group. Isensee et al. (p. 60) observe that "It is always tempting for developers to create products they would want to use or similar to what they have done before. However, in the Internet appliance space, it was essential to develop for a new audience that desires a simpler product than the computer industry has previously provided."

In these circumstances, a good indication of future behavior is current or past behavior. So it is always useful to start by understanding similar behavior that is already established. Apart from anything else, introducing something new into people's lives, especially a new "everyday" item such as a microwave oven, requires a culture change in the target user population, and it takes a long time to effect a culture change. For example, before cell phones were so widely available there were no users and no representative tasks available for study, per se. But there were standard telephones and so understanding the tasks people perform with, and in connection with, standard telephones was a useful place to start. Apart from making a telephone call, users also look up people's numbers, take messages for others not currently available, and find out the number of the last person to ring them. These kinds of behavior have been translated into memories for the telephone, answering machines, and messaging services for mobiles. In order to maximize the benefit of e-commerce sites, traders have found that referring back to customers' non-electronic habits and behaviors can be a good basis for enhancing e-commerce activity (CHI panel, 2000; Lee et al., 2000).
6.3.3 How do you generate alternative designs?

A common human tendency is to stick with something that we know works. We probably recognize that a better solution may exist out there somewhere, but it's very easy to accept this one because we know it works—it's "good enough." Setting for a solution that is good enough is not, in itself, necessarily "bad," but it may be undesirable because good alternatives may never be considered, and considering alternative solutions is a crucial step in the process of design. But where do these alternative ideas come from?

One answer to this question is that they come from the individual designer's flair and creativity. While it is certainly true that some people are able to produce wonderfully inspired designs while others struggle to come up with any ideas at all, very little in this world is completely new. Normally, innovations arise through cross-fertilization of ideas from different applications, the evolution of an existing product through use and observation, or straightforward copying of other, similar products. For example, if you think of something commonly believed to be an "invention," such as the steam engine, this was in fact inspired by the observation that the steam from a kettle boiling on the stove lifted the lid. Clearly there was an amount of creativity and engineering involved in making the jump from a boiling kettle to a steam engine, but the kettle provided the inspiration to translate experience gained in one context into a set of principles that could be applied in another. As an example of evolution, consider the word processor. The capabilities of suites of office software have gradually increased from the time they first appeared. Initially, a word processor was just an electronic version of a typewriter, but gradually other capabilities, including the spell-checker, thesaurus, style sheets, graphical capabilities, etc., were added.
So although creativity and invention are often wrapped in mystique, we do understand something of the process and of how creativity can be enhanced or inspired. We know, for instance, that browsing a collection of designs will inspire designers to consider alternative perspectives, and hence alternative solutions. The field of case-based reasoning (Maher and Pu, 1997) emerged from the observation that designers solve new problems by drawing on knowledge gained from solving previous similar problems. As Schank (1982; p. 22) puts it, "An expert is someone who gets reminded of just the right prior experience to help him in processing his current experiences." And while those experiences may be the designer's own, they can equally well be others'.

A more pragmatic answer to this question, then, is that alternatives come from looking at other, similar designs, and the process of inspiration and creativity can be enhanced by prompting a designer's own experience and by looking at others' ideas and solutions. Deliberately seeking out suitable sources of inspiration is a valuable step in any design process. These sources may be very close to the intended new product, such as competitors' products, or they may be earlier versions of similar systems, or something completely different.

Consider again the calendar system introduced at the beginning of the chapter. Reflecting on the process again, what do you think inspired your outline design? See if you can identify any elements within it that you believe are truly innovative.

Comment

For my design, I haven't seen an electronic calendar, although I have seen plenty of other software-based systems. My main sources of inspiration were my current paper-based books. Some of the things you might have been thinking of include your existing paper-based calendar, and other pieces of software you commonly use and find helpful or easy to use in some way. Maybe you already have access to an electronic calendar, which will have given you some ideas, too. However, there are probably other aspects that make the design somehow unique to you and may be innovative to a greater or lesser degree.

All this having been said, under some circumstances the scope to consider alternative designs may be limited. Design is a process of balancing constraints and constantly trading off one set of requirements with another, and the constraints may be such that there are very few viable alternatives available. As another example, if you are designing a software system to run under the Windows operating system, then elements of the design will be prescribed because you must conform to the Windows"look and feel," and to other constraints intended to make Windows programs consistent for the user. We shall return to style guides and standards in Chapter 8.

If you are producing an upgrade to an existing system, then you may face other constraints, such as wanting to keep the familiar elements of it and retain the same "look and feel." However, this is not necessarily a rigid rule. Kent Sullivan reports that when designing the Windows 95 operating system to replace the Windows 3.1 and Windows for Workgroups 3.11 operating systems, they initially focused too much on consistency with the earlier versions (Sullivan, 1996).
BOX 6.1 A Box Full of Ideas

The innovative product design company IDEO was introduced in Chapter 1. It has been involved in the development of many artifacts including the first commercial computer mouse and the PalmPilot V. Underlying some of their creative flair is a collection of weird and wonderful engineering housed in a large flatbed filing cabinet called the TechBox (see Figure 6.2). The TechBox holds around 200 gizmos and interesting materials, divided into categories: "Amazing Materials," "Cool Mechanisms," "Interesting Manufacturing Processes," "Electronic Technologies," and "Thermal and Optical." Each item has been placed in the box because it represents a neat idea or a new process. Staff at IDEO take along a selection of items from the TechBox to brainstorming meetings. The items may be chosen because they provide useful visual props or possible solutions to a particular issue, or simply to provide some light relief.

Each item is clearly labeled with its name and category, but further information can be found by accessing the TechBox's online catalog. Each item has its own page detailing what the item is, why it's interesting, where it came from, and who has used it or knows more about it. For example, the page in Figure 6.3 relates to a metal injection-molding technique.

Other items in the box include an example of metal-coated wood, materials with and without holes that stretch, bend, and change shape or color at different temperatures.

Each TechBox has its own curator who is responsible for maintaining and cataloging the items and for promoting its use within the office. Anyone can submit a new item for consideration and

Figure 6.2 The TechBox at IDEO.
as items become common place, they are removed from the TechBox to make way for the next generation of fascinating contraptions.

How are these things used? Well here is one example from Patrick Hall at the London IDEO office (see Figure 6.4):

IDEO was asked to review the design of a mass-produced hand-held medical product that was deemed to be too big.

As well as brainstorming and other conventional idea-generation methods, I was able to immediately pick out items which I knew about from having used the TechBox in the past: Deep Draw; Fibre-Optic magnifier; Metal Injection molding; Flexy Battery. Further browsing and searching using the keywords search engine highlighted in-mold assembly and light-intensifying film. The
associated web pages for these items enabled me to learn more about these items immediately and indicated who to talk to in IDEO to find out more, and the details of vendors to approach.

The project ended at the feasibility phase, with the client pursuing the technologies I had suggested. Only the fiber-optic magnifier proved (immediately) not to be worth pursuing (because of cost).
6.3 Some practical issues

6.3.4 How do you choose among alternative designs?

Choosing among alternatives is about making design decisions: Will the device use keyboard entry or a touch screen? Will the device provide an automatic memory function or not? These decisions will be informed by the information gathered about users and their tasks, and by the technical feasibility of an idea. Broadly speaking, though, the decisions fall into two categories: those that are about externally visible and measurable features, and those that are about characteristics internal to the system that cannot be observed or measured without dissecting it. For example, externally visible and measurable factors for a building design include the ease of access to the building, the amount of natural light in rooms, the width of corridors, and the number of power outlets. In a photocopier, externally visible and measurable factors include the physical size of the machine, the speed and quality of copying, the different sizes of paper it can use, and so on. Underlying each of these factors are other considerations that cannot be observed or studied without dissecting the building or the machine. For example, the number of

DILEMMA  Copying for Inspiration: Is It Legal?

Designers draw on their experience of design when approaching a new project. This includes the use of previous designs that they know work, both designs they have created themselves and those that others have created. Others’ creations often spark inspiration that also leads to new ideas and innovation. This is well known and understood. However, the expression of an idea is protected by copyright, and people who infringe that copyright can be taken to court and prosecuted. Note that copyright covers the expression of an idea and not the idea itself. This means, for example, that while there are numerous word processors all with similar functionality, this does not represent an infringement of copyright as the idea has been expressed in different ways, and it’s the expression that’s been copyrighted. Copyright is free and is automatically invested in the author of something, e.g., the writer of a book or a programmer who develops a program, unless he signs the copyright over to someone else. Authors writing for academic journals often are asked to sign over their copyright to the publisher of the journal. Various limitations and special conditions can apply, but basically, the copyright is no longer theirs. People who produce something through their employment, such as programs or products, may have in their employment contract a statement saying that the copyright relating to anything produced in the course of that employment is automatically assigned to the employer and does not remain with the employee.

On the other hand, patenting is an alternative to copyright that does protect the idea rather than the expression. There are various forms of patenting, each of which is designed to allow the inventor the chance to capitalize on an idea. It is unusual for software to be patented, since it is a long, slow, and expensive process, although there is a recent trend towards patenting business processes. For example, Amazon, the on-line bookstore, has patented its “one-click” purchasing process, which allows regular users simply to choose a book and buy it with one mouse click (US Patent No. 5960411, September 29, 1999). This is possible because the system stores its customers’ details and “recognizes” them when they access the site again.

So the dilemma comes in knowing when it’s OK to use someone else’s work as a source of inspiration and when you are infringing copyright or patent law. The issues around this question are complex and detailed, and well beyond the scope of this book, but more information and examples of law cases that have been brought successfully and unsuccessfully can be found in Bainbridge (1999).
power outlets will be dependent on how the wiring within the building is designed and the capacity of the main power supply; the choice of materials used in a photocopier may depend on its friction rating and how much it deforms under certain conditions.

In an interactive product there are similar factors that are externally visible and measurable and those that are hidden from the users' view. For example, exactly why the response time for a query to a database (or a web page) is, say, 4 seconds will almost certainly depend on technical decisions made when the database was constructed, but from the users' viewpoint the important observation is the fact that it does take 4 seconds to respond.

In interaction design, the way in which the users interact with the product is considered the driving force behind the design and so we concentrate on the externally visible and measurable behavior. Detailed internal workings are important only to the extent that they affect the external behavior. This does not mean that design decisions concerning a system's internal behavior are any less important: however, the tasks that the user will perform should influence design decisions no less than technical issues.

So, one answer to the question posed above is that we choose between alternative designs by letting users and stakeholders interact with them and by discussing their experiences, preferences and suggestions for improvement. This is fundamental to a user-centered approach to development. This in turn means that the designs must be available in a form that can be reasonably evaluated with users, not in technical jargon or notation that seems impenetrable to them.

One form traditionally used for communicating a design is documentation, e.g., a description of how something will work or a diagram showing its components. The trouble is that a static description cannot capture the dynamics of behavior, and for an interaction device we need to communicate to the users what it will be like to actually operate it.

In many design disciplines, prototyping is used to overcome potential client misunderstandings and to test the technical feasibility of a suggested design and its production. Prototyping involves producing a limited version of the product with the purpose of answering specific questions about the design's feasibility or appropriateness. Prototypes give a better impression of the user experience than simple descriptions can ever do, and there are different kinds of prototyping that are suitable for different stages of development and for eliciting different kinds of information. One experience illustrating the benefits of prototyping is described in Box 6.2. So one important aspect of choosing among alternatives is that prototypes should be built and evaluated by users. We'll revisit the issue of prototyping in Chapter 8.

Another basis on which to choose between alternatives is "quality," but this requires a clear understanding of what "quality" means. People's views of what is a quality product vary, and we don't always write it down. Whenever we use anything we have some notion of the level of quality we are expecting, wanting, or needing. Whether this level of quality is expressed formally or informally does not matter. The point is that it exists and we use it consciously or subconsciously to evaluate alternative items. For example, if you have to wait too long to download
6.3 Some practical issues

If you are a web page, then you are likely to give up and try a different site—you are applying a certain measure of quality associated with the time taken to download the web page. If one cell phone makes it easy to perform a critical function while another involves several complicated key sequences, then you are likely to buy the former rather than the latter. You are applying a quality criterion concerned with efficiency.

Now, if you are the only user of a product, then you don't necessarily have to express your definition of "quality" since you don't have to communicate it to anyone else. However, as we have seen, most projects involve many different stakeholder groups, and you will find that each of them has a different definition of quality and different acceptable limits for it. For example, although all stakeholders may agree on targets such as "response time will be fast" or "the menu structure will be easy to use," exactly what each of them means by this is likely to vary. Disputes are inevitable when, later in development, it transpires that "fast" to one set of stakeholders meant "under a second," while to another it meant "between 2 and 3 seconds." Capturing these different views in clear unambiguous language early in development takes you halfway to producing a product that will be regarded as "good" by all your stakeholders. It helps to clarify expectations, provides a benchmark against which products of the development process can be measured, and gives you a basis on which to choose among alternatives.

The process of writing down formal, verifiable—and hence measurable—usability criteria is a key characteristic of an approach to interaction design called usability engineering that has emerged over many years and with various proponents (Whiteside

---

**BOX 6.2 The Value of Prototyping**

I learned the value of a prototype through a very effective role-playing exercise. I was on a course designed to introduce new graduates to different possible careers in industry. One of the themes was production and manufacturing and the aim of one group exercise was to produce a notebook. Each group was told that it had 30 minutes to deliver 10 books to the person in charge. Groups were given various pieces of paper, scissors, sticky tape, staples, etc., and told to organize ourselves as best we could. So my group set to work organizing ourselves into a production line, with one of us cutting up the paper, another stapling the pages together, another sealing the binding with the sticky tape, and so on. One person was even in charge of quality assurance. It took us less than 10 minutes to produce the 10 books, and we rushed off with our delivery. When we showed the person in charge, he replied, "That's not what I wanted, I need it bigger than that." Of course, the size of the notebook wasn't specified in the description of the task, so we found out how big he wanted it, got some more materials, and scooted back to produce 10 more books. Again, we set up our production line and produced 10 books to the correct size. On delivery, we were again told that it was not what was required; he wanted the binding to work the other way around. This time we got as many of the requirements as we could and went back, developed one book, and took that back for further feedback and refinement before producing the 10 required.

If we had used prototyping as a way of exploring our ideas and checking requirements in the first place, we could have saved so much effort and resource!
et al., 1988; Nielsen, 1993). Usability engineering involves specifying quantifiable measures of product performance, documenting them in a usability specification, and assessing the product against them. One way in which this approach is used is to make changes to subsequent versions of a system based on feedback from carefully documented results of usability tests for the earlier version. We shall return to this idea later when we discuss evaluation.

**Activity 6.4**

Consider the calendar system that you designed in Activity 6.1. Suggest some usability criteria that you could use to determine the calendar's quality. You will find it helpful to think in terms of the usability goals introduced in Chapter 1: effectiveness, efficiency, safety, utility, learnability, and memorability. Be as specific as possible. Check your criteria by considering exactly what you would measure and how you would measure its performance.

Having done that, try to do the same thing for the user experience goals introduced in Chapter 1; these relate to whether a system is satisfying, enjoyable, motivating, rewarding, and so on.

**Comment**

Finding measurable characteristics for some of these is not easy. Here are some suggestions, but you may have found others. Note that the criteria must be measurable and very specific.

- **Effectiveness**: Identifying measurable criteria for this goal is particularly difficult since it is a combination of the other goals. For example, does the system support you in keeping appointments, taking notes, and so on. In other words, is the calendar used?
- **Efficiency**: Assuming that there is a search facility in the calendar, what is the response time for finding a specific day or a specific appointment?
- **Safety**: How often does data get lost or does the user press the wrong button? This may be measured, for example, as the number of times this happens per hour of use.
- **Utility**: How many functions offered by the calendar are used every day, how many every week, how many every month? How many tasks are difficult to complete in a reasonable time because functionality is missing or the calendar doesn't support the right subtasks?
- **Learnability**: How long does it take for a novice user to be able to do a series of set tasks, e.g., make an entry into the calendar for the current date, delete an entry from the current date, edit an entry in the following day?
- **Memorability**: If the calendar isn't used for a week, how many functions can you remember how to perform? How long does it take you to remember how to perform your most frequent task?

Finding measurable characteristics for the user experience criteria is even harder, though. How do you measure satisfaction, fun, motivation or aesthetics? What is entertaining to one person may be boring to another; these kinds of criteria are subjective, and so cannot be measured objectively.

### 6.4 Lifecycle models: showing how the activities are related

Understanding what activities are involved in interaction design is the first step to being able to do it, but it is also important to consider how the activities are related
to one another so that the full development process can be seen. The term lifecycle model is used to represent a model that captures a set of activities and how they are related. Sophisticated models also incorporate a description of when and how to move from one activity to the next and a description of the deliverables for each activity. The reason such models are popular is that they allow developers, and particularly managers, to get an overall view of the development effort so that progress can be tracked, deliverables specified, resources allocated, targets set, and so on.

Existing models have varying levels of sophistication and complexity. For projects involving only a few experienced developers, a simple process would probably be adequate. However, for larger systems involving tens or hundreds of developers with hundreds or thousands of users, a simple process just isn't enough to provide the management structure and discipline necessary to engineer a usable product. So something is needed that will provide more formality and more discipline. Note that this does not mean that innovation is lost or that creativity is stifled. It just means that a structured process is used to provide a more stable framework for creativity.

However simple or complex it appears, any lifecycle model is a simplified version of reality. It is intended as an abstraction and, as with any good abstraction, only the amount of detail required for the task at hand should be included. Any organization wishing to put a lifecycle model into practice will need to add detail specific to its particular circumstances and culture. For example, Microsoft wanted to maintain a small-team culture while also making possible the development of very large pieces of software. To this end, they have evolved a process that has been called "synch and stabilize," as described in Box 6.3.

In the next subsection, we introduce our view of what a lifecycle model for interaction design might look like that incorporates the four activities and the three key characteristics of the interaction design process discussed above. This will form the basis of our discussion in Chapters 7 and 8. Depending on the kind of system being developed, it may not be possible or appropriate to follow this model for every element of the system, and it is certainly true that more detail would be required to put the lifecycle into practice in a real project.

Many other lifecycle models have been developed in fields related to interaction design, such as software engineering and HCI, and our model is evolved from these ideas. To put our interaction design model into context we include here a description of five lifecycle models, three from software engineering and two from HCI, and consider how they relate to it.

---

1Sommerville (2001) uses the term process model to mean what we call a lifecycle model, and refers to the waterfall model as the software lifecycle. Pressman (1992) talks about paradigms. In HCI the term "lifecycle model" is used more widely. For this reason, and because others use "process model" to represent something that is more detailed than a lifecycle model (e.g., Comer, 1997) we have chosen to use lifecycle model.
Microsoft is one of the largest software companies in the world and builds some very complex software; for example, Windows 95 contains more than 11 million lines of code and required more than 200 programmers. Over a two-and-a-half-year period from the beginning of 1993, two researchers, Michael Cusumano and Richard Selby, were given access to Microsoft project documents and key personnel for study and interview. Their aim was to build up an understanding of how Microsoft produces software. Rather than adopt the structured software engineering practices others have followed, Microsoft’s strategy has been to cultivate entrepreneurial flexibility throughout its software teams. In essence, it has tried to scale up the culture of a loosely-structured, small software team. “The objective is to get many small teams (three to eight developers each) or individual programmers to work together as a single relatively large team in order to build large products relatively quickly while still allowing individual programmers and teams freedom to evolve their designs and operate nearly autonomously” (p. 54).

In order to maintain consistency and to ensure that products are eventually shipped, the teams synchronize their activities daily and periodically stabilize the whole product. Cusumano and Selby have therefore labeled Microsoft’s unique process “synchron and stabilize.” Figure 6.5 shows an overview of this process, which is divided into three phases: the planning phase, the development phase, and the stabilization phase.

The planning phase begins with a vision statement that defines the goals of the new product and the user activities to be supported by the product. (Microsoft uses a method called activity-based planning to identify and prioritize the features to be built; we return to this in Chapter 9.) The program managers together with the developers then write a functional specification in enough detail to describe features and to develop schedules and al-

**Planning Phase** Define product vision, specifications, and schedule.

- **Vision Statement** Product and program management use extensive customer input to identify and prioritize order product features.

- **Specification Document** Based on vision statement, program management and development group define feature functionality, architectural issues, and component interdependencies.

- **Schedule and Feature Team Formation** Based on specification document, program management coordinates schedule and arranges feature teams that each contain approximately 1 program manager, 3-8 developers, and 3-8 testers (who work in parallel 1:1 with developers).

**Development Phase** Feature development in a sequential subprojects that each results in a milestone release.

- **Subproject I** First 1/3 of features (Most critical features and shared components)

- **Subproject II** Second 1/3 of features

- **Subproject III** Final 1/3 of features (Least critical features)

**Stabilization Phase** Comprehensive internal and external testing, final product stabilization, and ship.

- **Internal Testing** Thorough testing of complete product within the company.

- **External Testing** Thorough testing of complete product outside the company by “beta” sites, such as OEMs, ISVs, and end users.

- **Release preparation** Prepare final release of “golden master” disks and documentation for manufacturing.
locate staff. The feature list in this document will change by about 30% during the course of development, so the list is not fixed at this time. In the next phase, the development phase, the feature list is divided into three or four parts, each with its own small development team, and the schedule is divided into sequential subprojects, each with its own deadline (milestone). The teams work in parallel on a set of features and synchronize their work by putting together their code and finding errors on a daily and weekly basis. This is necessary because many programmers may be working on the same code at once. For example, during the peak development of Excel 3.0, 34 developers were actively changing the same source code on a daily basis.

At the end of a subproject, i.e., on reaching a milestone, all errors are found and fixed, thus stabilizing the product, before moving on to the next subproject and eventually to the final milestone, which represents the release date. Figure 6.6 shows an overview of the milestone structure for a project with three subprojects. This synch-and-stabilize approach has been used to develop Excel, Office, Publisher, Windows 95, Windows NT, Word, and Works, among others.

---

**Milestone 1 (first 1/3 features)**
- Development (design, coding, prototyping)
- Usability Lab
- Private Release Testing
- Daily Builds
- Feature Debugging
- Feature Integration
- Code Stabilization (no severe bugs)
- Buffer Time (20%–50%)

**Milestone 2 (next 1/3)**
- Development
- Usability Lab
- Private Release Testing
- Daily Builds
- Feature Debugging
- Feature Integration
- Code Stabilization
- Buffer Time

**Milestone 3 (last set)**
- Development
- Usability Lab
- Private Release Testing
- Daily Builds
- Feature Debugging
- Feature Integration
- Feature Complete
- Code Complete
- Code Stabilization
- Buffer Time
- Zero Bug Release
- Release to Manufacturing

---

Figure 6.6 Milestones in the synch and stabilize approach (each taking two to four months).
6.4.1 A simple lifecycle model for interaction design

We see the activities of interaction design as being related as shown in Figure 6.7. This model incorporates iteration and encourages a user focus. While the outputs from each activity are not specified in the model, you will see in Chapter 7 that our description of establishing requirements includes the need to identify specific usability criteria.

The model is not intended to be prescriptive; that is, we are not suggesting that this is how all interactive products are or should be developed. It is based on our observations of interaction design and on information we have gleaned in the research for this book. It has its roots in the software engineering and HCI lifecycle models described below, and it represents what we believe is practiced in the field.

Most projects start with identifying needs and requirements. The project may have arisen because of some evaluation that has been done, but the lifecycle of the new (or modified) product can be thought of as starting at this point. From this activity, some alternative designs are generated in an attempt to meet the needs and requirements that have been identified. Then interactive versions of the designs are developed and evaluated. Based on the feedback from the evaluations, the team may need to return to identifying needs or refining requirements, or it may go straight into redesigning. It may be that more than one alternative design follows this iterative cycle in parallel with others, or it may be that one alternative at a time is considered. Implicit in this cycle is that the final product will emerge in an evolutionary fashion from a rough initial idea through to the finished product. Exactly how this evolution happens may vary from project to project, and we return to this issue in Chapter 8. The only factor limiting the number of times through the cycle is the resources available, but whatever the number is, development ends with an evaluation activity that ensures the final product meets the prescribed usability criteria.

Figure 6.7 A simple interaction design model.
6.4.2 Lifecycle models in software engineering

Software engineering has spawned many lifecycle models, including the waterfall, the spiral, and rapid applications development (RAD). Before the waterfall was first proposed in 1970, there was no generally agreed approach to software development, but over the years since then, many models have been devised, reflecting in part the wide variety of approaches that can be taken to developing software. We choose to include these specific lifecycle models for two reasons: First, because they are representative of the models used in industry and they have all proved to be successful, and second, because they show how the emphasis in software development has gradually changed to include a more iterative, user-centered view.

The waterfall lifecycle model

The waterfall lifecycle was the first model generally known in software engineering and forms the basis of many lifecycles in use today. This is basically a linear model in which each step must be completed before the next step can be started (see Figure 6.8). For example, requirements analysis has to be completed before

![Waterfall Lifecycle Model Diagram]

Figure 6.8 The waterfall lifecycle model of software development.
design can begin. The names given to these steps varies, as does the precise definition of each one, but basically, the lifecycle starts with some requirements analysis, moves into design, then coding, then implementation, testing, and finally maintenance. One of the main flaws with this approach is that requirements change over time, as businesses and the environment in which they operate change rapidly. This means that it does not make sense to freeze requirements for months, or maybe years, while the design and implementation are completed.

Some feedback to earlier stages was acknowledged as desirable and indeed practical soon after this lifecycle became widely used (Figure 6.8 does show some limited feedback between phases). But the idea of iteration was not embedded in the waterfall's philosophy. Some level of iteration is now incorporated in most versions of the waterfall, and review sessions among developers are commonplace. However, the opportunity to review and evaluate with users was not built into this model.

The spiral lifecycle model

For many years, the waterfall formed the basis of most software developments, but in 1988 Barry Boehm (1988) suggested the spiral model of software development (see Figure 6.9). Two features of the spiral model are immediately clear from Figure 6.9: risk analysis and prototyping. The spiral model incorporates them in an iterative framework that allows ideas and progress to be repeatedly checked and evaluated. Each iteration around the spiral may be based on a different lifecycle model and may have different activities.

In the spiral's case, it was not the need for user involvement that inspired the introduction of iteration but the need to identify and control risks. In Boehm's approach, development plans and specifications that are focused on the risks involved in developing the system drive development rather than the intended functionality, as was the case with the waterfall. Unlike the waterfall, the spiral explicitly encourages alternatives to be considered, and steps in which problems or potential problems are encountered to be re-addressed.

The spiral idea has been used by others for interactive devices (see Box 6.4). A more recent version of the spiral, called the WinWin spiral model (Boehm et al., 1998), explicitly incorporates the identification of key stakeholders and their respective "win" conditions, i.e., what will be regarded as a satisfactory outcome for each stakeholder group. A period of stakeholder negotiation to ensure a "win-win" result is included.

Rapid Applications Development (RAD)

During the 1990s the drive to focus upon users became stronger and resulted in a number of new approaches to development. The Rapid Applications Development (RAD) approach attempts to take a user-centered view and to minimize the risk caused by requirements changing during the course of the project. The ideas be-
hind RAD began to emerge in the early 1990s, also in response to the inappropriate nature of the linear lifecycle models based on the waterfall. Two key features of a RAD project are:

- Time-limited cycles of approximately six months, at the end of which a system or partial system must be delivered. This is called time-boxing. In effect, this breaks down a large project into many smaller projects that can deliver products incrementally, and enhances flexibility in terms of the development techniques used and the maintainability of the final system.
• JAD (Joint Application Development) workshops in which users and developers come together to thrash out the requirements of the system (Wood and Silver, 1995). These are intensive requirements-gathering sessions in which difficult issues are faced and decisions are made. Representatives from each identified stakeholder group should be involved in each workshop so that all the relevant views can be heard.

A basic RAD lifecycle has five phases (see Figure 6.10): project set-up, JAD workshops, iterative design and build, engineer and test final prototype, implementation review. The popularity of RAD has led to the emergence of an industry-standard RAD-based method called DSDM (Dynamic Systems Development Method) (Millington and Stapleton, 1995). This was developed by a non-profit-making DSDM consortium made up of a group of companies that recognized the need for some standardization in the field. The first of nine principles stated as underlying DSDM is that "active user involvement is imperative." The DSDM lifecycle is more complicated than the one we've shown here. It involves five phases: feasibility study, business study, functional model iteration, design and build iteration, and implementation. This is only a generic process and must be tailored for a particular organization.

**Activity 6.5** How closely do you think the RAD lifecycle model relates to the interaction design model described in Section 6.4.1?

**Comment** RAD and DSDM explicitly incorporate user involvement, evaluation and iteration. User involvement, however, appears to be limited to the JAD workshop, and iteration appears to be limited to the design and build phase. The philosophy underlying the interaction design model is present, but the flexibility appears not to be. Our interaction design process would be appropriately used within the design and build stage.

![Figure 6.10 A basic RAD lifecycle model of software development.](image-url)
Netpliance, which has moved into the market of providing Internet appliances, i.e., one-stop products that allow a user to achieve a specific Internet-based task, have adopted a user-centered approach to development based on RAD (Isensee et al. 2000). They attribute their ability to develop systems from concept to delivery in seven months to this strong iterative approach: the architecture was revised and iterated over several days; the code was developed with weekly feedback sessions from users; components were typically revised four times, but some went through 12 cycles. Their simple spiral model is shown in Figure 6.11.

The target audience for this appliance, called the i-opener, were people who did not use or own a PC and who may have been uncomfortable around computers. The designers were therefore looking to design something that would be as far away from the “traditional” PC model as possible in terms of both hardware and software. In designing the software, they abandoned the desktop metaphor of the Windows operating system and concentrated on an interface that provided good support for the user’s task. For the hardware design, they needed to get away from the image of a large heavy box with lots of wires and plugs, any one of which may be faulty and cause the user problems.

The device provides three functions: sending and receiving email, categorical content, and web accessibility. That is it. There are no additional features, no complicated menus and options. The device is streamlined to perform these tasks and no more. This choice of functions was based on user studies and testing that served to identify the most frequently used functions, i.e., those that most appropriately supported the users. An example screen showing the news channel for i-opener is shown in Figure 6.12.

Identifying requirements for a new device is difficult. There is no direct experience of using a similar product, and so it is difficult to know what will be used, what will be needed, what will be frustrating, and what will be ignored. The Netpliance team started to gather information for their device by focusing on existing data about PC users: demographics, usability studies, areas of dissatisfaction, etc. They employed marketing research, focus groups, and user surveys to identify the key features of the appliance, and concentrated on delivering these fundamentals well.

The team was multidisciplinary and included hardware engineers, user interface designers, marketing specialists, test specialists, industrial designers, and visual designers. Users were involved throughout development and the whole team took an active part in the design. The interface was designed first, to meet user requirements, and then the hardware and software were developed to fit the interface. In all of this, the emphasis was on a lean development process with a minimum of documentation, early prototyping, and frequent iterations for each component. For example, the design of the hardware proceeded from sketches through pictures to physical prototypes that the users could touch, pick up, move around, and so on. To complement prototyping, the team also used usage scenarios, which are basically descriptions of the appliance’s use to achieve a task. These helped developers to understand how the product could be used from a user’s perspective. We will return to similar techniques in Chapter 7.

Implementation was achieved through rapid cycles of implement and test. Small usability tests were conducted throughout implementation to
Another of the traditions from which interaction design has emerged is the field of HCI (human–computer interaction). Fewer lifecycle models have arisen from this field than from software engineering and, as you would expect, they have a stronger tradition of user focus. We describe two of these here. The first one, the Star, was derived from empirical work on understanding how designers tackled HCI design problems. This represents a very flexible process with evaluation at its core. In contrast, the second one, the usability engineering lifecycle, shows a more structured approach and hails from the usability engineering tradition.

The Star Lifecycle Model

About the same time that those involved in software engineering were looking for alternatives to the waterfall lifecycle, so too were people involved in HCI looking for alternative ways to support the design of interfaces. In 1989, the Star lifecycle
model was proposed by Hartson and Hix (1989) (see Figure 6.13). This emerged from some empirical work they did looking at how interface designers went about their work. They identified two different modes of activity: analytic mode and synthetic mode. The former is characterized by such notions as top-down, organizing, judicial, and formal, working from the systems view towards the user's view; the latter is characterized by such notions as bottom-up, free-thinking, creative and ad hoc, working from the user's view towards the systems view. Interface designers move from one mode to another when designing. A similar behavior has been observed in software designers (Guindon, 1990).

Unlike the lifecycle models introduced above, the Star lifecycle does not specify any ordering of activities. In fact, the activities are highly interconnected: you can move from any activity to any other, provided you first go through the evaluation activity. This reflects the findings of the empirical studies. Evaluation is central to this model, and whenever an activity is completed, its result(s) must be evaluated. So a project may start with requirements gathering, or it may start with evaluating an existing situation, or by analyzing existing tasks, and so on.

The Star lifecycle model has not been used widely and successfully for large projects in industry. Consider the benefits of lifecycle models introduced above and suggest why this may be.

Comment

One reason may be that the Star lifecycle model is extremely flexible. This may be how designers work in practice, but as we commented above, lifecycle models are popular because "they allow developers, and particularly managers, to get an overall view of the development effort so that progress can be tracked, deliverables specified, resources allocated, targets set, and so on.” With a model as flexible as the Star lifecycle, it is difficult to control these issues without substantially changing the model itself.

The Usability Engineering Lifecycle

The Usability Engineering Lifecycle was proposed by Deborah Mayhew in 1999 (Mayhew, 1999). Many people have written about usability engineering, and as
Figure 6.14 The Usability Engineering Lifecycle.
Mayhew herself says, "I did not invent the concept of a Usability Engineering Lifecycle. Nor did I invent any of the Usability Engineering tasks included in the lifecycle...". However, what her lifecycle does provide is a holistic view of usability engineering and a detailed description of how to perform usability tasks, and it specifies how usability tasks can be integrated into traditional software development lifecycles. It is therefore particularly helpful for those with little or no expertise in usability to see how the tasks may be performed alongside more traditional software engineering activities. For example, Mayhew has linked the stages with a general development approach (rapid prototyping) and a specific method (object-oriented software engineering (OOSE, Jacobson et al, 1992)) that have arisen from software engineering.

The lifecycle itself has essentially three tasks: requirements analysis, design/ testing/development, and installation, with the middle stage being the largest and involving many subtasks (see Figure 6.14). Note the production of a set of usability goals in the first task. Mayhew suggests that these goals be captured in a style guide that is then used throughout the project to help ensure that the usability goals are adhered to.

This lifecycle follows a similar thread to our interaction design model but includes considerably more detail. It includes stages of identifying requirements, designing, evaluating, and building prototypes. It also explicitly includes the style guide as a mechanism for capturing and disseminating the usability goals of the project. Recognizing that some projects will not require the level of structure presented in the full lifecycle, Mayhew suggests that some substeps can be skipped if they are unnecessarily complex for the system being developed.

**ACTIVITY 6.7** Study the usability engineering lifecycle and identify how this model differs from our interaction design model described in Section 6.4.1, in terms of the iterations it supports.

**Comment** One of the main differences between Mayhew's model and ours is that in the former the iteration between design and evaluation is contained within the second phase. Iteration between the design/test/development phase and the requirements analysis phase occurs only after the conceptual model and the detailed designs have been developed, prototyped, and
evaluated one at a time. Our version models a return to the activity of identifying needs and establishing requirements after evaluating any element of the design.

Assignment

Nowadays, timepieces (such as clocks, wristwatches etc) have a variety of functions. They not only tell the time and date but they can speak to you, remind you when it’s time to do something, and provide a light in the dark, among other things. Mostly, the interface for these devices, however, shows the time in one of two basic ways: as a digital number such as 23:40 or through an analog display with two or three hands— one to represent the hour, one for the minutes, and one for the seconds.

In this assignment, we want you to design an innovative timepiece for your own use. This could be in the form of a wristwatch, a mantelpiece clock, an electronic clock, or any other kind of clock you fancy. Your goal is to be inventive and exploratory. We have broken this assignment down into the following steps to make it clearer:

(a) Think about the interactive product you are designing: what do you want it to do for you? Find 3-5 potential users and ask them what they would want. Write a list of requirements for the clock, together with some usability criteria based on the definition of usability used in Chapter 1.

(b) Look around for similar devices and seek out other sources of inspiration that you might find helpful. Make a note of any findings that are interesting, useful or insightful.

(c) Sketch out some initial designs for the clock. Try to develop at least two distinct alternatives that both meet your set of requirements.

(d) Evaluate the two designs, using your usability criteria and by role playing an interaction with your sketches. Involve potential users in the evaluation, if possible. Does it do what you want? Is the time or other information being displayed always clear? Design is iterative, so you may want to return to earlier elements of the process before you choose one of your alternatives.

Once you have a design with which you are satisfied, you can send it to us and we shall post a representative sample of those we receive to our website. Details of how to format your submission are available from our website.

Summary

In this chapter, we have looked at the process of interaction design, i.e., what activities are required in order to design an interactive product, and how lifecycle models show the relationships between these activities. A simple interaction design model consisting of four activities was introduced and issues surrounding the identification of users, generating alternative designs, and evaluating designs were discussed. Some lifecycle models from software engineering and HCI were introduced.

Key points

- The interaction design process consists of four basic activities: identifying needs and establishing requirements, developing alternative designs that meet those requirements, building interactive versions of the designs so that they can be communicated and assessed, and evaluating them.
Key characteristics of the interaction design process are explicit incorporation of user involvement, iteration, and specific usability criteria.

Before you can begin to establish requirements, you must understand who the users are and what their goals are in using the device.

Looking at others' designs provides useful inspiration and encourages designers to consider alternative design solutions, which is key to effective design.

Usability criteria, technical feasibility, and users' feedback on prototypes can all be used to choose among alternatives.

Prototyping is a useful technique for facilitating user feedback on designs at all stages.

Lifecycle models show how development activities relate to one another.

The interaction design process is complementary to lifecycle models from other fields.

Further reading

RUDISILL, M., LEWIS, C., POLSON, P. B., AND McKAY, T. D. (1995) (eds.) Human-Computer Interface Design: Success Stories, Emerging Methods, Real-World Context. San Francisco: Morgan Kaufmann. This collection of papers describes the application of different approaches to interface design. Included here is an account of the Xerox Star development, some advice on how to choose among methods, and some practical examples of real-world developments.

BERGMAN, ERIC (2000) (ed.) Information Appliances and Beyond. San Francisco: Morgan Kaufmann. This book is an edited collection of papers which report on the experience of designing and building a variety of 'information appliances', i.e., purpose-built computer-based products which perform a specific task. For example, the Palm Pilot, mobile telephones, a vehicle navigation system, and interactive toys for children.

MAYHEW, DEBORAH J. (1999) The Usability Engineering Lifecycle. San Francisco: Morgan Kaufmann. This is a very practical book about product user interface design. It explains how to perform usability tasks throughout development and provides useful examples along the way to illustrate the techniques. It links in with two software development based methods: rapid prototyping and object-oriented software engineering.

SOMMERVILLE, IAN (2001) Software Engineering (6th edition). Harlow, UK: Addison-Wesley. If you are interested in pursuing the software engineering aspects of the lifecycle models section, then this book provides a useful overview of the main models and their purpose.

NIELSEN, JAKOB (1993) Usability Engineering. San Francisco: Morgan Kaufmann. This is a seminal book on usability engineering. If you want to find out more about the philosophy, intent, history, or pragmatics of usability engineering, then this is a good place to start.
GC: I believe that things should work but they should also delight. In the past, when it was really difficult to make things work, that was what people concentrated on. But now it's much easier to make software and much easier to make hardware. We've got a load of technologies but they're still often not designed for people—and they're certainly not very enjoyable to use. If we think about other things in our life, our clothes, our furniture, the things we eat with, we choose what we use because they have a meaning beyond their practical use. Good design is partly about working really well, but it's also about what something looks like, what it reminds us of, what it refers to in our broader cultural environment. It's this side that interactive systems haven't really addressed yet. They're only just beginning to become part of culture. They are not just a tool for professionals any more, but an environment in which we live.

HS: How do you think we can improve things?

GC: The parallel with architecture is quite an interesting one. In architecture, a great deal of time and expense is put into the initial design; I don't think very much money or time is put into the initial design of software. If you think of the big software engineering companies, how many people work in the design side rather than on the implementation side?

HS: When you say design do you mean conceptual design, or task design, or something else?

GC: I mean all phases of design. Firstly there's research—finding out about people. This is not necessarily limited to finding out about what they want necessarily, because if we're designing new things, they are probably things people don't even know they could have. At the Royal College of Art we tried to work with users, but to be inspired by them, and not constrained by what they know is possible.

The second stage is thinking, "What should this thing we are designing do?" You could call that conceptual design. Then a third stage is thinking how do you represent it, how do you give it form? And then the fourth stage is actually crafting the interface—exactly what color is this pixel? Is this type the right size, or do you need a size bigger? How much can you get on a screen?—all those things about the details.

One of the problems companies have is that the feedback they get is, "I wish it did x." Software looks as if it's designed, not with a basic model of how it works that is then expressed on the interface, but as a load of different functions that are strung together. The desktop interface, although it has great advantages, encourages the idea that you have a menu and you can just add a few more bits when people want more things. In today's word processors, for instance, there isn't a clear conceptual model about how it works, or an underlying theory people can use to reason about why it is not working in the way they expect.

HS: So in trying to put more effort into the design aspect of things, do you think we need different people in the team?

GC: Yes. People in the software field tend to think that designers are people who know how to give the product form, which of course is one of the things they do. But a graphic designer, for instance, is somebody who also thinks at a more strategic level, "What is the message that these people want to get over and to whom?" and then, "What is the best way to give form to a message like that?" The part you see is the beautiful design, the lovely poster or record sleeve, or elegant book, but behind that is a lot of thinking about how to communicate ideas via a particular medium.

HS: If you've got people from different disciplines, have you experienced difficulties in communication?

GC: Absolutely. I think that people from different disciplines have different values, so different results and different approaches are valued. People have different temperaments, too, that have led them to the different fields in the first place, and they've been trained in different ways. In my view the big differ-
ence between the way engineers are trained and the way designers are trained is that engineers are trained to focus in on a solution from the beginning whereas designers are trained to focus out to begin with and then focus in. They focus out and try lots of different alternatives, and they pick some and try them out to see how they go. Then they refine down. This is very hard for both the engineers and the designers because the designers are thinking the engineers are trying to hone in much too quickly and the engineers can't bear the designers faffing about. They are trained to get their results in a completely different way.

HS: Is your idea to make each more tolerant of the other?

GC: Yes, my idea is not to try to make renaissance people, as I don't think it's feasible. Very few people can do everything well. I think the ideal team is made up of people who are really confident and good at what they do and open-minded enough to realize there are very different approaches. There's the scientific approach, the engineering approach, the design approach. All three are different and that's their value—you don't want everybody to be the same. The best combination is where you have engineers who understand design and designers who understand engineering.

It's important that people know their limitations too. If you realize that you need an ergonomist, then you go and find one and you hire them to consult for you. So you need to know what you don't know as well as what you do.

HS: What other aspects of traditional design do you think help with interaction design?

GC: I think the ability to visualize things. It allows people to make quick prototypes or models or sketches so that a group of people can talk about something concrete. I think that's invaluable in the process. I think also making things that people like is just one of the things that good designers have a feel for.

HS: Do you mean aesthetically like or like in its whole sense?

GC: In its whole sense. Obviously there's the aesthetic of what something looks like or feels like but there's also the aesthetic of how it works as well. You can talk about an elegant way of doing something as well as an elegant look.

HS: Another trait I've seen in designers is being protective of their design.

GC: I think that is both a vice and a virtue. In order to keep a design coherent you need to keep a grip on the whole and to push it through as a whole. Otherwise it can happen that people try to make this a bit smaller and cut bits out of that, and so on, and before you know where you are the coherence of the design is lost. It is quite difficult for a team to hold a coherent vision of a design. If you think of other design fields, like film-making, for instance, there is one director and everybody accepts that it's the director's vision. One of the things that's wrong with products like Microsoft Word, for instance, is that there's no coherent idea in it that makes you think, "Oh yes, I understand how this fits with that."

Design is always a balance between things that work well and things that look good, and the ideal design satisfies everything, but in most designs you have to make trade-offs. If you're making a game it's more important that people enjoy it and that it looks good than to worry if some of it's a bit difficult. If you're making a fighter cockpit then the most important thing is that pilots don't fall out of the sky, and so this informs the trade-offs you make. The question is, who decides how to decide the criteria for the trade-offs that inevitably need to be made. This is not a matter of engineering: it's a matter of values—cultural, emotional, aesthetic.

HS: I know this is a controversial issue for some designers. Do you think users should be part of the design team?

GC: No, I don't. I think it's an abdication of responsibility. Users should definitely be involved as a source of inspiration, suggesting ideas, evaluating proposals—saying, "Yes, we think this would be great" or "No, we think this is an appalling idea." But in the end, if designers aren't better than the general public at designing things, what are they doing as designers?
Chapter 7

Identifying needs and establishing requirements

7.1 Introduction

An interaction design project may aim to replace or update an established system, or it may aim to develop a totally innovative product with no obvious precedent. There may be an initial set of requirements, or the project may have to begin by producing a set of requirements from scratch. Whatever the initial situation and whatever the aim of the project, the users' needs, requirements, aspirations, and expectations have to be discussed, refined, clarified, and probably re-scoped. This requires an understanding of, among other things, the users and their capabilities, their current tasks and goals, the conditions under which the product will be used, and constraints on the product's performance.
As we discussed in Chapter 6, identifying users' needs is not as straightforward as it sounds. Establishing requirements is also not simply writing a wish list of features. Given the iterative nature of interaction design, isolating requirements activities from design activities and from evaluation activities is a little artificial, since in practice they are all intertwined: some design will take place while requirements are being established, and the design will evolve through a series of evaluation-redesign cycles. However, each of these activities can be distinguished by its own emphasis and its own techniques.

This chapter provides a more detailed overview of identifying needs and establishing requirements. We introduce different kinds of requirements and explain some useful techniques.

The main aims of this chapter are to:

- Describe different kinds of requirements.
- Enable you to identify examples of different kinds of requirements from a simple description.
- Explain how different data-gathering techniques may be used, and enable you to choose among them for a simple description.
- Enable you to develop a "scenario," a "use case," and an "essential use case" from a simple description.
- Enable you to perform hierarchical task analysis on a simple description.

7.2 What, how, and why?

7.2.1 What are we trying to achieve in this design activity?

There are two aims. One aim is to understand as much as possible about the users, their work, and the context of that work, so that the system under development can support them in achieving their goals; this we call "identifying needs." Building on this, our second aim is to produce, from the needs identified, a set of stable requirements that form a sound basis to move forward into thinking about design. This is not necessarily a major document nor a set of rigid prescriptions, but you need to be sure that it will not change radically in the time it takes to do some design and get feedback on the ideas. Because the end goal is to produce this set of requirements, we shall sometimes refer to this as the requirements activity.

7.2.2 How can we achieve this?

The whole chapter is devoted to explaining how to achieve these aims, but first we give an overview of where we're heading.

At the beginning of the requirements activity, we know that we have a lot to find out and to clarify. At the end of the activity we will have a set of stable requirements that can be moved forward into the design activity. In the middle, there are activities concerned with gathering data, interpreting or analyzing the data, and

'We use interpretation to mean the initial investigation of the data, while analysis is a more detailed study, using a particular frame of reference and notation.
capturing the findings in a form that can be expressed as requirements. Broadly speaking, these activities progress in a sequential manner: first gather some data, then interpret it, then extract some requirements from it, but it gets a lot messier than this, and the activities influence one another as the process iterates. One of the reasons for this is that once you start to analyze data, you may find that you need to gather some more data to clarify or confirm some ideas you have. Another reason is that the way in which you document your requirements may affect your analysis, since it will enable you to identify and express some aspects more easily than others. For example, using a notation which emphasizes the data-flow characteristics of a situation will lead the analysis to focus on this aspect rather than, for example, on data structure. Analysis requires some kind of framework, theory or hypothesis to provide a frame of reference, however informal, and this will inevitably affect the requirements you extract. To overcome this, it is important to use a complementary set of data-gathering techniques and data-interpretation techniques, and to constantly revise and refine the requirements. As we discuss below, there are different kinds of requirements, and each can be emphasized or de-emphasized by the different techniques.

Identifying needs and establishing requirements is itself an iterative activity in which the subactivities inform and refine one another. It does not last for a set number of weeks or months and then finish. In practice, requirements evolve and develop as the stakeholders interact with designs and see what is possible and how certain facilities can help them. And as shown in the lifecycle model in Chapter 6, the activity itself will be repeatedly revisited.

7.2.3 Why bother? The importance of getting it right

An article published in January 2000 (Taylor, 2000) investigated the causes of IT project failure. The article admits that "there is no single cause of IT project failure," but requirements issues figured highly in the findings. The research involved detailed questioning of 38 IT professionals in the UK. When asked about which project stages caused failure, respondents mentioned "requirements definition" more than any other phase. When asked about cause of failure, "unclear objectives and requirements" was mentioned more than anything else, and for critical success factors, "clear, detailed requirements" was mentioned most often.

As stressed in previous chapters, understanding what the product under development should do and ensuring that it supports stakeholders' needs are critically important activities in any product development. If the requirements are wrong then the product will at best be ignored and at worst be despised by the users, and will cause grief and lost productivity. In either case, the implications for both producer and customer are serious: anxiety and frustration, lost revenue, loss of customer confidence, and so on. However we look at it, getting the requirements of the product wrong is a very bad move and something to be avoided at all costs.

Taking a user-centered approach to development is one way to address this. If users' voices and needs are clearly heard and taken into account, then it is more likely that the end result will meet users' needs and expectations. Involving users isn't always easy, however, and we explore in more detail how to do this effectively.
in Chapter 9. Here we focus on establishing the requirements, while keeping the emphasis clearly on users' needs.

### 7.2.4 Why establish requirements?

The activity of understanding what a product should do has been given various labels—for example, requirements gathering, requirements capture, requirements elicitation, requirements analysis, and requirements engineering. The first two imply that requirements exist out there and we simply need to pick them up or catch them. “Elicitation” implies that “others” (presumably the clients or users) know the requirements and we have to get them to tell us. Requirements, however, are not that easy to identify. You might argue that, in some cases, customers must know what the requirements are because they know the tasks that need to be performed, and may have asked for a system to be built in the first place. However, they may not have articulated requirements as yet, and even if they have an initial set of requirements, they probably have not explored them in sufficient detail for development to begin.

The term "requirements analysis" is normally used to describe the activity of investigating and analyzing an initial set of requirements that have been gathered, elicited, or captured. Analyzing the information gathered is an important step, since it is this interpretation of the facts, rather than the facts themselves, that inspires the design. Requirements engineering is a better term than the others because it recognizes that developing a set of requirements is an iterative process of evolution and negotiation, and one that needs to be carefully managed and controlled.

We chose the term establishing requirements to represent the fact that requirements arise from the data-gathering and interpretation activities and have been established from a sound understanding of the users' needs. This also implies that requirements can be justified by and related back to the data collected.

### 7.3 What are requirements?

Before we go any further, we need to explain what we mean by a requirement. Intuitively, you probably have some understanding of what a requirement is, but we should be clear. A requirement is a statement about an intended product that specifies what it should do or how it should perform. One of the aims of the requirements activity is to make the requirements as specific, unambiguous, and clear as possible. For example, a requirement for a website might be that the time to download any complete page is less than 5 seconds. Another less precise example might be that teenage girls should find the site appealing. In the case of this latter example, further investigation would be necessary to explore exactly what teenage girls would find appealing. Requirements come in many different forms and at many different levels of abstraction, but we need to make sure that the requirements are as clear as possible and that we understand how to tell when they have been fulfilled. The example requirement shown in Figure 7.1 is expressed using a template from the Volere process (Robertson and Robertson, 1999), which you'll hear more about later in this chapter and in Suzanne Robertson's interview at the end of this
7.3 What are requirements?

7.3.1 Different kinds of requirements

In software engineering, two different kinds of requirements have traditionally been identified: functional requirements, which say what the system should do, and non-functional requirements, which say what constraints there are on the system and its development. For example, a functional requirement for a word processor may be that it should support a variety of formatting styles. This requirement might then be decomposed into more specific requirements detailing the kind of formatting required such as formatting by paragraph, by character, and by document, down to a very specific level such as that character formatting must include 20 typefaces, each with bold, italic, and standard options. A non-functional requirement for a word processor might be that it must be able to run on a variety of platforms such as PCs, Macs, and Unix machines. Another might be that it must be able to function on a computer with 64 MB RAM. A different kind of non-functional requirement would be that it must be delivered in six months' time. This represents a constraint on the development activity itself rather than on the product being developed.

If we consider interaction devices in general, other kinds of non-functional requirements become relevant such as physical size, weight, color, and production.

*See Figure 7.5 for an explanation of these fields.*
feasibility. For example, when the PalmPilot was developed (Bergman and Haiti, 2000), an overriding requirement was that it should be physically as small as possible, allowing for the fact that it needed to incorporate batteries and an LCD display. In addition, there were extremely tight constraints on the size of the screen, and that had implications for the number of pixels available to display information. For example, formatting lines or certain typefaces may become infeasible to use if they take up even one extra pixel. Figure 7.2 shows two screen shots from the PalmPilot development. As you can see, removing the line at the left-hand side of the display in the top window released sufficient pixels to display the missing "s" in the bottom window.

Interaction design requires us to understand the functionality required and the constraints under which the product must operate or be developed. However, instead of referring to all requirements that are not functional as simply "non-functional" requirements, we prefer to refine this into further categories. The following is not an exhaustive list of the different requirements we need to be looking out for (see the figure in Suzanne Robertson’s interview at the end of this chapter for a more detailed list), nor is it a tight categorization, however, it does illustrate the variety of requirements that need to be captured.

**Functional requirements** capture what the product should do. For example, a functional requirement for a smart fridge might be that it should be able to tell when the butter tray is empty. Understanding the functional requirements for an interactive product is very important.

**Data requirements** capture the type, volatility, size/amount, persistence, accuracy, and value of the amounts of the required data. All interactive devices have to handle greater or lesser amounts of data. For example, if the system under consid-

---

**Figure 7.2** Every pixel counts.
7.3 What are requirements?

The operation is to operate in the share-dealing application domain, then the data must be up-to-date and accurate, and is likely to change many times a day. In the personal banking domain, data must be accurate, must persist over many months and probably years, is very valuable, and there is likely to be a lot of it.

**Environmental requirements** or **context of use** refer to the circumstances in which the interactive product will be expected to operate. Four aspects of the environment must be considered when establishing requirements. First is the physical environment such as how much lighting, noise, and dust is expected in the operational environment. Will users need to wear protective clothing, such as large gloves or headgear, that might affect the choice of interaction paradigm? How crowded is the environment? For example, an ATM operates in a very public physical environment. Using speech to interact with the customer is therefore likely to be problematic.

The second aspect of the environment is the social environment. The issues raised in Chapter 4 regarding the social aspects of interaction design, such as collaboration and coordination, need to be explored in the context of the current development. For example, will data need to be shared? If so, does the sharing have to be synchronous, e.g., does everyone need to be viewing the data at once, or asynchronous, e.g., two people authoring a report take turns in editing and adding to it? Other factors include the physical location of fellow team members, e.g., do collaborators have to communicate across great distances?

The third aspect is the organizational environment, e.g., how good is user support likely to be, how easily can it be obtained, and are there facilities or resources for training? How efficient or stable is the communications infrastructure? How hierarchical is the management? and so on.

Finally, the technical environment will need to be established; for example, what technologies will the product run on or need to be compatible with, and what technological limitations might be relevant?

**User requirements** capture the characteristics of the intended user group. In Chapter 6 we mentioned the relevance of a user's abilities and skills, and these are an important aspect of user requirements. But in addition to these, a user may be a novice, an expert, a casual, or a frequent user. This affects the ways in which interaction is designed. For example, a novice user will require step-by-step instructions, probably with prompting, and a constrained interaction backed up with clear information. An expert, on the other hand, will require a flexible interaction with more wide-ranging powers of control. If the user is a frequent user, then it would be important to provide short cuts such as function keys rather than expecting them to type long commands or to have to navigate through a menu structure. A casual or infrequent user, rather like a novice, will require clear instructions and easily understood prompts and commands, such as a series of menus. The collection of attributes for a "typical user" is called a user profile. Any one device may have a number of different user profiles.

Note that user requirements are not the same as usability requirements. We discuss the latter below.

**Usability requirements** capture the usability goals and associated measures for a particular product. In Chapter 6 we introduced the idea of usability engineering,
Identifying needs and establishing requirements

An approach in which specific measures for the usability goals of the product are established and agreed upon early in the development process and are then revisited, and used to track progress as development proceeds. This both ensures that usability is given due priority and facilitates progress tracking. In Chapter 1 we described a number of usability goals: effectiveness, efficiency, safety, utility, learnability, and memorability. If we are to follow the philosophy of usability engineering and meet these usability goals, then we must identify the appropriate requirements. Chapter 1 also described some user experience goals, such as making products that are fun, enjoyable, pleasurable, aesthetically pleasing, and motivating. As we observed in Chapter 6, it is harder to identify quantifiable measures that allow us to track these qualities, but an understanding of how important each of these is to the current development should emerge as we learn more about the intended product.

Usability requirements are related to other kinds of requirement we must establish, such as the kinds of users expected to interact with the product.

**BOX 7.1 Underwater PCs**

Developing a PC for undersea divers to take underwater has one major environmental factor: it's surrounded by water! However, waterproofing is not the main issue for the designers at WetPC, a company who have produced such a system. The interface has proved to be more of a problem. Divers typically have only one hand free to operate the computer, and are likely to be swimming and moving up and down in the water at the same time. So a traditional interface design is no good.

Early prototypes of the computer used voice recognition, but the bubbles made too much noise and distorted the sound. Tracker balls were also inappropriate because the divers are not working on a flat surface. So the main developer at WetPC, Bruce Macdonald, devised a "keyboard" called a KordGrip that has five keys (see Figure 7.3(a)). Combinations of keys represent different symbols, so that divers can choose items from menus. They can perform operations such as controlling a camera and sending messages. The system is also linked to a GPS system that tells the divers where they are. This makes it much easier to mark the location of mines and other underwater discoveries (See Figure 7.3(b) on Color Plate 8).

![Figure 7.3 (a) The KordGrip interface; (b) the KordGrip in use underwater.](image)
Suggest one key functional, data, environmental, user and usability requirement for each of the following scenarios:

(a) A system for use in a university's self-service cafeteria that allows users to pay for their food using a credit system.
(b) A system to control the functioning of a nuclear power plant.
(c) A system to support distributed design teams, e.g., for car design.

Comment

You may have come up with alternative suggestions; these are indicative of the kinds of answer we might expect.

(a) **Functional:** The system will calculate the total cost of purchases.

**Data:** The system must have access to the price of products in the cafeteria.

**Environmental:** Cafeteria users will be carrying a tray and will most likely be in a reasonable rush. The physical environment will be noisy and busy, and users may be talking with friends and colleagues while using the system.

**User:** The majority of users are likely to be under 25 and comfortable dealing with technology.

**Usability:** The system needs to be simple so that new users can use the system immediately, and memorable for more frequent users. Users won't want to wait around for the system to finish processing, so it needs to be efficient and to be able to deal easily with user errors.

(b) **Functional:** The system will be able to monitor the temperature of the reactors.

**Data:** The system will need access to temperature readings.

**Environmental:** The physical environment is likely to be uncluttered and to impose few restrictions on the console itself unless there is a need to wear protective clothing (depending on where the console is to be located).

**User:** The user is likely to be a well-trained engineer or scientist who is competent to handle technology.

**Usability:** Outputs from the system, especially warning signals and gauges, must be clear and unambiguous.

(c) **Functional:** The system will be able to communicate information between remote sites.

**Data:** The system must have access to design information that will be captured in a common file format (such as AutoCAD).

**Environmental:** Physically distributed over a wide area. Files and other electronic media need to be shared. The system must comply with available communication protocols and be compatible with network technologies.

**User:** Professional designers, who may be worried about technology but who are likely to be prepared to spend time learning a system that will help them perform their jobs better. The design team is likely to be multi-lingual.

**Usability:** Keeping transmission error rate low is likely to be of high priority.
Chapter 7  Identifying needs and establishing requirements

7.4 Data gathering

So how do we go about determining requirements? Data gathering is an important part of the requirements activity and also of evaluation. In this chapter, we concentrate on data gathering for the requirements activity. Further information about the techniques we present here and how to apply them in evaluation is in Chapters 12 through 14.

The purpose of data gathering is to collect sufficient, relevant, and appropriate data so that a set of stable requirements can be produced. Even if a set of initial requirements exists, data gathering will be required to expand, clarify, and confirm those initial requirements. Data gathering needs to cover a wide spectrum of issues because the different kinds of requirement we need to establish are quite varied, as we saw above. We need to find out about the tasks that users currently perform and their associated goals, the context in which the tasks are performed, and the rationale for why things are the way they are.

There is essentially a small number of basic techniques for data gathering, but they are flexible and can be combined and extended in many ways; this makes the possibilities for data gathering very varied, to give full leverage on understanding the variety of requirements we seek. These techniques are questionnaires, interviews, focus groups and workshops, naturalistic observation, and studying documentation. Some of them, such as the interview, require active participation from stakeholders, while others, such as studying documentation, require no involvement at all. In addition, various props can be used in data-gathering sessions, such as descriptions of common tasks and prototypes of possible new functionality. See Section 7.6 and Chapter 8 for further information on how to develop these props. Box 7.2 gives an

<table>
<thead>
<tr>
<th>BOX 7.2</th>
<th>Combining Data-Gathering Techniques and Props to Understand Different Requirements</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rudman and Engelbeck (1996) describe how they used different techniques to establish the requirements for a complex graphical user interface for a telephone company, and how different methods resulted in understanding different requirements. They used five different techniques:</td>
<td></td>
</tr>
</tbody>
</table>

1. On-site observation allowed them to understand the nature of the current business.
2. Participatory prototyping, i.e., active involvement of the stakeholders in designing a prototype, allowed them to take advantage of the employees' knowledge.
3. Interviews aimed at understanding the background business of the company allowed them to understand the complex nature of the wider domain.
4. Interviews aimed at understanding the decision sequences of employees allowed them to create dialogs to support two-party negotiations.
5. Role-playing prototype walkthroughs using simulated scenarios also helped to create dialogs to support two-party negotiations.

The difference between the third and fourth techniques lies in the focus of the questioning and in the notation used to capture data. In the third technique, interviewers focused on understanding the domain of the application and captured information using semantic nets, which are specifically designed to represent such information. In the fourth technique, decision trees were used to understand the goals, decision points, and options considered by employees when dealing with a customer.
example of how different methods and props can be combined to gain maximum advantage, while Box 7.3 describes a very different approach aimed at prompting inspiration rather than simple data gathering.

### 7.4.1 Data-gathering techniques

In addition to the most common forms of data-gathering techniques listed above, if a system is currently operational then data logging may be used. This involves instrumenting the software to record users' activity in a log that can be examined later. Each of the techniques will yield different kinds of data and are useful in different circumstances. In most cases, they are also used in evaluation, and how to implement them is described in Chapters 12 and 13. Here we describe what each technique involves and explain the circumstances for which they are most suitable, in the context of the requirements activity. The discussion is summarized in Table 7.1 on page 214.

**Questionnaires.** Most of us are familiar with questionnaires. They are a series of questions designed to elicit specific information from us. The questions may require different kinds of answers: some require a simple YES/NO, others ask us to choose from a set of pre-supplied answers, and others ask for a longer response or comment. Sometimes questionnaires are sent in electronic form and arrive via email or are posted on a website, and sometimes they are given to us on paper. In most cases the questionnaire is administered at a distance, i.e., no one is there to help you answer the questions or to explain what they mean.

Well-designed questionnaires are good at getting answers to specific questions from a large group of people, and especially if that group of people is spread across a wide geographical area, making it infeasible to visit them all. Questionnaires are often used in conjunction with other techniques. For example, information obtained through interviews might be corroborated by sending a questionnaire to a wider group of stakeholders to confirm the conclusions.

**Interviews.** Interviews involve asking someone a set of questions. Often interviews are face-to-face, but they don't have to be. Companies spend large amounts of money conducting telephone interviews with their customers finding out what they like or don't like about their service. If interviewed in their own work or home setting, people may find it easier to talk about their activities by showing the interviewer what they do and what systems and other artifacts they use. The context can also trigger them to remember certain things, for example a problem they have downloading email, which they would not have recalled had the interview taken place elsewhere.

Interviews can be broadly classified as structured, unstructured or semi-structured, depending on how rigorously the interviewer sticks to a prepared set of questions.

In the requirements activity, interviews are good at getting people to explore issues and unstructured interviews are often used early on to elicit scenarios (see Section 7.6 below). Interacting with a human rather than a sterile, impersonal piece of paper or electronic questionnaire encourages people to respond, and can make the exercise more pleasurable. In the context of establishing requirements, it is equally important for development team members to meet stakeholders and for users to feel involved. This on its own may be sufficient motivation to arrange interviews.
An alternative approach to understanding users and their needs was taken in a European Union-funded project called the Presence Project (Gaver et al., 1999). This work arose from research looking at novel interaction techniques to increase the presence of elderly people in their local community. Three different groups were studied: one in Oslo, Norway, one near Amsterdam, The Netherlands, and one near Pisa, Italy. One of the problems with designing for an unknown culture is that it can be difficult to understand or appreciate the needs of that culture. Rather than take a more traditional approach of questionnaires, interviews or ethnographic studies, this project used "cultural probes." These probes consisted of a wallet containing a variety of items: 8 to 10 postcards, about seven maps, a disposable camera, a photo album, and a media diary (see Figure 7.4). The intent was for the recipients to look through the wallet and answer questions associated with certain probes that they contained, then to return the items directly to the researchers when they had finished with them.

The postcards had pictures on the front and questions on the back, and were pre-addressed and stamped so that they could be easily returned. Questions included "Please tell us a piece of advice or insight that has been important to you," "What place does art have in your life?" and "Tell us about your favorite device." The maps and associated inquiries were designed to find out about the participants' attitudes towards their environment. They were printed on various textured papers and were in the form of folding envelopes, also to facilitate their return. On local maps, participants were asked to mark sites where they would go to meet people, to be alone, to daydream and where they would like to go, but can't. On a map of the world, they were asked to mark places where they had been.

Participants were asked to use the camera to take pictures of their home, what they will wear today (whenever "today" was), the first person you see today, something desirable, and something boring. In the photo album they were asked to tell the researchers their story in pictures. The media diary was to record their use of television and radio.

The approach taken by these researchers was not to identify specific user needs but to seek inspiration that would lead to new opportunities, new pleasures, new forms of sociability, and new cultural forms. Hence they were seeking inspiration rather than requirements.

The probes were returned over a period of a month or so, at different rates and in different quantities for each group. The data were not analyzed per se, but the resulting designs reflect what the designers learned.

For the Dutch site, they proposed building a network of computer displays with which the elderly could help inhabitants communicate their values and attitudes about the culture.

For the Norwegians, they proposed that the elders should lead a community-wide conversation about social issues, publishing questions
However, interviews are time consuming and it may not be feasible to visit all the people you'd like to see.

**Focus groups and workshops.** Interviews tend to be one on one, and elicit only one person's perspective. As an alternative or as corroboration, it can be very revealing to get a group of stakeholders together to discuss issues and requirements. These sessions can be very structured with set topics for discussion, or can be unstructured. In this latter case, a facilitator is required who can keep the discussion on track and can provide the necessary focus or redirection when appropriate. In some development methods, workshops have become very formalized. For example, the workshops used in Joint Application Development (Wood and Silver, 1995) are very structured, and their contents and participants are all prescribed.

In the requirements activity, focus groups and workshops are good at gaining a consensus view and/or highlighting areas of conflict and disagreement. On a social level it also helps for stakeholders to meet designers and each other, and to express their views in public. It is not uncommon for one set of stakeholders to be unaware that their views are different from another's even though they are in the same organization. On the other hand, these sessions need to be structured carefully and the participants need to be chosen carefully. It is easy for one or a few people to dominate discussions, especially if they have control, higher status, or influence over the other participants.

**Naturalistic observation.** It can be very difficult for humans to explain what they do or to even describe accurately how they achieve a task. So it is very unlikely that a designer will get a full and true story from stakeholders by using any of the techniques listed above. The scenarios and other props used in interviews and workshops will help prompt people to be more accurate in their descriptions, but observation provides a richer view. Observation involves spending some time with the stakeholders as they go about their day-to-day tasks, observing work as it happens, in its natural setting. A member of the design team shadows a stakeholder, making notes, asking questions (but not too many), and observing what is being done in the natural context of the activity. This is an invaluable way to gain insights into the tasks of the stakeholders that can complement other investigations. The level of involvement of the observer in the work being observed is variable along a spectrum with no involvement (outside observation) at one end and full involvement (participant observation) at the other.
### Table 7.1 Overview of data-gathering techniques used in the requirements activity

<table>
<thead>
<tr>
<th>Technique</th>
<th>Good for</th>
<th>Kind of data</th>
<th>Advantages</th>
<th>Disadvantages</th>
<th>Detail for designing in</th>
</tr>
</thead>
<tbody>
<tr>
<td>Questionnaires</td>
<td>Answering specific questions</td>
<td>Quantitative and qualitative data</td>
<td>Can reach many people with low resource</td>
<td>The design is crucial. Response rate may be low. Responses may not be what you want</td>
<td>Chapter 13</td>
</tr>
<tr>
<td>Interviews</td>
<td>Exploring issues</td>
<td>Some quantitative but mostly qualitative data</td>
<td>Interviewer can guide interviewee if necessary. Encourages contact between developers and users</td>
<td>Time consuming. Artificial environment may intimidate interviewee</td>
<td>Chapter 13</td>
</tr>
<tr>
<td>Focus groups and workshops</td>
<td>Collecting multiple viewpoints</td>
<td>Some quantitative but mostly qualitative data</td>
<td>Highlights areas of consensus and conflict. Encourages contact between developers and users</td>
<td>Possibility of dominant characters</td>
<td>Chapter 13</td>
</tr>
<tr>
<td>Naturalistic observation</td>
<td>Understanding context of user activity</td>
<td>Qualitative</td>
<td>Observing actual work gives insights that other techniques can't give</td>
<td>Very time consuming. Huge amounts of data</td>
<td>Chapter 12</td>
</tr>
<tr>
<td>Studying documentation</td>
<td>Learning about procedures, regulations and standards</td>
<td>Quantitative</td>
<td>No time commitment from users required</td>
<td>Day-to-day working will differ from documented procedures</td>
<td>N/A</td>
</tr>
</tbody>
</table>

Not only can naturalistic observation help fill in details and nuances that simply did not come out of the other investigations, it also provides context for tasks. **Contextualizing** the work or behavior that a device is to support provides data that other techniques cannot, and from which we can evolve requirements.

In the requirements activity, observation is good for understanding the nature of the tasks and the context in which they are performed. However, it requires more time and commitment from a member of the design team, and it can result in a huge amount of data.

**Studying documentation.** Procedures and rules are often written down in manuals and these are a good source of data about the steps involved in an activity and
7.4 Data gathering

Any regulations governing a task. Such documentation should not be used as the only source, however, as everyday practices may augment them and may have been devised by those concerned to make the procedures work in a practical setting. Taking a user-centered view of development means that we are interested in the everyday practices rather than an idealized account.

Other documentation that might be studied includes diaries or job logs that are written by the stakeholders during the course of their work.

In the requirements activity, studying documentation is good for understanding legislation and getting some background information on the work. It also doesn't involve stakeholder time, which is a limiting factor on the other techniques.

7.4.2 Choosing between techniques

Table 7.1 provides some information to help you choose a set of techniques for a specific project. It tells you the kind of information you can get, e.g., answers to specific questions, and the kind of data it yields, e.g., qualitative or quantitative. It also includes some advantages and disadvantages for each technique. The kind of information you want will probably be determined by where you are in the cycle of iterations. For example, at the beginning of the project you may not have any specific questions that need answering, so it's better to spend time exploring issues through interviews rather than sending out questionnaires. Whether you want qualitative or quantitative data may also be affected by the point in development you have reached, but is also influenced by the kind of analysis you need to do.

The resources available will influence your choice, too. For example, sending out questionnaires nationwide requires sufficient time, money, and people to do a good design, try it out (i.e., pilot it), issue it, collate the results and analyze them. If you only have three weeks and no one on the team has designed a survey before, then this is unlikely to be a success.

Finally, the location and accessibility of the stakeholders need to be considered. It may be attractive to run a workshop for a large group of stakeholders, but if they are spread across a wide geographical area, it is unlikely to be practical.

Olson and Moran (1996) suggest that choosing between data-gathering techniques rests on two issues: the nature of the data gathering technique itself and the task to be studied.

Data-gathering techniques differ in two main respects:

1. The amount of time they take and the level of detail and risk associated with the findings. For example, they claim that a naturalistic observation will take two days of effort and three months of training, while interviews take one day of effort and one month of training (p. 276).

2. The knowledge the analyst must have about basic cognitive processes.

Tasks can be classified along three scales:

1. Is the task a set of sequential steps or is it a rapidly overlapping series of subtasks?
2. Does the task involve high information content with complex visual displays to be interpreted, or low information content where simple signals are sufficient to alert the user?

3. Is the task intended to be performed by a layman without much training or by a practitioner skilled in the task domain?

Box 7.4 summarizes two examples to show how techniques can be chosen using these dimensions.

So, when choosing between techniques for data gathering in the requirements activity, you need to consider the nature of the technique, the knowledge required of the analyst, the nature of the task to be studied, the availability of stakeholders and other resources, and the kind of information you need.

### 7.4.3 Some basic data-gathering guidelines

Organizing your first data-gathering session may seem daunting, but if you plan the sessions well, and know what your objectives are then this will increase your confidence and make the whole exercise a lot more comfortable. Below we list some data-gathering guidelines to support the requirements activity.

- Focus on identifying the stakeholders' needs. This may be achieved by studying their existing behavior and support tools, or by looking at other products,

**Box 7.4 Coordinated Methods (Olson and Moran, 1996)**

**For a walk-up-and-use system.** An ATM is an example of a system with a simple task flow and relatively low information content that is targeted for the layman. Because of the user base, the emphasis will be on the ease with which the user can learn to operate the device. An understanding of the user's mental model may also yield insights, as evidenced by the assignment set at the end of Chapter 3.

To establish the components of the task, simple questionnaires might suffice, supplemented with naturalistic observation, i.e., observing current users at existing machines. The initial design guided by guidelines and checklists could be documented as a storyboard. A mockup of the entire system using a rapid prototyping system such as Visual Basic can be used to observe users' difficulties. After a series of such prototyping sessions, the system could be installed in a friendly site and logging data could be gathered.

**For a high-performance system.** The example used here is a system to support back-room workers at a bank who are reconciling the machine register with the information written on the back of the deposit slip by the customer. The task requires overlapping activation of physical actions and mental capabilities, is relatively high in information content, and is targeted for a skilled user.

This task is less obvious to the designer and we need to employ several techniques to understand it. If there is an existing system in place, then naturalistic observation and interview can be used. More detailed discovery of the objects, actions, and kinds of thinking can come from using interviews. Task analysis will help to understand the details of the task, and once understood, a series of design and evaluation steps follow, including prototyping, detailed analysis of the visual display and usability tests. The design would then iterate until it meets preset target criteria.
such as a competitor's product or an earlier release of your product under development.

- Involve all the stakeholder groups. It is very important to make sure that you get all the views of the right people. This may seem an obvious comment, but it is easy to overlook certain sections of the stakeholder population if you're not careful. We were told about one case where a large distribution and logistics company reimplemented their software systems and were very careful to involve all the clerical, managerial, and warehouse staff in their development process, but on the day the system went live, the productivity of the operation fell by 50%. On investigation it was found that the bottleneck was not in their own company, but in the suppliers' warehouses that had to interact with the new system. No one had asked them how they worked, and the new system was incompatible with their working routines.

- Involving only one representative from each stakeholder group is not enough, especially if the group is large. Everyone you involve in data gathering will have their own perspective on the situation, the task, their job and how others interact with them. If you only involve one representative stakeholder then you will only get a narrow view.

- Use a combination of data gathering techniques. Each technique will yield a certain kind of information, from a certain perspective. Using different techniques is one way of making sure that you get different perspectives (called triangulation, see Chapter 10), and corroboration of findings. For example, use observation to understand the context of task performance, interviews to target specific user groups, questionnaires to reach a wider population, and focus groups to build a consensus view.

- Support the data-gathering sessions with suitable props, such as task descriptions and prototypes if available. Since the requirements activity is iterative, prototypes or descriptions generated during one session may be reused or revisited in another with the same or a different set of stakeholders. Using props will help to jog people's memories and act as a focus for discussions.

- Run a pilot session if possible to ensure that your data-gathering session is likely to go as planned. This is particularly important for questionnaires where there is no one to help the users with ambiguities or other difficulties, but also applies to interview questions, workshop formats, and props. Any data collected during pilot sessions cannot be treated equally with other data, so don't mix them up. After running the pilot it is likely that some changes will be needed before running the session "for real."

- In an ideal world, you would understand what you are looking for and what kinds of analysis you want to do, and design the data-capture exercise to collect the data you want. However, data gathering is an expensive and time-consuming activity that is often tightly constrained on resources. Sometimes pragmatic constraints mean that you have to make compromises on the ideal
situation, but before you can make sensible compromises, you need to know what you'd really like.

- How you record the data during a face-to-face data-gathering session is just as important as the technique(s) you use. Video recording, audio recording, and note taking are the main options. Video and audio recording provide the most accurate record of the session, but they can generate huge amounts of data. You also need to decide on practical issues that can have profound effects on the data collected, such as where to position the camera. Note taking can be harder unless this is the person's only role in the session, but note taking always involves an element of interpretation. Taking impartial, accurate notes is difficult but can be improved with practice.

**ACTIVITY 7.2** For each of the situations below, consider what kinds of data gathering would be appropriate and how you might use the different techniques introduced above. You should assume that you are at the beginning of the development and that you have sufficient time and resources to use any of the techniques.

(a) You are developing a new software system to support a small accountant's office. There is a system running already with which the users are reasonably happy, but it is looking dated and needs upgrading.

(b) You are looking to develop an innovative device for diabetes sufferers to help them record and monitor their blood sugar levels. There are some products already on the market, but they tend to be large and unwieldy. Many diabetes sufferers rely on manual recording and monitoring methods involving a ritual with a needle, some chemicals, and a written scale.

(c) You are developing a **website** for a young person's fashion e-commerce site.

**Comment**

(a) As this is a small office, there are likely to be few stakeholders. Some period of observation is always important to understand the context of the new and the old system. Interviewing the staff rather than giving them questionnaires is likely to be appropriate because there aren't very many of them, and this will yield richer data and give the developers a chance to meet the users. Accountancy is regulated by a variety of laws and it would also pay to look at documentation to understand some of the constraints from this direction. So we would suggest a series of interviews with the main users to understand the positive and negative features of the existing system, a short observation session to understand the context of the system, and a study of documentation surrounding the regulations.

(b) In this case, your user group is spread about, so talking to all of them is infeasible. However, it is important to interview some, possibly at a local diabetic clinic, making sure that you have a representative sample. And you would need to observe the existing manual operation to understand what is required. A further group of stakeholders would be those who use or have used the other products on the market. These stakeholders can be questioned to find out the problems with the existing devices so that the new device can improve on them. A questionnaire sent to a wider group in order to back up the findings from the interviews would be appropriate, as might a focus group where possible.
(c) Again, you are not going to be able to interview all your users. In fact, the user group may not be very well defined. Interviews backed up by questionnaires and focus groups would be appropriate. Also, in this case, identifying similar or competing sites and evaluating them will help provide information for producing an improved product.

The problems of choosing among data-gathering techniques for the requirements activity have been recognized in requirements engineering. For example ACRE (Acquisition REquirements) is a quite extensive set of guidance to help requirements engineers choose between a variety of techniques for data gathering, including interviews and observation. The framework also includes other techniques from software engineering, knowledge engineering, and the social sciences. For more information on this framework, see Maiden and Rugg (1996).

7.5 Data interpretation and analysis

Once the first data-gathering session has been conducted, interpretation and analysis can begin. It's a good idea to start interpretation as soon after the gathering session as possible. The experience will be fresh in the minds of the participants and this can help overcome any bias caused by the recording approach. It is also a good idea to discuss the findings with others to get a variety of perspectives on the data.

The aim of the interpretation is to begin structuring and recording descriptions of requirements. Using a template such as the one suggested in Volere (Figure 7.5) highlights the kinds of information you should be looking for and guides the data interpretation and analysis. Note that many of the entries are concerned with trace-
ability. For example, who raised the requirement and where can more information about it be found. This information may be captured in documents or in diagrams drawn during analysis. Providing links with raw data as captured on video or audio recordings can be harder, although just as important. Haumer et al. (2000) have developed a tool that records concrete scenarios using video, speech, and graphic media, and relates these recorded observations to elements of a corresponding design. This helps designers to keep track of context and usage information while analyzing and designing for the system.

More focused analysis of the data will follow initial interpretation. Different techniques and notations exist for investigating different aspects of the system that will in turn give rise to the different requirements. For example, functional requirements have traditionally been analyzed and documented using data-flow diagrams,

![Class Diagram](image1)

![Sequence Diagram](image2)

**Figure 7.6** (a) Class diagram and (b) sequence diagram that might be used to analyze and capture static structure and dynamic behavior (respectively) if the system is being developed using an object-oriented approach.
state charts, work-flow charts, etc. (see e.g., Sommerville, 2001). Data requirements can be expressed using entity-relationship diagrams, for example. If the development is to take an object-oriented approach, then functional and data requirements are combined in class diagrams, with behavior being expressed in state charts and sequence diagrams, among others. Examples of two such diagrams representing a portion of a holiday booking system are given in Figure 7.6. These diagrams can be linked to the requirements through the "Event/use case" field in the template in Figure 7.5.

We don't go into the detail of how diagrams such as these might be developed, as whole books are dedicated to them. Instead, we describe four techniques that have a user-centered focus and are used to understand users' goals and tasks: scenarios, use cases, essential use cases, and task analysis. All of them may be produced during data-gathering sessions, and their output used as props in subsequent data-gathering sessions.

The requirements activity iterates a number of times before a set of stable requirements evolves. As more interpretation and analysis techniques are applied, a deeper understanding of requirements will emerge and the requirements descriptions will expand and clarify.
7.6 Task description

Descriptions of business tasks have been used within software development for many years. During the 1970s and 1980s, "business scenarios" were commonly used as the basis for acceptance testing, i.e., the last testing stage before the customer paid the final fee installment and "accepted" the system. In more recent years, due to the emphasis on involving users earlier in the development lifecycle and the large number of new interaction devices now being developed, task descriptions are used throughout development, from early requirements activities through prototyping, evaluation, and testing. Consequently, more time and effort has been put into understanding how best to structure and use them.

There are different flavors of task descriptions, and we shall introduce three of them here: scenarios, use cases, and essential use cases. Each of these may be used to describe either existing tasks or envisioned tasks with a new device. They are not mutually exclusive and are often used in combination to capture different perspectives or to document different stages during the development lifecycle.

In this section and the next, we use two main examples to illustrate the application of techniques. These are a library catalog service and a shared diary or calendar system. The library catalog is similar to any you might find in a public or
7.6 Task description

university library, and allows you to access the details of books held in the library: for example, to search for books by a particular author, or by subject, to identify the location of a book you want to borrow, and to check on a member's current loans and status.

The shared calendar application is to support a university department. Members of the department currently keep their own calendars and communicate their whereabouts to the department's administrator, who keeps the information in a central paper calendar. Unfortunately, the central calendar and the individuals' calendars easily become out of step as members of the department arrange their own engagements. It is hoped that having a shared calendar in which individuals can enter their own engagements will help overcome the confusion that often ensues due to this mismatch. Shared calendars raise some interesting aspects of collaboration and coordination, as discussed in Chapter 4, Box 4.2. In particular, people don't usually like to have their time filled with appointments without their consent, and so a mechanism is needed for people to protect some time from being booked by others.

7.6.1 Scenarios

A scenario is an "informal narrative description" (Carroll, 2000). It describes human activities or tasks in a story that allows exploration and discussion of contexts, needs, and requirements. It does not explicitly describe the use of software or other technological support to achieve a task. Using the vocabulary and phrasing of users means that the scenarios can be understood by the stakeholders, and they are able to participate fully in the development process. In fact, the construction of scenarios by stakeholders is often the first step in establishing requirements.

Imagine that you have just been invited along to talk to a group of users who perform data entry for a university admissions office. You walk in, and are greeted by Sandy, the supervisor, who starts by saying something like:

Well, this is where the admissions forms arrive. We receive about 50 a day during the peak application period. Brian here opens the forms and checks that they are complete, that is, that all the documentation has been included. You see, we require copies of relevant school exam results or evidence of work experience before we can process the application. Depending on the result of this initial inspection, the forms get passed to . . .

Telling stories is a natural way for people to explain what they are doing or how to achieve something. It is therefore something that stakeholders can easily relate to. The focus of such stories is also naturally likely to be about what the users are trying to achieve, i.e., their goals. Understanding why people do things as they do and what they are trying to achieve in the process allows us to concentrate on the human activity rather than interaction with technology.

This is not to say that the human activity should be preserved and reflected in any new device we are trying to develop, but understanding what people do now is a good starting point for exploring the constraints, contexts, irritations, facilitators and so on under which the humans operate. It also allows us to identify the stakeholders and the products involved in the activity. Repeated reference to a particular
form, book, behavior, or location indicates that this is somehow central to the activity being performed and that we should take care to understand what it is and the role it plays.

A scenario that might be generated by potential users of a library catalog service is given below:

_Say I want to find a book by George Jeffries. I don't remember the title but I know it was published before 1995. I go to the catalog and enter my user password. I don't understand why I have to do this, since I can't get into the library to use the catalog without passing through security gates. However, once my password has been confirmed, I am given a choice of searching by author or by date, but not the combination of author and date. I tend to choose the author option because the date search usually identifies too many entries. After about 30 seconds the catalog returns saying that there are no entries for George Jeffries and showing me the list of entries closest to the one I've sought. When I see the list, I realize that in fact I got the author's first name wrong and it's Gregory, not George. I choose the entry I want and the system displays the location to tell me where to find the book._

In this limited scenario of existing system use, there are some things of note: the importance of getting the author's name right, the annoyance concerning the need to enter a password, the lack of flexible search possibilities, and the usefulness of showing a list of similar entries when an exact match isn't clear. These are all indicators of potential design choices for the new catalog system. The scenario also tells us one (possibly common) use of the catalog system: to search for a book by an author when we don't know the title.

The level of detail present in a scenario varies, and there is no particular guidance about how much or how little should be included. Often scenarios are generated during workshop or interview sessions to help explain or discuss some aspect of the user's goals. They can be used to imagine potential uses of a device as well as to capture existing behavior. They are not intended to capture a full set of requirements, but are a very personalized account, offering only one perspective.

A simple scenario for the shared-calendar system that was elicited in an informal interview describes how one function of the calendar might work: to arrange a meeting between several people.

_The user types in all the names of the meeting participants together with some constraints such as the length of the meeting, roughly when the meeting needs to take place, and possibly where it needs to take place. The system then checks against the individuals' calendars and the central departmental calendar and presents the user with a series of dates on which everyone is free all at the same time. Then the meeting could be confirmed and written into peoples' calendars. Some people, though, will want to be asked before the calendar entry is made. Perhaps the system could email them automatically and ask that it be confirmed before it is written in._

An example of a futuristic scenario, devised by Symbian, showing one vision of how wireless devices might be used in the future is shown in Figure 7.7.

In this chapter, we refer to scenarios only in their role of helping to establish requirements. They have a continuing role in the design process that we shall return to in Chapter 8.
A businesswoman traveling to Paris from the US

A businesswoman is traveling from San Francisco to Paris on a business trip. On her way to the airport she narrowly misses a traffic delay. She avoids the traffic jam because her Smartphone beeps, then sends her a text message warning her of the traffic accident on her normal route from her office to the airport.

Upon arrival at the airport, the location-sensitive Smartphone notifies the airline that she will be checking in shortly, and an airline employee immediately finds her and takes her baggage. Her on-screen display shows that her flight is on time and provides a map to her gate. On her way to the gate she downloads tourist information such as maps and events occurring in Paris during her stay.

Once she finds her seat on the plane, she begins to review all the information she has downloaded. She notices than an opera is playing in Paris that she has been wanting to see, and she books her ticket. Her Smartphone can make the booking using her credit card number, which it has stored in its memory. This means that she does not need to re-enter the credit card number each time she uses wCommerce (i.e., wireless commerce), facilities. The security written into the software of the Smartphone protects her against fraud.

The Smartphone stores the opera booking along with several emails that she writes on the plane. As soon as she steps off the plane, the Smartphone makes the calls and automatically sends the emails.

As she leaves the airport, a map appears on her Smartphone's display, guiding her to her hotel.

Figure 7.7 A scenario showing how two technologies, a Smartphone and wCommerce (wireless commerce), might be used.

Capturing scenarios of existing behavior and goals helps in determining new scenarios and hence in gathering data useful for establishing the new requirements. The next activity is intended to help you appreciate how a scenario of existing activity can help identify the requirements for a future application to support the same user goal.

**Activity 7.3** Write a scenario of how you would currently go about choosing a new car. This should be a brand new car, not a second-hand car. Having written it, think about the important aspects of the task, your priorities and preferences. Then imagine a new interactive product that supports you in your goal and takes account of these issues. Write a futuristic scenario showing how this product would support you.

**Comment** The following example is a fairly generic view of this process. Yours will be different, but you may have identified similar concerns and priorities.

The first thing I would do is to observe cars on the road and identify ones that I like the look of. This may take some weeks. I would also try to identify any consumer reports that will include an assessment of car performance. Hopefully, these initial activities will result in me identifying a likely car to buy. The next stage will be to visit a car showroom and see at first hand what the car looks like, and how comfortable it is to sit in. If I still feel positive about the car, then I’ll ask for a test drive. Even a short test drive helps me to
understand how well the car handles, how noisy is the engine, how smooth are the gear changes, and so on. Once I've driven the car myself, I can usually tell whether I would like to own it or not.

From this scenario, it seems that there are broadly two stages involved in the task: researching the different cars available, and gaining first-hand experience of potential purchases. In the former, observing cars on the road and getting actual and maybe critical information about them has been highlighted. In the latter, the test drive seems to be quite significant.

For many people buying a new car, the smell and touch of the car's exterior and interior, and the driving experience itself are often the most influential factors in choosing a particular model. Other more factual attributes such as fuel consumption, amount of room inside, colors available, and price may rule out certain makes and models, but at the end of the day, cars are often chosen according to how easy they are to handle and how comfortable they are inside. This makes the test drive a vital part of the process of choosing a new car.

Taking these comments into account, we've come up with the following scenario describing how a new "one-stop" shop for new cars might operate. This product makes use of immersive virtual reality technology that is already used for other applications such as designing buildings and training bomb disposal experts.

I want to buy a new car, so I go down the street to the local "one-stop car shop." The shop has a number of booths in it, and when I go in I'm directed to an empty booth. Inside there's a large seat that reminds me of a racing car seat, and in front of that a large display screen, keyboard and printer. As I sit down, the display jumps into life. It offers me the options of browsing through video clips of new cars which have been released in the last two years, or of searching through video clips of cars by make, by model, or by year. I can choose as many of these as I like. I also have the option of searching through and reading or printing consumer reports that have been produced about the cars I'm interested in. I spend about an hour looking through materials and deciding that I'd like to experience a couple that look promising. I can of course go away and come back later, but I'd like to have a go with some of those I've found. By flicking a switch in my armrest, Z can call up the options for virtual reality simulations for any of the cars I'm interested in. These are really great as they allow me to take the car for a test drive, simulating everything about the driving experience in this car, from road holding, to windscreen display, and front pedal pressure to dash board layout. It even re-creates the atmosphere of being inside the car.

Note that the product includes support for the two research activities mentioned in the original scenario, as well as the important test drive facility. This would be only a first cut scenario which would then be refined through discussion and further investigation.

7.6.2 Use cases

Use cases also focus on user goals, but the emphasis here is on a user–system interaction rather than the user's task itself. They were originally introduced through the object-oriented community in the book *Object-Oriented Software Engineering* (Jacobson et al., 1992). Although their focus is specifically on the interaction between the user (called an "actor") and a software system, the stress is still very much on the user's perspective, not the system's. The term "scenario" is also used in the context of use cases. In this context, it represents one path through the use
case, i.e., one particular set of conditions. This meaning is consistent with the definition given above in that they both represent one specific example of behavior.

A use case is associated with an actor, and it is the actor's goal in using the system that the use case wants to capture. In this technique, the main use case describes what is called the "normal course" through the use case, i.e., the set of actions that the analyst believes to be most commonly performed. So, for example, if through data gathering we have found that most users of the library go to the catalog to check the location of a book before going to the shelves, then the normal course for the use case would include this sequence of events. Other possible sequences, called alternative courses, are then listed at the bottom of the use case.

A use case for arranging a meeting using the shared calendar application, with the normal course being that the meeting is written into the calendar automatically, might be:

1. The user chooses the option to arrange a meeting.
2. The system prompts user for the names of attendees.
3. The user types in a list of names.
4. The system checks that the list is valid.
5. The system prompts the user for meeting constraints.
6. The user types in meeting constraints.
7. The system searches the calendars for a date that satisfies the constraints.
8. The system displays a list of potential dates.
9. The user chooses one of the dates.
10. The system writes the meeting into the calendar.
11. The system **emails** all the meeting participants informing them of the appointment.

Alternative courses:

5. If the list of people is invalid,
   5.1 The system displays an error message.
   5.2 The system returns to step 2.
8. If no potential dates are found,
   8.1 The system displays a suitable message.
   8.2 The system returns to step 5.

Note that the number associated with the alternative course indicates the step in the normal course that is replaced by this action or set of actions. Also note how specific the use case is about how the user and the system will interact.

Use cases may be described graphically. Figure 7.8 shows the use case diagram for the above calendar example. The actor "Administrator" is associated with the use case "Arrange a meeting." Another actor we might identify for the calendar system is the "Departmental member" who updates his own calendar entries, also shown in Figure 7.8. Actors may be associated with more than one use case, so for
example the actor "Departmental member" can be associated with a use case "Retrieve contact details" as well as the "Update calendar entry" use case. Each use case may also be associated with more than one actor.

This kind of description has a different style and a different focus from the scenarios described above. The layout is more formal, and the structure of "good" use cases has been discussed by many (e.g., Cockburn, 1995; Gough et al., 1995; Ben Achour, 1999). The description also focuses on the user–system interaction rather than on the user's activities; thus a use case presupposes that technology is being used. This kind of detail is more useful at conceptual design stage than during requirements or data gathering, but use cases have been found to help some stakeholders express their views on how existing systems are used and how a new system might work.

To develop a use case, first identify the actors, i.e., the people or other systems that will be interacting with the system under development. Then examine these actors and identify their goal or goals in using the system. Each of these will be a use case.
Consider the example of the library catalog service again. One use case is "Locate book," and this would be associated with the "Library member" actor. Identify one other main actor and an associated use case, and draw a use case diagram.

Write out the use case for "Locate book" including the normal and some alternative courses. You may assume that the normal course is for users to go to the catalog to find the location, and that the most common path to find this is through a search by author.

Comment

One other main actor is the "Librarian." A use case for the "Librarian" would be "Update catalog." Figure 7.9 is the associated use case diagram. There are other use cases you may have identified.

The use case for "Locate book" might be something like this:

1. The system prompts for user name and password.
2. The user enters his or her user name and password into the catalog system.
3. The system verifies the user's password.
4. The system displays a menu of choices.
5. The user chooses the search option.
6. The system displays the search menu.
7. The user chooses to search by author.
8. The system displays the search author screen.
9. The user enters the author's name.
10. The system displays search results.
11. The user chooses the required book.
12. The system displays details of chosen book.
13. The user notes location.
14. The user quits catalog system.

Alternative courses:

4. If user password is not valid
   4.1 The system displays error message.
   4.2 The system returns to step 1.
5. If user knows the book details
   5.1 The user chooses to enter book details.
   5.2 The system displays book details screen.
   5.3 The user enters book details.
   5.4 The system goes to step 12.

7.6.3 Essential use cases

Essential use cases were developed by Constantine and Lockwood (1999) to combat what they see as the limitations of both scenarios and use cases as described
Chapter 7  Identifying needs and establishing requirements

Figure 7.10  An essential use case for arranging a meeting in the shared calendar application.

above. Scenarios are concrete stories that concentrate on realistic and specific activities. They therefore can obscure broader issues concerned with the wider organizational view. On the other hand, traditional use cases contain certain assumptions, including the fact that there is a piece of technology to interact with, and also assumptions about the user interface and the kind of interaction to be designed.

Essential use cases represent abstractions from scenarios, i.e., they represent a more general case than a scenario embodies, and try to avoid the assumptions of a traditional use case. An essential use case is a structured narrative consisting of three parts: a name that expresses the overall user intention, a stepped description of user actions, and a stepped description of system responsibility. This division between user and system responsibilities can be very helpful during conceptual design when considering task allocation and system scope, i.e., what the user is responsible for and what the system is to do.

An example essential use case based on the library example given above is shown in Figure 7.10. Note that the steps are more generalized than those in the use case in Section 7.6.2, while they are more structured than the scenario in Section 7.6.1. For example, the first user intention does not say anything about typing in a list of names, it simply states that the user identifies meeting attendees. This could be done by identifying roles, rather than people’s names, from an organizational or project chart, or by choosing names from a list of people whose calendars the system keeps, or by typing in the names. The point is that at the time of creating this essential use case, there is no commitment to a particular interaction design.

Instead of actors, essential use cases are associated with user roles. One of the differences is that an actor could be another system, whereas a user role is just that: not a particular person, and not another system, but a role that a number of different people may play when using the system. Just as with actors, though, producing an essential use case begins with identifying user roles.

ACTIVITY 7.5  Construct an essential use case “locateBook” for the user role “Library member” of the library catalog service discussed in Activity 7.4.
### 7.7 Task analysis

Task analysis is used mainly to investigate an existing situation, not to envision new systems or devices. It is used to analyze the underlying rationale and purpose of what people are doing: what are they trying to achieve, why are they trying to achieve it, and how are they going about it? The information gleaned from task analysis establishes a foundation of existing practices on which to build new requirements or to design new tasks.

**Task analysis** is an umbrella term that covers techniques for investigating cognitive processes and physical actions, at a high level of abstraction and in minute detail. In practice, task analysis techniques have had a mixed reception. The most widely used version is Hierarchical Task Analysis (HTA) and this is the technique we introduce in this chapter. Another well-known task analysis technique called GOMS (goals, operations, methods, and selection rules) that models procedural knowledge (Card et al., 1983) is described in Chapter 14.

#### 7.7.1 Hierarchical task analysis

Hierarchical Task Analysis (HTA) was originally designed to identify training needs (Annett and Duncan, 1967). It involves breaking a task down into subtasks and then into sub-subtasks and so on. These are then grouped together as plans that specify how the tasks might be performed in an actual situation. HTA focuses on the physical and observable actions that are performed, and includes looking at actions that are not related to software or an interaction device at all. The starting point is a user goal. This is then examined and the main tasks associated with achieving that goal are identified. Where appropriate, these tasks are subdivided into subtasks.

Consider the library catalog service, and the task of borrowing a book. This task can be decomposed into other tasks such as accessing the library catalog, searching by name, title, subject, or whatever, making a note of the location of the book, going to the correct shelf, taking it down off the shelf (provided it is there) and finally tak-
0. In order to borrow a book from the library
1. go to the library
2. find the required book
   2.1 access library catalog
   2.2 access the search screen
   2.3 enter search criteria
   2.4 identify required book
   2.5 note location
3. go to correct shelf and retrieve book
4. take book to checkout counter

plan 0: do 1-3-4. If book isn't on the shelf expected, do 2-3-4.
plan 2: do 2.1-2.4-2.5. If book not identified do 2.2-2.3-2.4-2.5.

Figure 7.11 An HTA for borrowing a book from the library.

it to the check-out counter. This set of tasks and subtasks might be performed in a
different order depending on how much is known about the book, and how familiar
the user might be with the library and the book's likely location. Figure 7.11 shows
these subtasks and some plans for different paths through those subtasks. Indentation
shows the hierarchical relationship between tasks and subtasks.

Note how the numbering works for the task analysis: the number of the plan
corresponds to the number of the step to which the plan relates. For example, plan
2 shows how the subtasks in step 2 can be ordered; there is no plan 1 because step 1
has no subtasks associated with it.

An alternative expression of an HTA is a graphical box-and-line notation. Figure
7.12 shows the graphical version of the HTA in Figure 7.11. Here the subtasks
are represented by named boxes with identifying numbers. The hierarchical relation-
ship between tasks is shown using a vertical line. If a task is not decomposed
any further then a thick horizontal line is drawn underneath the corresponding box.

Figure 7.12 A graphical representation of the task analysis for borrowing a book.
Plans are also shown in this graphical form. They are written alongside the vertical line emitting from the task being decomposed. For example, in Figure 7.12 plan 2 is specified next to the vertical line from box 2 "find required book."

ACTIVITY 7.6

Look back at the scenario for arranging a meeting in the shared calendar application. Perform hierarchical task analysis for the goal of arranging a meeting. Include all plans in your answer. Express the task analysis textually and graphically.

Comment

The main tasks involved in this are to find out who needs to be at the meeting, find out the constraints on the meeting such as length of meeting, range of dates, and location, find a suitable date, enter details into the calendar, and inform attendees. Finding a suitable date can be decomposed into other tasks such as looking in the departmental calendar, looking in individuals' calendars, and checking potential dates against constraints. The textual version of the HTA is shown below. Figure 7.13 shows the corresponding graphical representation.

0. In order to arrange a meeting
   1. compile a list of meeting attendees
   2. compile a list of meeting constraints
   3. find a suitable date
      3.1 identify potential dates from departmental calendar
      3.2 identify potential dates from each individual's calendar
      3.3 compare potential dates
      3.4 choose one preferred date
   4. enter meeting into calendars
   5. inform meeting participants of calendar entry

plan 0: do 1-2-3. If potential dates are identified, do 4-5. If no potential dates can be identified, repeat 2-3.

plan 3: do 3.1-3.2-3.3-3.4 or do 3.2-3.1-3.3-3.4

Figure 7.13 A graphical representation of the meeting HTA.
What do you think are the main problems with using task analysis on real problems? Think of more complex tasks such as scheduling delivery trucks, or organizing a large conference.

Comment

Real tasks are very complex. One of the main problems with task analysis is that it does not scale very well. The notation soon becomes unwieldy, making it difficult to follow. Imagine what it would be like to produce a task analysis in which there were hundreds or even thousands of subtasks.

A second problem is that task analysis is limited in the kind of tasks it can model. For example, it cannot model tasks that are overlapping or parallel, nor can it model interruptions. Most people work through interruptions of various kinds, and many significant tasks happen in parallel.

Assignment

This assignment is the first of four assignments that together take you through the complete development lifecycle for an interactive product. This assignment requires you to use techniques described in this chapter for identifying needs and establishing requirements. The further three assignments are at the end of Chapters 8, 13, and 14.

The overall assignment is for you to design and evaluate an interactive website for booking tickets online for events like concerts, the theatre and the cinema. This is currently an activity that in many instances, can be difficult or inconvenient to achieve using traditional means (e.g., waiting for ages on the phone to get hold of an agent, queuing for hours in the rain at a ticket office).

For this assignment, you should:

(a) Identify users' needs for this website. You could do this in a number of ways. For example, you could observe people using ticket agents, think about your own experience of purchasing tickets, look at existing websites for booking tickets, talk to friends and family about their experiences, and so on. Record your data carefully.

(b) Based on your user requirements, choose two different user profiles and produce one main scenario for each one, capturing how the user is expected to interact with the system.

(c) Using the scenarios generated from your data gathering, perform a task analysis on the main task associated with the ticket booking system, i.e., booking a ticket.

(d) Based on the data gathered in part (a) and your subsequent interpretation and analysis, identify different kinds of requirements for the website, according to the headings introduced in Section 7.3 above. Write up the requirements in the style of the Volere template.

Summary

In this chapter, we have looked in more detail at how to identify users' needs and establish requirements for interaction design. Various data-gathering techniques can be used to collect data for interpretation and analysis. The most common of these are questionnaires, interviews, focus groups, workshops, naturalistic observation, and studying documentation. Each of these has advantages and disadvantages that must be balanced against your constraints when choosing which techniques to use for a particular project. They can be combined in many different ways, and can be supported by props such as scenarios and prototypes. How
to carry out these techniques is covered in Chapters 12 through 14. Scenarios, use cases, and essential use cases are helpful techniques for beginning to document the findings from the data-gathering sessions. Task analysis is a little more structured, but does not scale well.

**Key points**
- Getting the requirements right is crucial to the success of the interactive product.
- There are different kinds of requirements: functional, data, environmental, user, and usability. Every system will have requirements under each of these headings.
- The most commonly used data-gathering techniques for this activity are: questionnaires, interviews, workshops or focus groups, naturalistic observation, and studying documentation.
- Descriptions of user tasks such as scenarios, use cases, and essential use cases help users to articulate existing work practices. They also help to express envisioned use for new devices.
- Task analysis techniques help to investigate existing systems and current practices.

**Further reading**

ROBERTSON, SUZANNE, AND ROBERTSON, JAMES (1999) *Mastering the Requirements Process*. Boston: Addison-Wesley. In this book, Robertson and Robertson explain a useful framework for software requirements work (see also the interview with Suzanne Robertson after this chapter).

CONSTANTINE, LARRY L., AND LOCKWOOD, LUCY A. D. (1999) *Software for Use*. Boston: Addison-Wesley. This very readable book provides a concrete approach for modeling and analyzing software systems. The approach has a user-centered focus and contains some useful detail. It also includes more information about essential use cases.

JACOBSON, I., BOOCH, G., AND RUMBAUGH, J. (1992) *The Unified Software Development Process*. Boston: Addison-Wesley. This is not an easy book to read, but it is the definitive guide for developing object-oriented systems using use cases and the modeling language Unified Modeling Language (UML).


SOMMERVILLE, IAN (2001) *Software Engineering* (6th ed.). Boston: Addison-Wesley. If you are interested in pursuing notations for functional and data requirements, then this book introduces a variety of notations and techniques used in software engineering.
HS: What are requirements?
SR: Well the problem is that "requirements" has turned into an elastic term. Requirements is an enormously wide field and there are so many different types of requirements. One person may be talking about budget, somebody else may be talking about interfacing to an existing piece of software, somebody else may be talking about a performance requirement, somebody else may be talking about the calculation of an algorithm, somebody else may be talking about a data definition, and I could go on for hours as to what requirement means. What we advise people to do to start with is to look for something we call "linguistic integrity" within their own project. When all people who are connected with the project are talking about requirements, what do they mean? This gets very emotional, and that's why we came up with our framework. We gathered together all this experience of different types of requirements, tried to pick the most common organization, and then wrote them down in a framework.

HS: Please would you explain your framework? (The version discussed in this interview is shown in the figure on page 238. The most recent version may be downloaded from www.systemsguild.com.)
SR: Imagine a huge filing cabinet with 27 drawers, and in each drawer you've got a category of knowledge that is related to requirements. In the very first drawer for example you've got the goals, i.e., the reason for doing the project. In the second drawer you've got the stakeholders. These are roles because they could be played by more than one person, and one person may play more than one role. You've got the client who's going to pay for the development, and the customer who's making the decision about buying it. Then you've got stakeholders like the project leader, the developers, the requirements engineers, the designers, the quality people, and the testers. Then you've got the less obvious stakeholders like surrounding organizations, professional bodies, and other people in the organization whose work might be affected by the project you're doing, even if they're never going to use the product.

HS: So do you find the stakeholders by just asking questions?
SR: Yes, partly that and partly by using the domain model of the subject matter, which is in drawer 9, as the driver to ask more questions about the stakeholders. For example, for each one of the subject matter areas, ask who have we got to represent this subject matter? For each one of the people that we come across, ask what subject matter are we expecting from them? Drawer 3 contains the end users. I've put them in a separate drawer because an error that a lot of people make when they're looking for requirements is that the only stakeholder they talk about is the end user. They decide on the end user too quickly and they miss opportunities. So you end up building a product that is possibly less competitive. I keep them a bit fuzzy to start with, and as you start to fix on them then you can go into really deep analysis about them: What is their psychology? What are their characteristics? What's their subject-matter knowledge? How do they feel about their work? How do they feel about technology? All of these things help you to come up with the most competitive non-functional requirements for the product.

HS: How do you resolve conflict between stakeholders?
SR: Well, part of it is to get the conflicts out in the open up front, so people stop blaming each other, but that certainly doesn't resolve it. One of the ways is to make things very visible all the way through and to keep reminding people that conflict is respectable, that it's a sign of creativity, of people having ideas. The other thing that we do is that in our individual requirements (that is atomic requirements), which end up living in drawers 9 to 17 of this filing cabinet, we've got a place to say "Conflict: Which other requirement is this in conflict with?" and we encourage people to
identify them. Sometimes these conflicts resolve themselves because they’re on people’s back burners, and some of the conflicts are resolved by people just talking to one another. We have a point at which we cross-check requirements and look for conflicts and if we find some that are just not sorting themselves out, then we stop and have a serious negotiation.

In essence, it's bubbling the conflicts up to the surface. Keep on talking about them and keep them visible. De-personalize it as much as you can. That helps.

**HS:** What other things are associated with these atomic requirements?

**SR.** Each one has a unique number and a description that is as close as you can get to what you think the thing means. It also has a rationale that helps you to figure out what it really is. Then the next component is the fit criterion, which is, "If somebody came up with a solution to this requirement, how would you know whether or not it satisfies the requirement?" So this means making the requirement quantifiable, measurable. And it’s very powerful because it makes you think about the requirement. One requirement quite often turns into several when you really try and quantify it. It also provides a wonderful opportunity for involving testers, because at that point if you write the fit criterion you can get a tester and ask whether this can be used as input to writing a cost-effective test. Now this is different from the way we usually use the testers, which is to build tests that test our solutions. Here I want to get them in much earlier, I want them to test whether this requirement really is a requirement.

**HS:** So what's in drawers 18 through 27?

**SR.** Well here you can get into serious quarrels. The overall category is "project issues," and people often say they're not really requirements, and they aren't. But if the project is not being managed according to the real work that's being done, in other words the contents of the drawers, then the project goes off the rails. In project issues we create links so that a project manager can manage the project according to what's happening to the requirements.

In the last drawer we have design ideas. People say when you're gathering requirements you should not be concerned with how you're going to solve the problem. But mostly people tell you requirements in the form of a solution anyway. The key thing is to learn how to separate the real requirements from solution ideas, and when you get a solution idea, pop it in this drawer. This helps requirements engineers, I think, because we are trained to think of solutions, not to dig behind and find the real problem.

**HS:** How do you go about identifying requirements?

**SR.** For too long we've been saying the stakeholders should give us their requirements: we'll ask them and they'll give them to us. We've realized that this is not practical—partly because there are many requirements people don't know they've got. Some requirements are conscious and they're usually because things have gone wrong or they'd like something extra. Some requirements are unconscious because maybe people are used to it, or maybe they haven't a clue because they don't see the overall picture. And then there are undreamed-of requirements that people just don't dream they could ever have, because we've all got boundaries based on what we think technology is capable of doing or what we know about technology or what our experience is. So it's not just asking people for things, it's also inventing requirements. I think that's where prototyping comes in and scenario modeling and storyboarding and all of those sorts of techniques to help people to imagine what they could have.

If you're building a product for the market and you want to be more competitive you should be inventing requirements. Instead of constraining yourself within the product boundary, say, "Can I push myself out a bit further? Is there something else I could do that isn't being done?"

**HS:** So what kinds of techniques can people use to push out further?

**SR.** One of the things is to learn how to imagine what it's like to be somebody else, and this is why going into other fields, for example family therapy, is helpful. They've learned an awful lot about how to imagine you might be somebody else. And that's not something that software engineers are taught in college normally and this is why it's very healthy for us to be bringing together the ideas of psychology and sociology and so on with software and systems engineering. Bringing in these human aspects—the performance, the usability features, the "look and feel" features—that's going to make our products more competitive. I always tell people to read a lot of novels. If you're having trouble relating to some stakeholders, for example, go and read some Jane Austen and then try to
imagine what it would have been like to have been the heroine in *Pride and Prejudice*. What would it have been like to have to change your clothes three times a day? I find this helps me a lot, it frees your mind and then you can say, "OK, what's it really like to be that other person?" There's a lot to learn in that area.

**HS:** So what you're saying really is that it's not easy.

**SR.** It's not easy. I don't think there's any particular technique. But what we have done is we have come up with a lot of different "trawling" techniques, along with recommendations, that can help you.

**HS:** Do you have any other tips for gathering requirements?

**SR:** It's important for people to feel that they've been heard. The waiting room (drawer number 26) was invented because of a very enthusiastic high-level stakeholder in a project we were doing. She was very enthusiastic and keen and very involved. Wonderful! She really gave us tremendous ideas and support. The problem was she kept having ideas, and we didn't know what to do. We didn't want to stop her having ideas, on the other hand we couldn't always include them because then we would never get anything built. So we invented the waiting room. All the good ideas we have put in there and every so often we go into the waiting room and review the ideas. Some of them get added to the product, some are discarded, and some are left waiting. The psychology of it is very good because the idea's in the waiting room, everyone knows it's in there, but it's not being ignored. When people feel heard, they feel better and consequently they're more likely to cooperate and give you time.

---

**The Template**

**PROJECT DRIVERS**
1. The Purpose of the Product
2. Client, Customer and other Stakeholders
3. Users of the Product

**PROJECT CONSTRAINTS**
4. Mandated Constraints
5. Naming Conventions and Definitions
6. Relevant Facts and Assumptions

**FUNCTIONAL REQUIREMENTS**
7. The Scope of the Work
8. The Scope of the Product
9. Functional and Data Requirements

**NON-FUNCTIONAL REQUIREMENTS**
10. Look and Feel Requirements
11. Usability Requirements
12. Performance Requirements
13. Operational Requirements
14. Maintainability and Portability Requirements
15. Security Requirements
16. Cultural and Political Requirements
17. Legal Requirements

**PROJECT ISSUES**
18. Open Issues
19. Off-the-shelf Solutions
20. New Problems
21. Tasks
22. Cutover
23. Risks
24. Costs
25. User Documentation and Training
26. Waiting Room
27. Ideas for Solutions

Chapter 8

Design, prototyping and construction

8.1 Introduction

8.2 Prototyping and construction
   8.2.1 What is a prototype?
   8.2.2 Why prototype?
   8.2.3 Low-fidelity prototyping
   8.2.4 High-fidelity prototyping
   8.2.5 Compromises in prototyping
   8.2.6 Construction: from design to implementation

8.3 Conceptual design: moving from requirements to first design
   8.3.1 Three perspectives for developing a conceptual model
   8.3.2 Expanding the conceptual model
   8.3.3 Using scenarios in conceptual design
   8.3.4 Using prototypes in conceptual design

8.4 Physical design: getting concrete
   8.4.1 Guidelines for physical design
   8.4.2 Different kinds of widget

8.5 Tool support

8.1 Introduction

Design activities begin once a set of requirements has been established. Broadly speaking, there are two types of design: conceptual and physical. The former is concerned with developing a conceptual model that captures what the product will do and how it will behave, while the latter is concerned with details of the design such as screen and menu structures, icons, and graphics. The design emerges iteratively, through repeated design-evaluation-redesign cycles involving users.

For users to effectively evaluate the design of an interactive product, designers must produce an interactive version of their ideas. In the early stages of development, these interactive versions may be made of paper and cardboard, while as design progresses and ideas become more detailed, they may be polished pieces of software, metal, or plastic that resemble the final product. We have
called the activity concerned with building this interactive version prototyping and construction.

There are two distinct circumstances for design: one where you’re starting from scratch and one where you’re modifying an existing product. A lot of design comes from the latter, and it may be tempting to think that additional features can be added, or existing ones tweaked, without extensive investigation, prototyping or evaluation. It is true that if changes are not significant then the prototyping and evaluation activities can be scaled down, but they are still invaluable activities that should not be skipped.

In Chapter 7, we discussed some ways to identify user needs and establish requirements. In this chapter, we look at the activities involved in progressing a set of requirements through the cycles of prototyping to construction. We begin by explaining the role and techniques of prototyping and then explain how prototypes may be used in the design process. Tool support plays an important part in development, but tool support changes so rapidly in this area that we do not attempt to provide a catalog of current support. Instead, we discuss the kinds of tools that may be of help and categories of tools that have been suggested.

The main aims of this chapter are to:

- Describe prototyping and different types of prototyping activities.
- Enable you to produce a simple prototype.
- Enable you to produce a conceptual model for a system and justify your choices.
- Enable you to attempt some aspects of physical design.
- Explain the use of scenarios and prototypes in conceptual design.
- Discuss standards, guidelines, and rules available to help interaction designers.
- Discuss the range of tool support available for interaction design.

8.2 Prototyping and construction

It is often said that users can't tell you what they want, but when they see something and get to use it, they soon know what they don't want. Having collected information about work practices and views about what a system should and shouldn't do, we then need to try out our ideas by building prototypes and iterating through several versions. And the more iterations, the better the final product will be.

8.2.1 What is a prototype?

When you hear the term prototype, you may imagine something like a scale model of a building or a bridge, or maybe a piece of software that crashes every few minutes. But a prototype can also be a paper-based outline of a screen or set of screens, an electronic "picture," a video simulation of a task, a three-dimensional paper and cardboard mockup of a whole workstation, or a simple stack of hyperlinked screen shots, among other things.
8.2 Prototyping and construction

In fact, a prototype can be anything from a paper-based storyboard through to a complex piece of software, and from a cardboard mockup to a molded or pressed piece of metal. A prototype allows stakeholders to interact with an envisioned product, to gain some experience of using it in a realistic setting, and to explore imagined uses.

For example, when the idea for the PalmPilot was being developed, Jeff Hawkins (founder of the company) carved up a piece of wood about the size and shape of the device he had imagined. He used to carry this piece of wood around with him and pretend to enter information into it, just to see what it would be like to own such a device (Bergman and Haitani, 2000). This is an example of a very simple (some might even say bizarre) prototype, but it served its purpose of simulating scenarios of use.

Ehn and Kyng (1991) report on the use of a cardboard box with the label "Desktop Laser Printer" as a mockup. It did not matter that, in their setup, the printer was not real. The important point was that the intended users, journalists and typographers, could experience and envision what it would be like to have one of these machines on their desks. This may seem a little extreme, but in 1982 when this was done, desktop laser printers were expensive items of equipment and were not a common sight around the office.

So a prototype is a limited representation of a design that allows users to interact with it and to explore its suitability.

8.2.2 Why prototype?

Prototypes are a useful aid when discussing ideas with stakeholders; they are a communication device among team members, and are an effective way to test out ideas for yourself. The activity of building prototypes encourages reflection in design, as described by Schon (1983) and as recognized by designers from many disciplines as an important aspect of the design process. Liddle (1996), talking about software design, recommends that prototyping should always precede any writing of code.

Prototypes answer questions and support designers in choosing between alternatives. Hence, they serve a variety of purposes: for example, to test out the technical feasibility of an idea, to clarify some vague requirements, to do some user testing and evaluation, or to check that a certain design direction is compatible with the rest of the system development. Which of these is your purpose will influence the kind of prototype you build. So, for example, if you are trying to clarify how users might perform a set of tasks and whether your proposed device would support them in this, you might produce a paper-based mockup. Figure 8.1 shows a paper-based prototype of the design for a handheld device to help an autistic child communicate. This prototype shows the intended functions and buttons, their positioning and labeling, and the overall shape of the device, but none of the buttons actually work. This kind of prototype is sufficient to investigate scenarios of use and to decide, for example, whether the buttons are appropriate and the functions sufficient, but not to test whether the speech is loud enough or the response fast enough.
Durable **case**—the tough plastic exterior enables complete protection of the device if dropped, and the rubberized outer casing lessens the impacts of shocks. In addition, the exterior is lightweight and makes the design ideal for use in virtually any environment.

Battery indicator shows amount of battery left before recharging is required.

**Communication keys**—these are sensitive touch-panel buttons. On being triggered, a recorded message related to that key is output from the speaker.

In addition, symbols and photos familiar to the user can be used on the keypads to enable usability of device to be immediate in the case of some individuals.

**Amplified speaker** provides excellent output.

**Ring attachment for belt/trousers**. This enables the device to hang from a person's trouser/belt in a similar way to a key ring.

Heather Martin and Bill Gaver (2000) describe a different kind of prototyping with a different purpose. When prototyping audiophotography products, they used a variety of different techniques including video scenarios similar to the scenarios we introduced in Chapter 7, but filmed rather than written. At each stage, the prototypes were minimally specified, deliberately leaving some aspects vague so as to stimulate further ideas and discussion.
8.2.3 Low-fidelity prototyping

A low-fidelity prototype is one that does not look very much like the final product. For example, it uses materials that are very different from the intended final version, such as paper and cardboard rather than electronic screens and metal. The lump of wood used to prototype the Palm Pilot described above is a low-fidelity prototype, as is the cardboard-box laser printer.

Low-fidelity prototypes are useful because they tend to be simple, cheap, and quick to produce. This also means that they are simple, cheap, and quick to modify so they support the exploration of alternative designs and ideas. This is particularly important in early stages of development, during conceptual design for example, because prototypes that are used for exploring ideas should be flexible and encourage rather than discourage exploration and modification. Low-fidelity prototypes are never intended to be kept and integrated into the final product. They are for exploration only.

Storyboarding Storyboarding is one example of low-fidelity prototyping that is often used in conjunction with scenarios, as described in Chapter 7. A storyboard consists of a series of sketches showing how a user might progress through a task using the device being developed. It can be a series of sketched screens for a GUI-based software system, or a series of scene sketches showing how a user can perform a task using the device. When used in conjunction with a scenario, the storyboard brings more detail to the written scenario and offers stakeholders a chance to role-play with the prototype, interacting with it by stepping through the scenario. The example storyboard shown in Figure 8.2 (Hartfield and Winograd,
1996) depicts a person using a new system for digitizing images. This example doesn't show detailed drawings of the screens involved, but it describes the steps a user might go through in order to use the system.

**Sketching** Low-fidelity prototyping often relies on sketching, and many people find it difficult to engage in this activity because they are inhibited about the quality of their drawing. Verplank (1989) suggests that you can teach yourself to get over this inhibition. He suggests that you should devise your own symbols and icons for elements you might want to sketch, and practice using them. They don't have to be anything more than simple boxes, stick figures, and stars. Elements you might require in a storyboard sketch, for example, include "things" such as people, parts of a computer, desks, books, etc., and actions such as give, find, transfer, and write. If you are sketching an interface design, then you might need to draw various icons, dialog boxes, and so on. Some simple examples are shown in Figure 8.3. Try copying these and using them. The next activity requires other sketching symbols, but they can still be drawn quite simply.

**ACTIVITY 8.1** Produce a storyboard that depicts how to fill a car with gas (petrol).

**Comment** Our attempt is shown in Figure 8.4.

**Prototyping with Index Cards** Using index cards (small pieces of cardboard about 3 × 5 inches) is a successful and simple way to prototype an interaction, and is used quite commonly when developing websites. Each card represents one screen or one element of a task. In user evaluations, the user can step through the cards, pretending to perform the task while interacting with the cards. A more detailed example of this kind of prototyping is given in Section 8.3.4.