Frank E. Ritter
Gordon D. Baxter
Elizabeth F. Churchill

Foundations for Designing User-Centered Systems

What System Designers Need to Know about People

Springer

Foundations for Designing User-Centered Systems

Frank E. Ritter · Gordon D. Baxter
Elizabeth F. Churchill

Foundations for Designing User-Centered Systems

What System Designers Need to Know about People

Springer

Frank E. Ritter
College of IST
The Pennsylvania State University
University Park, PA
USA

Elizabeth F. Churchill
eBay Research Labs
eBay Inc.
San Jose, CA
USA

Gordon D. Baxter
School of Computer Science
University of St Andrews
St Andrews, Fife
UK

ISBN 978-1-4471-5133-3 ISBN 978-1-4471-5134-0 (eBook)
DOI 10.1007/978-1-4471-5134-0
Springer London Heidelberg New York Dordrecht

Library of Congress Control Number: 2013957359

Cover Image: Badrul Sarwar

Printed on acid-free paper

Springer is part of Springer Science+Business Media (www.springer.com)

Foreword

Our core Masters in Software Engineering course at the University of Southern California is a 2-semester course in which students form into about 15–20 teams of six people to define, design, develop, and deploy working software systems for clients in the local South Los Angeles community. The clients range from IT startups, neighborhood small businesses, local government and local community service organizations, to USC doctors, faculty members, librarians, administrators, and student organizations. The student developers come from many countries and cultures: mostly from the US, India, and China; but also from Europe, Latin America, and other parts of Asia.

One concept that seems to be common among all of their cultures is a version of the Golden Rule: "Do unto others as you would have others do unto you." One of the first things that we now teach the students is that this rule carries a dangerous assumption. How, we originally wondered, could such a universally accepted tenet be dangerous? However, we found that it carries the assumption "Everyone is like me," and that many of the students would follow it to create programmer-friendly user interfaces, and say, for example, "Hard to use? What do you mean? Its tight syntax minimizes keystrokes. It gives you the power of direct access to the operating system. It doesn't need to pinpoint errors because they're obvious from scanning the erroneous command."

We now teach them the Platinum Rule, "Do unto others as others would be done unto," emphasize development and exercise of user prototypes, and provide readings, user domain models, exercises, and win–win negotiation capabilities to help them learn how their clients would like to be done unto.

As we've evolved the course over the last 16 years, we've learned a lot about developers and users the hard way, by trying things out and rethinking approaches that didn't work very well.

We could have avoided a great deal of this learning-the-hard-way if we'd had access to the book that you're holding now. *Foundations for Designing User-Centered Systems: What System Designers Need to Know about People* is a well-organized treasure trove of useful insights and case studies about the characteristics of users and how to develop systems that best fit their strengths and avoid their weak spots.

The book begins with some good motivation, context, underlying science, and conceptual frameworks for human-systems integration. It covers considerations of

users' physiology (Chap. 3), senses (primarily vision and hearing) (Chap. 4), a strong coverage of users' memory, attention, and learning aspects (Chap. 5), and several good chapters on how to improve human–computer interaction. These provide useful information and guidance on human cognitive capabilities and their implications for considerations such as organizing text and menus, mental models (for problem solving and decision making), groupware and social processes, types of users and their design implications (age, gender, disabilities), error avoidance, task analysis, human-system evaluation considerations, and process models supporting human-systems integration, such as the incremental commitment spiral model.

Just to elaborate on one of these, the book is particularly strong in an area most frequently in need of improvement: groupware and social processes. Most computer systems have been developed to help individuals perform individual tasks, and tend to focus on improving individuals' performance. A lot of groupware also gets developed using such systems, so that the individual-focus gets supported more strongly than the group-focus.

An example of the consequences of this has been our series of win–win requirements negotiation tools we've developed and used in our project course mentioned above. Our first three versions of the tools began by enabling stakeholders to enter and classify the win conditions they wanted from the project, after which efforts were made to identify and resolve conflicts among the win conditions. This was often difficult after they had bought into the things they wanted.

Our fourth version of the negotiation toolset was built on top of a group-oriented support system (the Ventana/GroupSystems infrastructure). There, once stakeholders entered a win condition, they did not stay in their own space, but were presented with another entry window showing some win conditions entered by the other stakeholders. This often shifted their thinking to focus on understanding and accommodating others' win conditions (oh, they want this to run on Windows, Mac, and Unix platforms; we'd better not use any one-platform COTS (commercial off-the-shelf) products; maybe we should use a Java virtual machine or make this a Web application; and do they have all three platforms for us to test on?). This opened our eyes to the differences between individual-focused and group-focused user interfaces, but it left us wondering how many other dimensions of group-oriented user interfaces we needed to consider.

At that point, if we could have had Chaps. 8 and 9 of *Foundations for Designing User-Centered Systems*, we would have been way ahead. It covers various cooperation settings (zero-sum, nonzero-sum, and behavioral games); techniques for promoting cooperation; social networking; critical influence factors for group performance (group size, group composition, social distance, spatial distance, collaboration support, leadership capabilities, task attractiveness); types of motivation to contribute to solutions; and social responsibility effects (demotivators to contribute to solutions).

The section on What Leads to Good Teamwork makes another distinction between the knowledge, skills, and abilities (KSAs) traditionally used to measure individual performance and those needed for group performance. "Knowledge" focuses not only on technical and domain knowledge, but also on knowledge of

team objectives and team-mate awareness. "Skills" focuses not only on analysis and synthesis skills, but also on shared situational awareness and conflict resolution skills. The "A" does not represent Abilities but Attitudes, such as mutual trust, team cohesion, and collective orientation. The chapter also has valuable sections on models of social processes and general implications for system design (e.g., structuring user measurement on contributions to mission effectiveness vs. user efficiency as a computer peripheral).

Other strengths of the book are its inclusion of stories, good and bad usage snapshots, puzzles to stimulate learning and make it fun, and many references to helpful sources of further information. A nice observation was "A year in the laboratory can save an hour in the library."

As a bottom line, getting the user interface right can make a fundamental difference (just consider Apple Computer's Fall 2011 quarterly sales of $46 billion and profits of $13 billion). This book may not make you the next Apple, but I believe that it can help make most people and organizations perceptibly better at understanding and satisfying user needs.

Barry Boehm
TRW Professor of Software Engineering
Computer Science Department
University of Southern California

Member, Committee on Human-System Design
National Academy of Sciences' National Research Council

Former Director of the Information Science and Technology Office, and
Director of the DDR&E Software and Computer Technology Office
DARPA

Fellow ACM, AIAA, IEEE, INCOSE
Member U.S. National Academy of Engineering

Preface

Many books on user centered design and HCI focus on the way people interact with technology. This is an important issue, because people routinely interact with technology on a daily basis—personal computers, mobile phones, airplane cockpits, or even more mundane things like electric kettles and toasters. Despite everything that we know about interaction, however, technology still does not always support what we, as *users*, are trying to do, or behave in the way we expect it to. This can be exasperating for us: as users, as designers, *and* as developers.

In *Foundations for Designing User-Centered Systems* we help you to understand *why* people behave and interact with technology in the way they do. By helping you understand both *how* and *why* people behave in the way they do, and by helping you to develop a more systems oriented perspective, we provide you with a framework that will enable you to develop technologies that are both useful and usable. These technologies will also be more acceptable to users because they will be better suited to the way users work in their normal environment.

Our Approach

The people who use technology must be considered to be part of the systems they use. Although people–"users"–are diverse, they also have many characteristics in common. Not all of these characteristics are directly visible or available to system designers without much closer investigation. By understanding the characteristics of users, designers are better able to create safer, more usable, and more acceptable systems.

We have designed *Foundations for Designing User-Centered Systems* to encourage you to ask critical and reflective questions throughout the design process about how your users will work with your technology. Whilst we provide key facts and characteristics about people as users, we have resisted creating a source book filled with lists of endless facts about human characteristics. We have also avoided the temptation of promoting design by rules, so we do not provide lists of guidelines that must be rigidly followed, or known problems that must be avoided.

Our goal is to help you understand the process of designing interactive technologies and to introduce you to a user-centered, systems oriented approach to design. We present a detailed, theoretically grounded approach to understanding people: how they accomplish the things they do and how they work out what they need to do (their tasks) in particular situations.

We have tried to select the most important things you should know about people, based on our experience of working in industry and academia. *Foundations for Designing User-Centered Systems* will help you develop a principled model of users, based on regularities of human behavior, which encapsulates this information so that you can predict how users will behave in different situations. This model will incorporate aspects of how perception, action, cognition, and social processes all contribute to human behavior.

We believe it is important to have the grounding for innovation as well as the ability to evaluate existing systems. Our approach will give you a solid foundation for dealing with a wide range of situations and provide you with the analytical skills to design in innovative ways—including introducing you to computational and cognitive models of how users think. We build on existing methods and techniques, providing you with the basic knowledge that will let you invent your own methods for design and evaluation based on the different settings that you find yourself in.

For Practitioners

As the book has developed, many of our colleagues and collaborators from industry have reiterated the importance of the issues that we address, and how much they support the idea of *Foundations for Designing User-Centered Systems*. They often find that they have to train their staff about users, their tasks, and the context in which they perform those tasks. To address this we provide an extensive theoretical information about design-relevant user characteristics to make practitioners aware of the important issues. In addition, throughout the book we consider the implications for system design, where we offer concrete examples of how the information we present can be applied.

For Teachers and Advanced Students

Our book provides enough material for a semester-long course on users, human–computer interaction, human factors, interface design, or human behavior modeling where users are an inherent part of the envisaged systems. While much more

is known about users than we present here, we have intentionally limited ourselves to what can be covered in a semester. We provide follow-up reading for those who wish to take things further at the end of each chapter. More resources on the topics we cover are continually becoming available online and these could be used to extend our material to support longer or more advanced courses. You will also find some useful resources on the *Foundations for Designing User-Centered Systems* web site (www.frankritter.com/fducs).

Acknowledgments

The book has evolved over time as we and our erstwhile colleague, David Gilmore, have taught human–computer interaction, human factors, user interface design, cognitive ergonomics, and cognitive modeling at the University of Nottingham, Penn State, the University of York (UK), and the University of St Andrews. Collating the material was made possible through the original web site created by David as a way to help support students. The idea of turning it into a book emerged as the web site expanded, and as the material has been updated.

While any mistakes remain ours, we need to thank the many people who have offered feedback and encouragement along the way. In particular, we would like to thank the following people. Peter Lonsdale prepared a talk for a class that turned into lecture notes on the application of our approach to the web, and the students at Penn State (Andrew Freed) and at the University of Nottingham helped refine many of the exercises. Dan Gao, Soo Yeon Lee, and B. S. Sowmyalatha (PSU/UP) provided great feedback on improving this text, constantly encouraging more examples. Alexander Daise, Mark Kozlowski, David Kaethner, Lars Guenther, and Marcel Richter (TU/Chemnitz) also offered many good suggestions on how to improve the presentation.

Our colleagues (and where they used it to teach) provided useful feedback based on use. These include Mithu Bhattacharya (PSU/UP), Michael Qin (NSMRL, U. of Connecticut/WPI), Mark Ackerman (Michigan), Kuo-Chuan (Martin) Yeh (Penn State/World Campus), Marcela Borge (PSU/UP), Pat Clemson (PSU/Beaver), and Olivier Georgeon (PSU/UP).

We received comments from several people at PSU, notably Andrew Freed, C. Lee Giles, Alexander Ororbia II, James Wang, and Luke Zhang. Rob St. Amant (NCSU) and Magy Seif El-Nasr (Northeastern) provided useful comments to improve the direction of this book. Simon Robbie (CMU, Apple) provided extensive suggestions throughout the book after a chance meeting at a pterodactyl ride. Jack Sparks (MCWL) read each chapter and the breadth and depth of his biting but not hurtful comments were encouraging. Lisa Dow (University of St Andrews), Ben Dyson (Ryerson University), David Grayson (Fluent Interaction), Chandra Harrison, Junya Morita (JAIST), Les Nelson (PARC), and Margaret Ritter all provided encouragement and support as well as useful feedback on multiple chapters as the book developed. General discussions with Bill Webber (Erlbaum) and Rajal Cohen (PSU) improved the presentation. We should also note

that books by John R. Anderson, Boff and Lincoln, Don Norman, and Chris Wickens and his colleagues have helped shape this work, and some of the ways they organize topics are reflected here. Don Meeker (PSU) provided photos and useful comments about the use of his photos.

Many people provided feedback on individual chapters which has greatly helped the book as well, including Jennifer Bittner (Indiana), Shawn Clark (PSU/UP), Georgious Christous (European University Cyprus), Ed Glantz (PSU/UP), David Golightly (University of Nottingham), Kate Hone (Brunel University), M. Cameron Jones (Google), Bill Kennedy (GMU), Russell Lock (Loughborough University), Faidon Loumakis (Fluent Interaction), Naomi Malone (UCF), Sylvie Noel (Communications Research Centre Canada/Centre de recherches sur les communications Canada), Shamil Parbhoo (Fluent Interaction), Ling Rothrock (PSU/UP), Marco de Sa (Twitter), Joe Sanford (PSU/UP), Elaine Seery (Science Editing), Sarah Sharples (University of Nottingham), Tim Storer (University of Glasgow), and Fiona Woodcock (Fluent Interaction).

Gordon Baxter's work on the book was supported by funding from the UK EPSRC's Large Scale Complex IT Systems project. Frank Ritter has drawn on material developed with support from ONR and DTRA and applied this material to their projects, having been influenced by these projects and having used this material to help train researchers on those projects. A Senior Fulbright Fellowship provided support to teach this material at TU/Chemnitz, and the College of IST has been supportive.

Finally, Beverley Ford, Ben Bishop, Jake Kirby, and a copyeditor at Springer have been very helpful and encouraging. They helped us push this book over the finish line with their kind words and support. Figures and pictures used with permission by their authors. Unattributed figures are copyright by the authors and are available for use by instructors on an instructors' web site.

Citations are done in Springer house style; references are done in APA format.

Contents

Part III Methods

Part IV Summary

Overview of Book

Foundations for Designing User-Centered Systems is organized into four parts, as shown in the Table of Contents. The first part has two chapters. Chapter 1 introduces the approach of understanding people (commonly referred to as "users"), their tasks, and their context. It motivates when to study the user, including examples and some risks that arise when you do not. This chapter also notes some ways to organize this knowledge, including risk-driven design and the use of cognitive models.

Chapter 2 provides an overview of the fields that contribute to our approach to designing user-centered systems. This chapter will help readers understand the relationship between different research communities and point to relevant literature and to where further information can be found.

The second part of the book describes what we consider to be the core, design relevant characteristics of users. These chapters build up the foundations for describing users using what we refer to as the ABCS framework: A for anthropometrics, B for behavior, C for cognition, and S for social aspects that underlie human activity. Chapter 3 describes important aspects of users' bodies, *anthropometrics*, including how they sit at terminals, how they type, and how they touch. Chapter 4 deals with the underpinnings of human *behavior*, describing the basic senses used to interact, particularly sight and hearing, as well as why individuals are motivated to behave in particular ways. Chapters 5–7 address *cognition*. Chapter 5 describes the foundations of cognition, that of memory, attention, and learning, particularly the aspects that apply to system design. Chapter 6 describes higher level cognitive capabilities related to system design, that of mental representations influencing mental models, problem solving, and decision making. Chapter 7 examines communication between users and technology. These aspects include some fundamental factors of language related to interfaces, how users read, and typical information-seeking behaviors. Chapters 8 and 9 look at *social* aspects of users. Chapter 8 examines social effects on decision making and factors affecting teamwork. Chapter 9 looks at larger scale, network effects, and provides some models to summarize behavior in this area.

Chapter 10 introduces the study of errors—errors are often a good source of information about human behavior when interacting with technologies. We can ask several questions. What went wrong? Why did it go wrong? How can we prevent the same thing happening again? Chapter 10 provides some background

knowledge on errors, including error rates and how technological and human factors interact to cause system errors. The chapter also provides some tools for studying and ameliorating the effects of errors.

The third part of the book provides some methods for studying users in systems. Chapter 11 introduces task analysis. We note several uses for task analysis and illustrate how it can be a very cost-effective method. Worked examples are provided for each method.

Chapter 12 provides two additional methods for improving the design of systems. These methods also help to summarize and apply what we know about users. Cognitive Dimensions (CDs) is a way to summarize how users interact with systems. CDs also offer a framework for making predictions about potential errors; these predictions can provide the groundwork for directed usability tests and for formal or informal quality testing. The chapter also describes Norman's Gulfs of Evaluation and Execution. The Gulfs offer a framework for understanding where users need to be helped to understand and to interact with systems.

Chapter 13 describes empirical evaluation focusing on *user studies*. This chapter describes how to start to run a usability study, and provides suggestions about what to do and what to measure.

Chapter 14 provides a summary of users and how to design user-centered systems. We first summarize the ABCS and then offer an introduction to user modeling as a way to encapsulate the detailed knowledge we have about users as a quick way to generate predictions. We conclude by describing the Risk-Driven Incremental Commitment model as a way to apply what we know about users to system design.

The Appendix describes an air accident that occurred several years ago, known as the Kegworth accident because it took place near the small town of Kegworth in the midlands of the UK. Although a simple diagnosis of *pilot error* was offered as the cause of the accident, on closer analysis this accident resulted from multiple issues which transpired at a number of system levels. The Kegworth accident is used as an example in several places in the book to illustrate how many levels and aspects of a system can influence system performance—and to underscore the complexity of systems that are made up of people and of interactive and interacting technologies. This complexity means we often cannot and should not come up with simple assertions about errors, but rather look for weak points in the overall system and deal with those weak points systematically and in a grounded way.

We believe knowing more about people will help you develop the kind of grounding you need. We also believe that developing a systems approach will protect you from erring toward simple design assumptions and narrow solutions.

Each chapter includes an abstract, an introduction, and a summary to orient the reader and to increase understanding. We include consideration of what the implications are for system design at the end of each major section. There are also lists of other resources for those people who want to find out more.

Endorsements

For all of us who have been 'put on hold,' recorded for quality purposes, been forced to talk to a mindless, uncaring voice non-recognition system, or simply beaten at the computer keyboard in sheer frustration, hope and help are at hand. For Ritter and his colleagues are injecting rational, user-centered design into such systems development. It is a timely contribution, devoutly to be wished. Their text is a shining example of their advocated principles. Readable, informative, easy to use, and innovative, this works puts into practice what it preaches. It should be on the desk of everyone who looks to conceive, design, fabricate, and manufacture any modern technological system—no matter how hard, no matter how soft. Even if only a proportion of designers and users read this book we will be so much better off. If it gets the circulation it deserves it could change our world—and that very much for the better. If not, technorage will only grow and the Luddites will once again become a viable social Party!

Peter Hancock
Provost Distinguished Research Professor
Pegasus Professor, and University Trustee Chair
University of Central Florida

As a software engineer, I've been advocating for the past 20 years that we will only see real improvements in our software when we move away from a technocentric view and adopt a wider perspective that takes into account what users really do. Too many software engineers consider this to be a 'CHI issue' and believe that they can focus on the technology and leave the 'soft stuff' to designers of the user experience.

Well, they are wrong. Not only is it the case that most companies don't employ specialist UX designers, all too often these designers don't understand the underlying technological issues that have to be taken into account if our software is to work effectively, efficiently, and securely. The only way forward in my view is for software engineering education to include education in the human, social, and organizational factors that influence the ways in which software is designed and used.

Up till now, this has been very difficult. Conventional texts on CHI have a different audience and, all too often, focus on current technology rather than

underlying fundamentals. This book is different and it's one we've been waiting for. It explains in depth fundamental human capabilities, cognitive strengths, and cognitive limitations that influence the way that we choose, understand, and use software systems. It explains how we communicate and how that affects the ways that interfaces are used; it discusses collaborative working, factors that support and inhibit collaboration, and methods that can be used to understand how people work.

Most importantly, I think, it doesn't just present these fundamentals in isolation. Every chapter in the book has a section discussing the implications for design so that readers not only learn fundamentals but understand why these are important and how they might influence their work. These bring unfamiliar material to life for software engineers and clearly demonstrate why this is important for practical systems design.

This is both a textbook and a reference book. It would be a great basis for a course in human-centered software engineering but, as well as this, practicing engineers can access and learn from the individual chapters and the follow-up material that is suggested. The lack of accessible and comprehensive material on human factors for software engineers has been an important barrier to more widespread acceptance of a human-centered approach to systems design. This book has broken down that barrier and I can thoroughly recommend it to all engineers.

Ian Sommerville
Professor of Computer Science
University of St Andrews, and Author of *Software Engineering*

This is the book I really needed when I developed a course on Applied Cognitive Science within our Master's program in HCI with Ergonomics at UCL. At the time, I had to improvise with a mix of texts on cognitive psychology, engineering psychology, and HCI. *Foundations for Designing User-Centered Systems* fills an important gap in the space of texts for students and practitioners of HCI, focusing, as it does, on understanding people and their interactions (both social and with technology). Critically, it also draws out the implications of this understanding for design. It manages to cover all the key topics in this space while also being engaging and, at times, quirky. A textbook that makes one smile and want to read more is a textbook that works.

Ann Blandford
Professor of Human–Computer Interaction
University College London

I really enjoyed the reading of this lively book that I believe can be appreciated by different kinds of readers. A useful publication written with wit, helping the reader to discover the human capabilities and limitations, the patterns of user's attention and the fundamental principles to adopt at the early stages of system design.

The authors take into consideration not only the usefulness of the artifacts, but also the impact they have on safety. In fact, the main cause of accident nowadays in aviation is the loss of control of the aircraft, often induced by a poor human–machine interaction. This is due, mainly, by poorly conceived interfaces, as the result of a lack of understanding of who the final user is. The overall problem lies in the very fact that the one who produces the artifacts is not the one using them. Eventually, after many years, the study of the human factors as a discipline at the cross-road between medicine, psychology and engineering is addressing the design of the interfaces.

As a human factor specialist, involved in flight operations, I think this book should become a 'must' even in the flight safety domain.

Antonio Chialastri
Senior Captain and Independent Human Factors
Consultant in Aviation and Medicine, Italy

This broad ranging survey of user-centered design techniques provides an effective introduction for designers into what people do, why and when they do it, and what motivates those behaviors.

If you ever wanted to know what a 'steep learning curve' actually looks like and how the user will interact with your system at different points along this curve then this is the book for you!

Through well-illustrated examples, it considers a wide range of topics from traditional ergonomics, through user behavior, cognitive models, and social factors. Many of the examples take off the traditional 'blinkers' of user centred design and show how a human decision at the 'sharp end' may well have its roots in a much wider and blunter context.

As a chief architect for large programs, this book has given me access to a variety of new techniques and an extended vocabulary that I look forward to introducing my design teams to.

Richard Hopkins
Chief Architect and IBM Distinguished Engineer
Co-author of *Eating the IT Elephant*

The HCI profession emerged when psychologists teamed with developers. Design was missing. Today, good teams have strong designers and technologists—but psychological insight is often in short supply. This book fills that gap with a fresh look at established and new knowledge and approaches.

Jonathan Grudin
Principal Researcher at Microsoft Research
ACM Fellow

If you want to design or build interactive systems that are both useful and usable, *Foundations for Designing User-Centered Systems* is an excellent place to begin.

Philippe Palanque
Head of Interactive Critical Systems Group
Universite Paul Sabatier Toulouse
Co-chair of CHI 2014

The "Who, What, When, Where and Why of Human-Systems Interaction"—a practitioner's primer for Systems Designers looking to advance human computer symbiosis in their designs. The book provides a straightforward, easy-to-read introduction to the process of designing interactive technologies using human-centered approaches that avoid the cookie-cutter, simplistic recipes all too common in other publications. Also worth noting is that this guide not only covers foundations for beginners, but also includes practical, real-word examples, as well as emerging essential topics for the design of systems, for more advanced practitioners. The reader will quickly discover that this book provides essential, innovative, and targeted tools for designers who are focused on enabling seamless interactions between humans and technologies. For anyone looking to advance human-computer-symbiosis, this book will not gather dust on your shelf!

Dylan Schmorrow, Ph.D.
Chief Scientist, Soar Technology, Inc.

Anything that helps software developers think more about the mental states of their users and how that affects the utility and usability of their software is a good thing. Even if you don't plan to become a human factors expert, you will find good ideas in this book to help make your applications more successful.

William A. Woods
Research Scientist and Software Engineer

The foundations for designing user-centered systems really delivers on its title. The book succinctly captures the key anthropometric, behavioral, cognitive, and social concepts that are the foundations for designing user-centered systems. Furthermore, the authors artfully imbedded human factors principles into the manner in which materials are presented, turning the book into a demonstration of good practices. I find the structure and layout of the book make it an excellent introductory text for a course in HCI as well as a useful initial reference source.

Michael "Q" Qin
Adjunct professor, WPI

Part I
Introduction: Aims, Motivations, and Introduction to Human-Centered Design

Chapter 1
Introducing User-Centered Systems Design

Abstract If designers and developers want to design better technologies that are intended for human use they need to have a good understanding of the people who are or who will be using their systems. Understanding people, their characteristics, capabilities, commonalities, and differences allows designers to create more effective, safer, efficient, and enjoyable systems. This book provides readers with resources for thinking about people—commonly called "users"—their tasks and the context in which they perform those tasks. Our intention is to enable you to make more informed decisions when designing complex interactive systems. This chapter thus introduces this argument through example design problems. We then present the benefits and costs associated with understanding the user. Two approaches for understanding users are introduced. The first is a framework called the ABCS for understanding, in broad strokes, different aspects of users. The second is user knowledge and action simulation for developing and testing how users approach tasks in more detail. After reading this chapter you should be able to appreciate why it is important to understand users, and the associated benefits and costs of doing so.

1.1 Introduction

Most of us use interactive technologies every day—cell phones, TVs, alarm clocks, cars, vending machines, computers, cameras, microwaves, ovens, ticket machines—the list is endless.

Technology can help us achieve what we desire to do or need to do, but it can also hinder us. When we cannot get something done, when our expectations are not met, or when technology is too hard to use, we get frustrated. When technologies and systems are unpredictable, delays and unforeseen problems can occur.

This book is about designing technology and systems for use by people. We offer an introduction to what we know about why humans do what they do when they do it as users of technology. The book has one central premise:

F. E. Ritter et al., *Foundations for Designing User-Centered Systems*,
DOI: 10.1007/978-1-4471-5134-0_1, © Springer-Verlag London 2014

Understanding people will help you build better interactive technologies and systems.

When we say "understanding people" we mean:

- Knowing how to observe and document *what* people do
  - Using appropriate methods to get credible results and differentiate anecdotes from reliable data
- Understanding *why* people do what they do
  - Developing insights into people's conscious and unconscious motivations for doing things
- Understanding and predicting *when* people are likely do things
  - Understanding people's patterns of behavior
- Understanding *how* they choose to do things the way they do them
  - Understanding what options people actually have and/or perceive they have available to them, understanding the constraints they are under and assessing what the resources they have available to them.

We propose that systems should be designed in a user-centered way. Being user-centered means considering human characteristics and capabilities during system design. It means explicitly asking: who is going to use the system/technology and why; what are they hoping to achieve in using the system/technology; how much effort are they willing to put into learning how to use the system/technology; whether they will be operating the system alone or with others…. Being user-centered means knowing why, as well as how, users do what they do when they do it. We propose that consideration of users' basic human characteristics should be in place before system development begins. Reflection and experimentation with potential users of the system should take place throughout the design and development process using methods like brainstorming, storyboarding, low to high fidelity prototyping, and, as the system gets closer to full functionality, with more formal use testing.

This book assumes no previous knowledge; it is designed to be accessible to those without a background in psychology or computer science; if you have already taken a traditional human–computer interaction (HCI) course, this material may be a quite easy read and help you organize your thoughts. If you have taken several psychology courses, you are likely to recognize much, but perhaps not all, of the material here.

1.2 Starting to Understand Users

Many designers and developers make two fundamental errors. They assume that understanding how a technology will be used can be derived from introspection: from imagining how it will be used. This assumption is based on a second

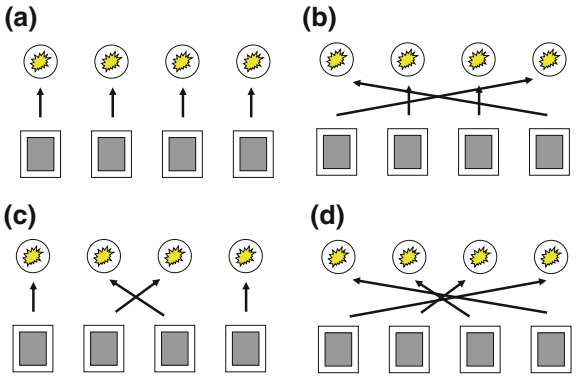

Fig. 1.1 **a** Rank order the quality of these switch to light mappings. **b** Note how long, on average, it will take to push a button on each panel. (Adapted from Payne 1995)

error—that everyone is the same. We know the second assumption is not true from simply observing that the world is made up of very different people with different motivations, different backgrounds, and different skill sets.

To illustrate how our intuitions about people may be incorrect, and why it is always worth testing your designed system with people who will use that system, we offer the following examples.

1.2.1 Designing Mappings Between Buttons and Lights

It is generally claimed that the better designs are those that offer simple, clearer mappings between an action and a response. However, the question is: what is a clearer mapping? Consider Figs. 1.1 and 1.2 taken from a study by Payne (1995) on how well naive subjects could judge the quality of interface designs for a simple system. Payne's experiment assessed what design people predicted would rank best to worst on a very simple interface, where a number of different mappings between the controls and resulting system state were compared (what is called "stimulus-response compatibility" in the scientific literature). In the following example, study participants were able to rank order the designs in Fig. 1.1 from best to worst. They were asked two questions: (1) what is the mapping of lights to switches that gives the fastest response time? and (2) can you give a prediction of how long they will take on average?

Sixty out of 70 subjects got the top ranked one correct. However, only four out of 70 got the complete order correct. The results of the study suggest that, when confronted with anything but the most obvious choices, designers without training may make poor design choices. Before going on, you may wish to try this task yourself. The correct order is given in the exercises at the end of this chapter.

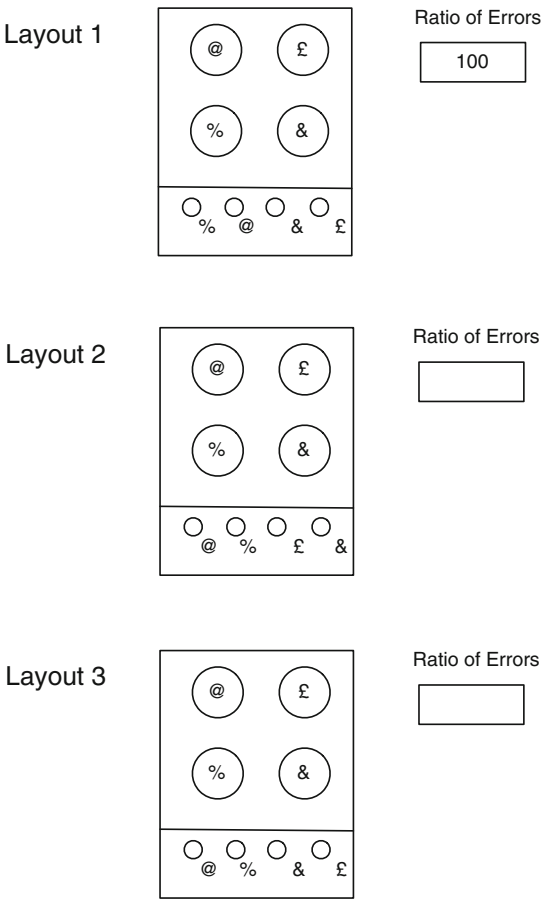

Fig. 1.2 Rank order the quality of these stove burner to knob pairings. If layout 1 will give 100 errors, how many errors will the other pairings lead to? Adapted from Chapanis and Lindenbaum (1959)

1.2.2 Designing Stove-Top Mappings

For our second example, take a look at the stove-top designs in Fig. 1.2. Which is the best burner to control knob mapping? If you think you know the best mapping, can you provide a quantitative measure of how much better? If layout 1 has 100 errors for a given amount of use, how many errors will the other two have?

For the examples in Fig. 1.2, only four out of 53 subjects selected the correct order of layout to be layout 3 (76 errors per 1,200 trials), layout 2 (116 errors per 1,200 trials), and then layout 1 (129 errors per 1,200 trials), and only 15 out of 53 could correctly identify the best design.

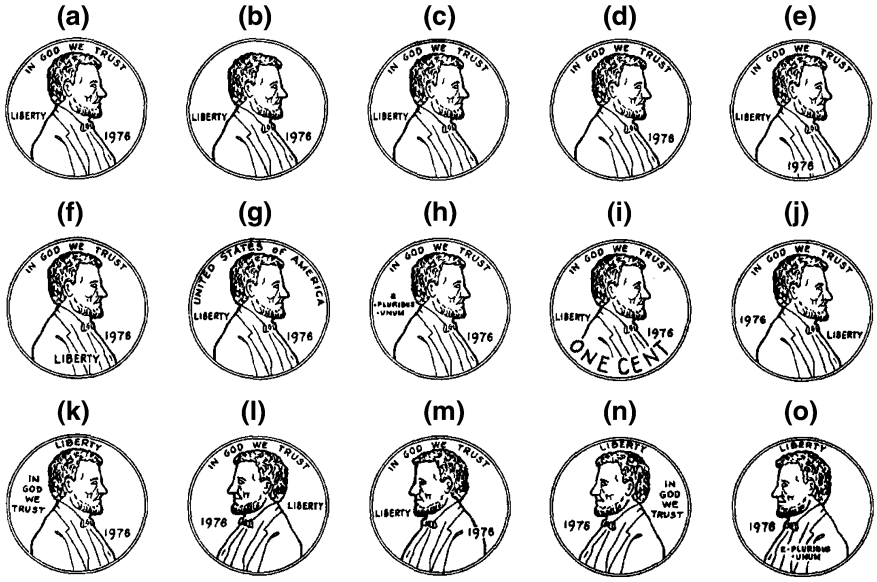

Fig. 1.3 Possible views of a US penny. Without looking in your pocket choose the correct one. Taken from a study by Nickerson and Adams (1979). (Used with permission of Elsevier)

1.2.3 Designing Coins

For our third example, we would like to look at coins. Can you pick out which is a US penny in Fig. 1.3 without looking at a real one?

Most Americans would think that they could recognize a US penny, but more than half of Nickerson and Adams' (1979) American subjects shown the pennies in Fig. 1.3 could not pick out the penny from a set of 15 examples. The correct answer is given at the end of this chapter in Exercise 1.2.

We all know well enough what a penny looks like—relative to the other coins we might encounter—but not in any more detail than is necessary. With the set of alternatives provided by Nickerson and Adams, the choice has to be based on recalling specific features of a penny, which most people have never memorized and have never needed to memorize. You can see similar effects in computer interfaces where users cannot recall which commands are located on which menus (Exercise 1.2 explores this question further).

Although coinage systems may appear a long way removed from the design of user interfaces, they provide good examples of how and why we can benefit from considering the users' perspective in design to avoid system failure. France and the

USA have both tried to introduce new coins (e.g., the Susan B. Anthony dollar) with little success, partly due to the lack of care in the design of the new coin. In contrast, when Britain got a new currency in 1971, switching to a system where one UK pound was equal to 100 pennies, the introduction of the new coinage was a resounding success. It turned out that one reason for the success was a substantial body of research on how people perceived the value of coins (e.g., Bruce et al. 1983) as well as attention to how the different proposed coins might be made least confusing to the elderly or sight impaired. During the research it was recognized that many people need to identify coins from touch alone (e.g., the coin in your pocket) and that designing for the blind user actually meant designing for everyone. The cost of this research was a very small component of the costs of introducing a new coinage system (e.g., all of the new vending machines to be developed), but it helped ensure the success of the whole enterprise. Subsequent changes to the coins have also followed these guidelines, with the two pound coin, for example, being the same basic shape as the one pound coin, but larger and heavier.

In these examples we see one of the first universals of human behavior—people remember those details that they pay attention to but only in sufficient detail for the tasks they are performing. This is universal, but it does not enable us to predict fully what details someone will remember, because there are differences in how much attention people have to spare, what tasks they are performing, and thus what details they will remember. The first two problems in Figs. 1.1 and 1.2 are difficult because the differences in performance of the tasks are not particularly available to consciousness, and most people's representation of how they think they perform these tasks in this area do not reflect how people actually perform the task. The penny question (and the menu question) represent the difference between recognition and recall memory. Usually identifying a penny just requires being able to discriminate between it and other coins. With the set of alternatives provided by Nickerson and Adams, the choice has to be based on recalling the features of a penny, which most people have never bothered to commit to memory (why would they?).

Another classic example is remembering your new cell phone number. It takes a long time to learn it because you, personally, never need to use it (unless you misplace your phone, in which case calling it is a good strategy for finding it!). However, if someone asks you for it, you either have to recall it or have to go through the menus on your phone to find it, eventually recognizing the steps that enabled you to find the number.

This disconnection between how we think we behave and how users *really* behave is common and there are plenty of reasons for it. In most cases we are too busy doing a task to properly observe how we are doing it. When we can observe how we are doing it, it is rare that we can correctly and completely infer why we are doing the task—the observation of behavior is separate from the generation of it. Ericsson and Simon (1993) provide a full explanation of why it is hard for people to examine their own thinking processes. Their explanation includes that when we recognize how we behave, we rarely make written notes and thus any

Table 1.1 Summary of some of the causal factors in the Kegworth air accident and lessons to note

- The Engine Instrument System (EIS) used digital displays. A survey of the airline's pilots after the Kegworth accident showed that nearly two-thirds of them believed that the new EIS was not effective in drawing their attention to rapid changes in the engine parameters. If it had, the accident might have been avoided. Thus, the design of the interface for the EIS did not present data in a format that could easily be perceived by the pilots

  Lesson: You need to understand how people look at the user interface to extract information which they then use to make decisions and take actions
- Neither pilot could recall having seen any indication of the abnormally high vibration levels on the EIS. The Captain noted that he rarely scanned the vibration gauges because he had found them to be unreliable in other aircraft in the past. Experts, such as pilots, have a highly developed mental model of the world in which they normally operate, which helps them carry out their tasks. The Captain appears to have excluded the vibration gauges from his mental model because he believed the readings were unreliable

  Lesson: You need to understand how people create and use mental models to help them use a system
- The B737-400 was a glass cockpit aircraft, in which the information is presented on digital displays, rather than on analogue instruments and electro-mechanical displays. The airline (BMA) did not have a glass cockpit flight training simulator for the B737-400, so pilots could only gain experience through actually flying it (i.e., on the job). The only training the pilots were given for the B737-400 was a 1-day audio-visual conversion course

  Lesson: You need to understand how people learn to use new and, particularly, complex systems
- Three members of the cabin crew said they saw evidence of the fire in the #1 engine, but they did not report this to the pilots. The flight crew believed that the problem was with the #2 engine. This seems to have been a failure in what is called Crew Resource Management, a procedure designed to ensure that all the members of a flight crew (pilots and cabin crew) communicate with one another and work together as a team

  Lesson: You need to understand how social issues, including communication, can affect how people use a system
- The B737-400 was fitted with a new type of engine. The engine was thoroughly tested on the ground before being certified by the appropriate authorities. The engine was not tested either in an altitude test cell (which simulates the conditions of flying at high altitudes) or in flight, however. This scenario illustrates how decisions that are made at remote distances from the user interface in a system can have an impact on the way that the users behave. Emergency events, like engine failures, are normally covered by checklists in the QRH (Quick Reference Handbook) that is used by all pilots to deal with known situations

  Lesson: You need to understand that decisions taken at a place and time that are greatly removed from where the system is used can affect the way the system will behave

memories of how we behaved are subject to the inherent frailties of human memory. We are also not very good at estimating time accurately and have trouble keeping track of successes and failures. Finally, there are some particular aspects of our own behavior that are very hard to observe, such as basic perception, and some that are hard to describe and reason about verbally, such as performing some spatial reasoning tasks.

1.2.4 What Happens If You do not Take Proper Account of Users, Tasks, and Context?

The Kegworth Air Accident (see the Appendix for a detailed account) was, like many air accidents, the result of several events. Many of these events happened in a short space of time, but some were more distant both in time and in space. From the point when a problem was detected with one of the engines, to the point at which the plane crashed took less than 8 min. Table 1.1 lists some examples of the types of things that went awry and contributed to the accident and the lessons you should note.

This book will help you to understand the underlying issues, and show how you can analyze them. Once you understand that these issues arise at different levels, and how they can interact, you can start to take appropriate steps to make sure they are prevented (or their effects are at least mitigated) when designing systems.

1.3 The Benefits and Costs of Understanding Users

Assuming that your users are just like you can be described as a fundamental attribution error of design. This error is essentially the inverse of what is called the *fundamental attribution error* in social psychology (described in Chap. 8). In the fundamental attribution error you assume that people are not like you when, in fact, they are.

In the fundamental attribution error of design, you assume that your users are like you when, in fact, they are not! Users often can't use the systems as well as the designers because they do not know as much. Sometimes the opposite is true. Users can to be quite resourceful and innovative, and will often use technology in ways that the designer had never fully contemplated; for example, spreadsheets were originally designed for use by accountants, but many people now use them for processing all forms of tabular data. Similarly, the short messaging system (SMS) was originally designed to help engineers debug mobile phone communication systems, but now it is widely used by everyone with a mobile phone as a means of general communication.

If you understand your users and take appropriate account of them when designing your system, there are three main types of benefits (or payoffs) that can result: more usable products (which can lead to wider adoption and more accelerated adoption rates), cost savings, and safer systems. There are some caveats, however, which can be seen as costs: understanding and taking appropriate account of the user in your design does not necessarily guarantee success, and knowing how much you need to understand about your users is a hard question to answer. We discuss all of these below.

1.3.1 Benefit 1: More Usable Products

Understanding people can help you design systems that are more usable, more learnable, and more efficient. For example, the adoption of email has become more widespread because of wider availability of devices and infrastructure but also because email interfaces have progressed from requiring in-depth computer science and systems administration knowledge to manage installation and use to relatively easy-to-use web interfaces with help documentation, message management and retrieval, directly usable features like formatting, and the ability to easily attach or embed multimedia.

Web design provides another example. Changes in the costs of hardware and bandwidth have made it faster and cheaper to develop web pages, but many people and businesses would not be generating as many web pages if they still had to use raw HTML. The rise of special purpose, easy-to-use HTML editors (e.g., Webbly) is one reason for the massive uptake of the web (Holmes 2005). AOL and the initial Netscape browser were both successful products because they made an existing service more usable and more widely accessible. Work in eCommerce suggests that ease of use of existing products and the expected cost of learning to use a new interface can also lead to a type of brand recognition and later to loyalty (Johnson et al. 2003).

Sometimes the usability of a tool does not increase directly through improving an interface but because its utility increases and its frequency of use increases. The decreasing weight (and size) of cell phones has made them easier to carry around, and thus always available. Part of the success of the web arises out of the creation of search engines: Bing, Google, DuckDuckGo, Ask, and the more specific search engines like Citeseer and DBLP (dblp.uni-trier.de) increase the usability of the web by helping users find information based on the topic they are looking for rather than by the location of the information. This increase of usability is not driven by the visual interface, but at a deeper level of supporting the user's tasks.

Users sometimes have problems in understanding what they are looking at. Norman (2013) refers to this as the Gulf of Evaluation. Users also encounter problems in knowing or being able to discover what they have to do to execute their task using a particular interface. Norman describes this as the Gulf of Execution. The Internet examples above are examples where the Gulf of Execution has been made smaller, and thus made more tractable for a wider range of users. We will return to Norman's Gulfs in Chap. 12.

More often, lack of understanding of users leads to some groups of users being excluded. We have worked with web sites that now make sure that they have text versions available to support not only the visually impaired (through text readers and descriptions of pictures), but also two types of users that do not come to mind easily in a US college environment with ubiquitous broadband—those users separated from the site via dialup lines or by vast distances (Ritter et al. 2005).

1.3.2 Benefit 2: Financial Savings

Designing to support users can save companies money, even of the order of millions of dollars. One case is Nynex, the New York telephone company, in the early 1990s. The toll and assistance operators (TAO) are the people who help you when you dial "0". They help customers with collect calls, billing, and other more complex calls. In the early 1990s Nynex was considering upgrading their TAO workstation. They had a room with about 100 of these operators; it was believed that new graphical user workstations could improve productivity. The cost of upgrading all the workstations was going to be about $500,000 (in 1990s dollars). The company engaged a team of applied psychologists to look at how much faster the new workstations would be. The results of a task analysis (using a form of GOMS which is described in more detail in Chap. 11) suggested that the new workstations would not be faster, but would, in fact, be 4% slower to operate. This may seem like a small difference, but a 4% reduction in productivity was going to cost Nynex $2.4 million a year—in addition to the cost of the workstations.

Nynex ran a study to discover how much faster the new workstations would really be. After allowing time for the operators to learn to use the workstations, the operators' performance plateaued about where it was predicted—4% slower. NYNEX now claims that this study saved them $2.4 million per year. The user study paid for itself in the first week (see Gray et al. 1992, 1993 for more details). The slowdown in operator performance was not caused by the fact that the new workstations simply required the user to take more steps to achieve the same goal. The reason was the new interface did not allow multi-tasking; the operators could not type while waiting for the caller to speak which they could with the old. Improved computer processor speed could not compensate for the loss in parallel activity the users had with the previous design.

The NYNEX example reveals the benefits of considering people—in this case the operators—even when there does not appear to be a problem. In many instances the main advantage of studying people using systems—that is, conducting "user studies"—is to identify where people make errors, so that we can prevent them or mitigate their consequences in the final product. People often do not type what they want to type, and sometimes push buttons that they did not intend to push. Strangely enough, this problem can be more prevalent amongst highly-skilled expert users, than amongst beginners. Errors that occur when someone knows the right thing to do, but accidentally does something different, are commonly referred to as 'slips' to distinguish them from mistakes, where the action is taken on the basis of an incorrect plan (Norman 1981; Reason 1990).

These slips can also occur on well-practiced interfaces that do not attempt to catch such slips. These slips can also be very expensive. A local paper (Centre Daily Times, 15 Feb 2002, p. C38) reported that a financial services firm lost up to $100 million because it executed a sell order of 610,000 shares at 16 yen instead of the correct order of 16 shares at 610,000 yen (approximately 100 yen/US$).

In a review of the application of user models to evaluate Army systems, Booher and Minninger (2003) report many instances where millions of dollars over the course of a device's lifetime were saved by better interface design, through reduced training, for example. They also highlight several cases where systems had to be scrapped or modified at great expense because they were not usable.

Use of machines that have different modes can often mislead users into making errors. Photocopier machines that can be used to send faxes have a common problem. The default mode of these machines is to copy, but users may not realize this. Users may type in the area code (say, the U.S. area code 415) as the starting point for sending a fax, but the copier interprets this as a request to make 415 copies of the document that the user intended to send as a fax! More explicit displays and more intelligent systems might attempt to catch this type of error. Photocopies may be relatively cheap, but this type of problem with airline tickets, machine tools, or photographic films quickly become expensive.

1.3.3 Benefit 3: Safer Systems

Much of the work that has made airplanes the safest transportation per passenger mile (Gigerenzer 2004) has gone into supporting pilots and air traffic controllers to avoid and, more importantly, catch and recover from errors. This has led to a drastic decrease in accidents previously ascribed to 'pilot error'. As the cause of these accidents were typically attributed to well-trained and alert pilots, it is fairer to diagnose these errors as poor fits between the pilot's capabilities and the machine at particular times and for particular sets of tasks. Improving this fit thus improved airplane safety.

Medicine provides a rich source of examples too. For instance, interfaces that allow users (e.g., nurses) to type in the digits of a drug dose are inherently more dangerous than those that force users to dial them in using a wheel for each digit (Pew and Mavor 2007). When typing, a repeated digit can increase the dosage by a factor of ten. This type of mistake is not possible with a dial-based interface.

Medical X-ray machines are powerful devices and often offer little margin for error. In addition to their technical requirements, they can have usability problems because their effects are not directly and immediately visible. In the case of radiation treatments for cancer, multiple professionals are involved in their use, from the oncologists and radiologists who specify the treatment, the technicians who administer it, to the physicists who maintain it. There are many examples of where interface design for treatment using X-ray machines and other medical devices have ignored the user's capabilities, tasks, context, or some combination of these, and this has led to loss of life. Perhaps the most famous case is the Therac 25 (Leveson and Turner 1993). Between 1985 and 1987 there were six known accidents involving massive radiation overdoses with the Therac. Notably, such accidents rarely arise from a single cause. The user interface was only one of several

contributory factors. In addition to problems with the technology and safety interlocks, the system (including the technician) was poorly prepared to deal with typing mistakes by the technician, and in many installations the level of feedback from the Therac to the radiation technician was not sufficient to help them catch the mistakes sooner.

1.3.4 Cost 1: Understanding the Users Does Not Guarantee Success

Although improving the usability of a system can save lives, lead to product success, and save money, usability is neither a necessary nor sufficient condition for success, nor is it a protection against loss of money or life. Systems with poor usability can still be successful for a variety of reasons. For example, they may offer a functionality that is unique and useful. The earliest versions of planes, computers, printing presses, and satellite phones were all difficult to use, but successful because of their unique functionality.

Products that are well designed with the user in mind may still not be successful. Most or all aspects must be right for a product or system to succeed. Making the usability right does not make the time to market right, it does not make the price appropriate, and other critical aspects such as reliability or marketing may fail.

The system also has to be acceptable to the users. The interface may be well designed on a local level, but if it clashes too much with existing practice (even if the new system is correct) it can quickly fall into disuse (e.g., see Baxter et al. 2005). Similarly, if management does not appropriately support the transition to the new system, it may also fall into disuse. Glashko and Tabas (2009) argue that to understand success you need to understand the user, the business model, and the technology.

The lack of usability can be a sufficient reason for *failure* and this is sometimes overlooked. For some systems, however, usability is not the biggest risk to success. Indeed there are many factors that contribute to success, and none of them on their own are necessary or sufficient to guarantee success. Pew and Mavor (2007) suggest taking a risk driven spiral-based approach to development to deal with this; we describe this approach later in the book.

1.3.5 Cost 2: Knowing When to Stop Analyzing the Users can be Difficult

Knowing when you should stop analyzing the users and start building your system is a difficult problem to address. It is an ongoing issue for HCI (and system design

in general) that should be able to demonstrate a worthwhile return on investment. Nielsen (1993), for example, argues that many usability-related design problem issues can be identified by studying a small numbers of users (about five to eight). The caveats are that the results of this approach are highly dependent on the types of users involved and the particular interface. As yet there are no hard and fast rules that can be applied to all systems which will tell you when to stop the user analysis and start building.

The traditional approach to systems deployment largely focused on making the users fit the system. In other words, companies employed the right people (selected using psychometric testing, qualifications, and so on) and, where necessary, trained them as a way to bridge any remaining gaps between the system and the users. This approach has become increasingly unacceptable as people have become better informed about technology and now expect to use it out of the box, In addition, there have been recent political changes which require that systems are accessible to more people (e.g., the Americans with Disabilities Act), rendering the idea of fitting the user to the system less unacceptable.

It is now generally the case that you should design (or re-design) your system to make it fit your users. We would strongly argue that you need to think right from the very start of the project about your users, the tasks they will perform using your system, and the context in which your system will be used. In other words, when you are defining the system's functional requirements, you should also be defining the usability requirements.

The level of detail required here should be guided by the associated risks involved. If you only talk to developers as proxy users to determine usability requirements, for example, there is a large risk that the delivered system will not be acceptable to the real end users because the proxy users will not understand how the real users will carry out their work using the system in a work context that may constrain how the system can be used. If your system will be used in extreme or safety critical environments (e.g., in space or aviation), for example, your users will be highly skilled practitioners making decisions and performing actions in a limited time frame, where the results may have life or death importance. These risks are increased if the designers are unlike the users (Casper and Murphy 2003 provides a nice case study). In these cases we recommend that you do some background work in the domain, looking at existing systems similar to the one you are designing and consulting appropriate resources such as books, as well meeting the users and seeing their context and tasks and running some studies to test out your ideas and develop your understanding of the system's context.

For simple, straightforward systems developed for your own purposes (like many systems that are used in research, for example), you may not need to worry too much about the design of the user interface. Even for very small numbers of expert users it may not be worthwhile spending large amounts of time and effort on developing the user interface because the costs may well exceed the benefits.

Often your population of users will be heterogeneous, and if you are not aware of this heterogeneity you could end up disenfranchising large sections of your users. We have worked with web sites that now incorporate text versions so that they can also support the visually impaired (through screen readers and descriptions of pictures), and users that access the web via low speed connections, such as dialup lines or from very remote locations (Ritter et al. 2005). Neither of these types of users is likely to be the same as many designers.

Although there is no general solution to the question of when to stop analyzing the user and start building the system, Pew and Mavor's (2007) approach provides a subjective answer. In their risk driven approach, the risks to success are re-evaluated as the design process progresses. In some cases, progress with the technology is the largest risk to success; in others, not knowing the user and their tasks will be the largest risk. So Pew and Mavor's answer is that you should study the user and their tasks until the risk of not knowing more about them is lower than the other risks to success. As noted above, we will return to describe this approach in more detail in the final chapter, Chap. 14.

1.4 Summarizing Design Relevant User Characteristics: The ABCS Framework

The purpose of this book is to help you to come up with principled opinions about what designs are most likely to be effective. We introduce the idea of *design relevant user characteristics*. Attempts to define a complete list of human characteristics stretch from hundreds (e.g., Brown 1988) to thousands of pages (Boff and Lincoln 1988; Salvendy 1997). Table 1.2 offers some examples that are often discussed, taken from Brown (1988).

To help organize design relevant human characteristics, we offer a framework that we call the ABCS. The abbreviation represents four aspects of users that often need to be examined when designing systems:

A Anthropometrics: the shape of the body and how it influences what is designed; consideration of the physical characteristics of intended users such as what size they are, what muscle strength they have, and so on

B Behavior: perceptual and motivational characteristics, looking at what people can perceive and why they do what they do

C Cognition: learning, attention, and other aspects of cognition and how these processes influence design; users defined by how they think and what they know and what knowledge they can acquire

S Social factors: how groups of users behave, and how to support them through design; users defined by where they are—their context broadly defined including their relationships to other people.

We now briefly explain each of these areas in more detail.

Table 1.2 Human characteristics relevant for system design

• Physical characteristics, limitations, and disabilities
• Perceptual abilities, strengths, and weaknesses
• Frequency of product use
• Past experience with same/similar product
• Activity "mental set" (the attitude toward and level of motivation you have for the activity)
• Tolerance for error
• Patience and motivation for learning
• Culture/language/population expectations and norms

1.4.1 Anthropometrics Approach

Anthropometrics is concerned with the physical aspects of the user and the system. For example, Fig. 1.4 shows an input glove. How do people use this? What are their natural movements in it, and do these movements change with a glove on? How long can they use it before becoming fatigued or hurt by it? The answers to questions like these would involve resolving the issues in Table 1.3.

These physical aspects are often studied in the area of human factors and ergonomics and applied to standard office equipment like desks and chairs. A lot is known about how to improve the fit of the hardware to the user's body, including back, knees, waist, and arms (Pheasant and Haslegrave 2006). The optimal work surface height, for example, varies by the individual concerned but also by the task to be performed.

It is also important that we consider whether we need to design for individuals (e.g., desk setups need to be specifically tailored to avoid upper limb disorders), for the average (e.g., seats in buses and trains are designed for averages), or for extremes (e.g., plane ejector seats). For example, Ted Williams, the famous American baseball player and fighter aircraft pilot, reportedly crash-landed a plane rather than eject so that he would be able to play baseball again after the Korean war—the design error was that ejector seats were designed for the average height of a pilot, which left those in the upper 5–10% of the height range in danger of damaged or removed kneecaps if they ejected.[1]

In computer systems these problems include making sure that controls (knobs, dials, buttons, and so on) are of a size that can be manipulated by a wide range of users. Weight and button size are important for their usability and the perceived usability for their marketing. For example, the release of many different sizes of interactive tablet computers into the market over recent years suggests the physical sizes of these devices matter for different use scenarios. Early mobile phones were

[1] http://www.tedwilliams.com/index.php?page=burnjet

Fig. 1.4 An input glove. (Photo taken by and used with permission of Georgios Christou)

Table 1.3 Example usability issues arising from the anthropometric level

- Providing normative data on limb sizes, body weight/height, and so on
- Providing descriptions of how sensitive touch is for input and output, particularly for the hands
- Measurement of muscle strain (to assess length of time on a particular job)
- Measurement of posture during particular tasks (to facilitate redesign of equipment)

the size of briefcases, and laptops weighed 30 pounds; these failed to be as popular as expected partly because they were not small enough.

These issues are equally important in the design of mobile devices. Weight and button size are important for their usability and the perceived usability for their marketing. These issues will be more important for reality-based interfaces, where computing is embedded into objects such as passports, children's toys, and objects that have RFID tags which allow them to be tracked. These interfaces include both computational aspects as well as the physical nature of the objects and the opportunities and constraints that physical realization provides.

Table 1.4 Example usability issues arising from the behavioral level

Car interfaces—questions of legibility of characters, avoidance of glare in bright sunlight, avoiding parallax problems with different heights of drivers, and making sure that the dials are not obscured by the steering wheel
Making knobs and levers tactually discriminable to enable them to be used without looking to check whether the correct control is being used (e.g., putting a wheel on the landing gear lever in a plane)
Problem of ascertaining the physical actions of how something is used, to see whether it can be made quicker/safer/more productive, and so on
Looking at simple errors (slips of action) that are made, to see how they can be mitigated or prevented

Table 1.5 The original Fitts (1951) list

Humans appear to surpass present-day (i.e., 1951) machines with respect to:
• Ability to detect small amounts of visual or acoustic energy
• Ability to perceive patterns of light or sound
• Ability to improvise and use flexible procedures
• Ability to store very large amounts of information for long periods and to recall relevant facts at the appropriate time
• Ability to reason inductively
• Ability to exercise judgment
Present-day machines appear to surpass humans with respect to:
• Ability to respond quickly to control signals, and to apply great force smoothly and precisely
• Ability to perform repetitive, routine tasks
• Ability to store information briefly and then to erase it completely
• Ability to reason deductively, including computational ability
• Ability to handle highly complex operations, that is, to do many different things at once

1.4.2 Behavioral Aspects

When we discuss the behavioral aspects of the user, we refer to the basic behaviors users can produce. The behavioral level builds on the anthropometric level as the physical aspects of the body are used to produce simple behaviors. Table 1.4 provides several examples, drawing on a wide range of application areas.

Behavioral analysis has supported and led to the creation of checklists of those tasks best performed by humans and those best performed by machines. Table 1.5 shows an example of such a list, where human and machine tasks could be assigned on the basis of better fit. Such lists have been critiqued for being too static (Sheridan 1992), but the exercise of making such lists, updated according to technological innovations and changes in human expectations and abilities through training, can be useful. An excellent example of the evolution of the way we think about task allocation is the recent success of IBM's Watson system. Historically,

because humans reason creatively and inductively through association and using mnemonics, they could easily beat computers on standard general knowledge tests. However, since the advent of the Internet, which represents a massive database of general knowledge that has been supplied through untold hours of human content contribution and improvements in computational processing power and search algorithms, it is possible for a machine, Watson, to beat a human at such tests. What we see in this example is that the notion of even what a *machine* is can change over time. Therefore, when considering allocation of processing activities between humans and computational devices, we need to ensure we are using the most appropriate sense of the term *machine*.

We also include simple interaction at the behavioral level. For example, Norman (2013) has written about "forcing functions". These are aspects of devices that suggest particular uses or styles of interaction. In some cases, affordances force a particular style of interaction. One of Norman's favorite examples is door design. Doors with handles suggest that they should be pulled, while doors without handles suggest that they should be pushed. A *forcing function* would be where the door with the handle cannot be pushed, thus forcing that it be pulled. Violation of these affordances (doors that can only be pushed yet have handles) leads to confusion. Perceptual issues sit between the behavior and cognition. For example, PowerPoint presentations where the font is too small means people cannot read the content unless they move physically closer (Kosslyn 2007).

In addition to noting the basic foundations that explain *how* people behave, we also have to consider *why* people behave in the way that they do. The motivation that people have for performing particular tasks will vary, partly depending on internal factors, but also partly on external factors (e.g., is it a work task, is it being carried out in an informal setting, and so on.)

1.4.3 Cognition

The cognitive level uses the previous two levels and builds upon them. On this level, how the user thinks about their task and the system is considered, as well as both basic and higher level cognitive capabilities. These capabilities include a variety of memory systems that the user has available, as well how these memories are organized and how they are used by a central processor. Higher level constructs include how attention and learning affect these structures and processes. Some example cognitive issues are shown in Table 1.6.

Work on the cognitive level will often involve observation of the tool/environment in use, asking the question of why and when is it used? This is necessary because users will vary more on this level of analysis than on the previous two levels. On this level, people will vary based on previous experience, which can include training, formal education, previous use, personal style, and strategy choice.

Table 1.6 Example cognitive level issues

How do users decide where to look for information?
What information do users need to develop a strategy for performing a particular task? Do they need absolute or relative values?
How much experience do the users have with the task and with the interface?
What is the user's mental model of the interface and task (which will often differ from the designer's or the observer's mental model of the same interface and task)?
Is there so much noise and interruption that users cannot process information, for example, the interruptions in the Kegworth aircraft crash?
How can users tell if things are not going well? What feedback do they get? What strategies are available to the user when the system goes wrong?
How can we ensure that users do not lose their ability to perform the task manually as a result of automation?
How can word processors be made more accessible to novice users or casual users? How do these factors change when the systems considered are more complex?

A better understanding of how users think and feel can be used to create better designs. An improved system can come from understanding the mental effort involved in tasks in terms of the information processing mechanisms (architecture) that support our thinking, including constraints such as how many arbitrary symbols users can remember. These issues may help us to understand how complex devices with high functionality (e.g., personal video recorders) can be made more accessible to non-expert users by providing information, by guiding the user, and by not asking them to perform difficult tasks (like remembering more than their short-term memory can hold).

For example, consider the fairly common task of determining differences between two or more items. Imagine you had two dials to read, each with a different number, and these numbers varied randomly. Your task is to press a button when the difference between the two numbers exceeds 10. This task would require you to process the information from both dials, make a mental calculation and evaluate whether the difference exceeds 10. The existence of a third dial which just showed the difference would make your task easier and faster, removing the need for the mental calculation (the cognitive effort).

1.4.4 Social Factors

The final level is the social level. How do users interact with other people in relation to their task? In some cases this interaction will be to work on the task jointly with others using the computer to support and mediate their communication. In other cases, users will ask other users for help in understanding systems, or will use the system for, or with others (such as bank tellers, loan officers, and airline pilots), or the interaction could be constrained by some regulatory authority (as in aviation and the nuclear power industry).

Table 1.7 Example issues on the social level

• A crew distracted by interruptions that failed to complete a safety checklist did not confirm that the aeroplane's flaps were extended, causing the plane to crash on take-off
• A co-pilot failed to get the attention of a more senior captain about concerns that take-off thrust was not properly set, causing the aircraft to crash into a river
• A communications breakdown between captain, co-pilot, and air traffic control on the amount of fuel in the plane caused a crash when the fuel ran out
• You want to buy the same video game as your best friend so you can play him at your house, and so you can practice to beat him!

Like the previous levels, this level builds upon and uses the constructs and theories of the previous level. In this case, the cognitive level, including the mental models of others, is particularly important.

The motivation that people have for performing particular tasks and working in teams will vary, partly depending on internal factors but also partly on external factors (e.g., is it a work task, is it being carried out in an informal setting, and so on).

The social level can be very important. Many of the accidents in safety–critical systems, for example planes, have their roots in the dynamics of the social processes between people controlling various parts of the systems, and their social environment. Perhaps the most typical failure is for a subordinate not to tell a superior or not to tell them forcefully enough about an impending problem, which then becomes unmanageable because of the delay. Simple failures in inter-personal communications can also cause accidents. Table 1.7 lists some further examples of issues on the social level.

Flowers (1997) explains how the task of moving the London ambulance dispatching system from paper to computer went wrong. The designers seriously misunderstood how the dispatchers worked, how the drivers worked, and how the two groups worked together. There were also software development and implementation problems. While no loss of life was reported, ambulance response times were seriously compromised, the director was sacked, and about 3 million pounds ($4.5 million) worth of development was written off. This is an example where social factors were ignored in system design.

Organizational, professional, and national cultural issues—how users from different cultural backgrounds have different characteristics—are also grouped under this heading in this book. Examples of these differences include how colors can mean different things in different cultures: green does not always mean go, and white may be the traditional color for funeral dress rather than black. Other examples include how the most natural ordering of objects may be left to right in many cultures but right to left in others, and how some cultures encourage appropriate questioning of people in responsible positions (such as aircraft pilots), whilst others frown upon it.

As can be seen from the Kegworth air disaster (described in the Appendix), the cabin crew (as well as the passengers) knew that the wrong engine may have been

turned off, but did not interrupt the pilots, possibly because of the high social status accorded to the pilots. This type of cultural issue, where someone knows something that could help a team or a project and does not raise the issue, is a well documented problem. How to adjust appropriately the social dynamics to fix problems like this remains an important and interesting problem.

Another example comes from a nuclear power plant in Europe. A reporting system was set up to allow staff to report incidents (near misses) so that the company could learn from them to try and prevent the same thing happening (Masson 1991). This was all working fine, and management and staff had settled into using the system. Staff were happy to report incidents and were not blamed when the incidents did occur. The management then decided that they would change the (negotiated) culture, in which the emphasis had been on reporting incidents, to one that focused on incidents as a measure of safety, and decided that the shift that reported the least incidents would be regarded as the safest shift and would receive some reward.

The net effect was that staff stopped reporting incidents in a bid to make their shift appear to be the safest. In the end a new incident reporting system had to be developed, and all the data about the unreported incidents was lost because the management had unwittingly changed the culture from one that was designed to use reported problems as a way of learning and improving safety to one that was effectively designed to reward the lack of reporting of problems.

1.5 Simulating User Characteristics: Cognitive Architectures

One of the main aims of this book is to help you to develop a better understanding of why users do things the way they do. Understanding the way users think and behave will help you design systems that support users. The ABCS framework, described above, provides one way of organizing this information about user characteristics. It is also possible to encapsulate relevant details in a model. For example, if one is interested specifically in human information processing, cognitive architectures provide a convenient way of modeling human information processing under different conditions, because they include mechanisms that are specifically designed for modeling human cognition.

Figure 1.5 is a schematic of the major components of a computer model of a user. The major components in this model are designed to mimic the major components of users. The top box, ACT-R, refers to a simplified form of the ACT-R cognitive architecture (Anderson and Lebiere 1998). (There are other architectures, but they are similar for our purposes.) In this instance the architecture has been combined with an extension that allows it to interact effectively with the external world. So the combined cognitive architecture takes the bitmap from a computer screen and, in a process approximating vision, computes the objects and some of their features in the image and puts the results into a perceptual buffer.

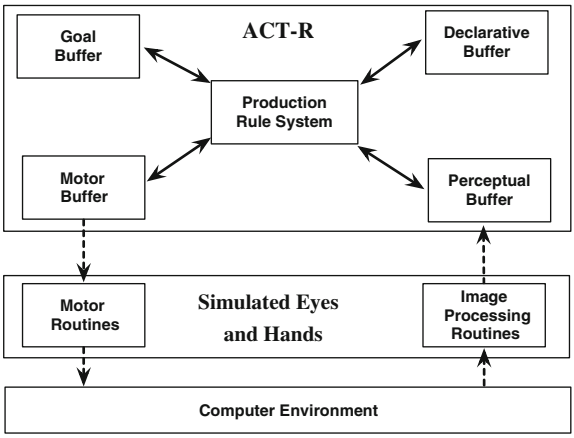

Fig. 1.5 A representation of the ACT-R cognitive architecture with the SegMan extension to allow it to interact with interfaces through computer bitmaps

This perceptual buffer represents many of the aspects of the human vision system. Similar buffers should be created for hearing and the other senses.

At the center of the architecture is a reasoning system that can access information in the perceptual buffer and access and modify information in the declarative memory and goal buffers. It uses the information in these buffers to guide its processing, using a process that mimics human behavior. Output actions to the external world are put into a motor buffer, where motor routines generate the results.

This representation is somewhat generic and far from complete, but illustrates a theory of the major information processing components of the user. There are other theories of how users process information, and even the ACT-R theory has changed over time. Work is ongoing to apply this approach to create models of users that can be used to test interfaces (e.g., Byrne 2001; Freed and Remington 2000; Kieras et al. 1997), to serve as intelligent colleagues (Weiland et al. 2002) and opponents in computer-based games and simulations (e.g., Jones et al. 1999; Laird and van Lent 2001; Tambe et al. 1995), and to summarize our knowledge of human behavior (e.g., Anderson et al. 2004; Jones et al. 2000; Lovett et al. 2000).

1.6 Summary

This chapter has introduced the study of people who use artifacts (i.e., *users*), given you some definitions and useful terms, and provided an overview to organize your further reading. We have also highlighted some common (mistaken) preconceptions about what makes for good design and noted why studying users is important. In particular, we have introduced the concept of the fundamental attribution error of design, where designers think that users are like themselves, which is very often not the case.

We believe that understanding users is fundamentally important and often leads to more *usable* systems. There are costs as well as benefits associated with studying and understanding users, however, and it is important to realize when you should decide to stop analyzing the users and start building your system. A risk-based approach to development can help inform this decision. We explain this more fully in the Chap. 14 that you might wish to preview.

The information about users' design related capabilities can be organized using the ABCS framework to encapsulate people's physical shape, how they perceive, how they think, and how they interact with other people. This simple abbreviation can be used to remember the information and to guide your considerations during system design. Chapters 3–10 cover these levels.

We also introduced the idea of cognitive architectures, which can be used to develop models that simulate how people perceive, think, and act. Although we used the ACT-R architecture as an exemplar, other cognitive architectures could also be used to organize and apply knowledge about users. We take up this approach again in the Chap. 14.

1.6.1 Structure of the Rest of the Book

You can think of the book as being divided into four parts. The first two chapters introduce the notion of user centered design and examine the underlying history.

The second part of the book (Chaps. 3–10) examines specific characteristics of the user. Our perspective draws heavily on psychology. We focus on the sensory and cognitive, information processing, aspects of the user, partly because this is where our expertise lies, but also because much interaction requires people to sense, understand, communicate, and learn. We limit our discussions to the topics that are most relevant to designers while also offering pointers to those people who want to read more about these topics.

The third part of the book (Chaps. 11–13) introduces some methods that can be used to inform and evaluate design. This is an active area of application and research, and there are now so many different approaches that it would be impossible to cover them all here.

The book concludes with a short summary (Chap. 14). This highlights how you can organize what you have learned about users and notes some possible directions that are currently being explored to apply it.

On completing this book you will have acquired an understanding of humans and their behavior when interacting with complex, interactive systems. You will have sufficient grounding to be better able to design systems that take appropriate account of your users, their tasks, and the context in which they perform those tasks. You will be able to justify how (and why) a particular design is appropriate from a conceptual level down to a practical level, using the toolbox of methods and techniques that we have placed at your disposal.

Table 1.8 Some remaining problems in contemporary system design

- The size of systems and diversity of users and tasks are increasing. How are we to find, represent, and use this information?
- The complexity of the systems is increasing: users do not always get adequate feedback on what is going on, and cannot see the internal state of the system. Norman (2013) provides further examples. How are we to keep users informed without overwhelming them with information?
- The nuances of social and organization factors and the communication of these nuances through computer supported communication are not fully understood and predictable. How can designers get and use this information? How can it be represented?
- How can we improve the usability of designer's tools to help them improve usability for users?
- Studies of the user need to go beyond recommendations about the design of technology—can we offer a conceptual basis for these recommendations? One approach is to create a unified theory of how users behave, but this theory has not yet been fully created or made available for automatic application
- With further understanding come questions about lower and higher levels. Once we know how users work in small groups we can see that larger groups also have influence as do previous groups who used the system. What is this information and how do we include this information?
- Esthetics and emotions are difficult factors to explain and quantify. Users, particularly users of personal and home technologies, generally care about how the system looks and what pleasure it gives them, sometimes irrespective of how it works or how well it works. In these areas, then, esthetics is closely related to acceptability, and there is some evidence of a high correlation between esthetics and usability (e.g., Tractinsky 1997). The inter-relationship between usability, functionality, emotional responses, and esthetics, however, still needs to be worked out

1.6.2 Future Work

You should now be aware that there is a lot to be learned about users. While technology can change rapidly basic user capabilities and characteristics change slowly, if at all. It is important to be aware of critical user characteristics, their relative importance for a design, and their likelihood of change over time.

There is also much more that needs to be learned. As you read the rest of this book, you should become aware of the remaining problems, some of which are listed in Table 1.8.

1.7 Other Resources

Here we note some further books and online resources.

The books by Christopher Wickens (Wickens et al. 1998; Wickens and Hollands 2000) provide more details than this book does. These books focus more on human factors and physical engineering of workplace.

The books by Preece et al. (2002) and Dix, Finlay, Abowd, and Beale, both titled *Human–Computer Interaction*, and *Interaction Design: Beyond Human–*

Computer Interaction, 3rd Edition, by Sharp, Rogers, and Preece are all worth a closer look. These books include more on technology and on designing interface technology rather than focusing in detail on the various aspects of the users that affect interaction. Clayton Lewis and John Rieman's shareware book: *Task-Centered User Interface Design*, hcibib.org/tcuid/ covers similar material but focuses more on design and task analyses, and does not include as much psychology. It helps put the material in this book into perspective for design purposes.

Descriptions of design failures often make interesting reading, and their lesson can sometimes be generalized to other systems. For large systems, Petroski's (1985) book is one of the most popular. Petroski's observation that engineering has progressed over the centuries by learning from failure also applies to interface design. The ACM Committee on Computers and Public Policy's Risks Digest at catless.ncl.ac.uk/Risks/ provides numerous examples of where poor usability has led to problems.

There are many online resources available on HCI and human factors. The HCI Bibliography Project, hcibib.org, provides an online bibliography of papers in journals and conferences related to HCI. The Computer–Human Interaction Special Interest Group of the Association for Computing Machinery (ACM-SIGCHI) has a very useful web site at www.acm.org/sigchi/ They organize the CHI conference every year, as well as producing a magazine, *interactions*.

You can also examine the design and graphic design literature to understand esthetics better. There are books on the design of graphic information (e.g., Tufte 1990), on the design of pages (e.g., White 2002), on the design of fonts, page layouts, graphics, and so on. If you are interested in this area, consider looking through libraries (personal, institutional, and public), using search engines, and consulting academic course catalogues, all of which can provide you with much more information.

Don Norman's books, including the classic *The design of everyday things*[2] (1988/2013), provide a useful set of examples of why design matters. At times the examples may seem small, but in nearly every case their impact is magnified by the number of people affected, the possible severity of the consequences, or their clarity. Many people have been convinced of the importance of HCI as a result of reading this book.

The importance of esthetics should not be underestimated, even though we do not cover it in any great detail here. Jordan (2000), for example, argues that esthetics of devices will increase over time. When systems are equivalent, it may be the case that their users differentiate between them on the basis of how pleasing their appearance is. Some researchers have found that pleasing interfaces work better (Helander and Tham 2003), and Norman (2006, 2009) argues this position repeatedly, including that esthetics and functionality should be considered equally important.

---

[2] Also published sometimes as *The psychology of everyday things.*

[3] A millisecond is a thousandth of a second, and is abbreviated ms.

1 File Home Insert Page Layout References Mailings Review View

2 Home Insert Page Layout References Mailings Review View Add-Ins

3 [OS Icon] Word File Edit View Insert Format Font Tools Table Window Work Help

4 File Edit View Insert Format Tools Table Window Help

5 Home File Insert Page References Layout Review View Help

6 Home File Edit View Insert Font Tools Review Window Help

7 [OS Icon] File Edit View Insert Font Tools Review Window Help

8 [OS Icon] Home View File Insert Review Layout Table Work

9 Word Home File Insert Page Layout References Review View Help

10 [OS Icon] File Edit View Format Arrange Inspectors Stencils Window Help

Fig. 1.6 Possible views of the top menu on MS Word. Without looking at Word, which of the ten menus is the one you use?

1.8 Exercises

1.1 The correct ranked order for the buttons in Fig. 1.1 is as follows: (a) (411 ms[3]), (d) (469 ms), (b) and (c) (each 539 ms). The lights in D have a simple rule that users can follow– look to the diagonal. The lights in B and C have two rules (or at least a more complex rule): if the lights are in the middle, hit the button below; if the lights are on the end, hit the diagonal button. The extra conditions, or rules, take extra time to learn and extra time to perform. They would also lead to more errors.

Ask some of your friends and or family to choose the best designs in Figs. 1.1 and 1.2. How do they compare to your performance and to Payne's subjects? If you have access to interface designers, ask them. How does your own stove compare to these mappings? What do your results mean for interface design?

1.2 In Fig. 1.3 the correct coin is Penny A. The point is, most people do not memorize the features of objects in detail, they memorize just enough to recognize and differentiate the coin from typical distracters, like other legal coins, or to find the right menu item when it is in front of them. Try this simple test for the penny with your friends, and also ask computer users to identify the correct Word menu in Fig. 1.6.

1.3 Consider a mobile device (such as a smartphone or tablet). What difficulties might people encounter in using the device for the first time? What difficulties might people have in understanding how to use it and what to do with it? How would they go about learning how to use it?

[3] A millisecond is a thousandth of a second, and is abbreviated ms.

Write short notes (about one side of a page) on some of these issues you have identified, classifying them as anthropometric, behavioral, cognitive, or social. Think about what design changes you might make to the device to make it easier for novices to use.

1.4 Consider an airplane crash like Kegworth, the Asiana in San Francisco, or another one where you can obtain some of the details. Classify the problems that led to the disaster with respect to the four levels introduced here. Summarize what level was the most important and could have stopped the disaster.

1.5 Select something you use every day that you think is well designed. Think about why this is well designed. You may wish to consider esthetics, mappings of actions to responses, how you learned to use it, and what kinds of mistakes or errors you still make.

1.6 What are some other payoffs from studying the user? As ways to brainstorm, consider the types of outputs the various fields related to users would provide. As another hint, consider fields that study users or aspects of users, and consider what they might want from interfaces or from interactions with interfaces or systems.

References

Anderson, J. R., & Lebiere, C. (1998). *The atomic components of thought*. Mahwah, NJ: Erlbaum.

Anderson, J. R., Bothell, D., Byrne, M. D., Douglass, S., Lebiere, C., & Qin, Y. (2004). An integrated theory of the mind. *Psychological Review, 111*(4), 1036–1060.

Baxter, G. D., Monk, A. F., Tan, K., Dear, P. R. F., & Newell, S. J. (2005). Using cognitive task analysis to facilitate the integration of decision support systems into the neonatal intensive care unit. *Artificial Intelligence in Medicine, 35*, 243–257.

Boff, K. R., & Lincoln, J. E. (Eds.). (1988). *Engineering data compendium (User's guide)*. Wright-Patterson Air Force Base, OH: Harry G. Armstrong Aerospace Medical Research Laboratory.

Booher, H. R., & Minninger, J. (2003). Human systems integration in army systems acquisition. In H. R. Booher (Ed.), *Handbook of human systems integration* (pp. 663–698). Hoboken, NJ: John Wiley.

Brown, C. M. L. (1988). *Human-computer interface design guidelines*. Norwood, NJ: Ablex.

Bruce, V., Gilmore, D., Mason, L., & Mayhew, P. (1983). Factors affecting the perceived value of coins. *Journal of Economic Psychology, 4*(4), 335–347.

Byrne, M. D. (2001). ACT-R/PM and menu selection: Applying a cognitive architecture to HCI. *International Journal of Human-Computer Studies, 55*(1), 41–84.

Casper, J., & Murphy, R. (2003). Human-robot interactions during the robot-assisted urban search and rescue response at the World Trade Center. *IEEE Transactions on Systems, Man, and Cybernetics Part B*, *33*(3), 367–385.

Chapanis, A., & Lindenbaum, L. E. (1959). A reaction time study of four control-display linkages. *Human Factors, 1*(4), 1–7.

Ericsson, K. A., & Simon, H. A. (1993). *Protocol analysis: Verbal reports as data* (2nd ed.). Cambridge, MA: MIT Press.

Fitts, P. M. (1951). Engineering psychology and equipment design. In S. S. Stevens (Ed.), *Handbook of experimental psychology* (pp. 1287–1340). New York, NY: John Wiley.

Flowers, S. (1997). *Software failure: Management failure… Amazing stories and cautionary tales*. New York, NY: Wiley.

Freed, M., & Remington, R. (2000). Making human-machine system simulation a practical engineering tool: An APEX overview. In *Proceedings of the 3rd International Conference on Cognitive Modelling* (pp. 110–117). Veenendaal, The Netherlands: Universal Press.

Gigerenzer, G. (2004). Dread risk, september 11, and fatal traffic accidents. *Psychological Science, 15*(4), 286–287.

Glushko, R. J., & Tabas, L. (2009). Designing service systems by bridging the "front stage" and "back stage". *Information Systems and e-Business Management, 7*(4), 407–427.

Gray, W. D., John, B. E., & Atwood, M. E. (1992). The precis of project Ernestine or an overview of a validation of GOMS. In *Proceedings of the CHI'92 Conference on Human Factors in Computer Systems*. New York, NY: ACM Press.

Gray, W. D., John, B. E., & Atwood, M. E. (1993). Project Ernestine: Validating a GOMS analysis for predicting and explaining real-world task performance. *Human-Computer Interaction, 8*(3), 237–309.

Helander, M. G., & Tham, M. P. (2003). Hedonomics: Affective human factors design. *Ergonomics, 46*(13/14), 1269–1272.

Holmes, N. (2005). The Internet, the Web, and the Chaos. *IEEE Computer, 38*(108), 106–107.

Johnson, E. J., Bellman, S., & Lohse, G. L. (2003). Cognitive lock-in and the power law of practice. *Journal of Marketing, 67*, 62–75.

Jones, G., Ritter, F. E., & Wood, D. J. (2000). Using a cognitive architecture to examine what develops. *Psychological Science, 11*(2), 93–100.

Jones, R. M., Laird, J. E., Nielsen, P. E., Coulter, K. J., Kenny, P., & Koss, F. V. (1999). Automated intelligent pilots for combat flight simulation. *AI Magazine, 20*(1), 27–41.

Jordan, P. W. (2000). *Designing pleasurable products*. London: Taylor & Francis.

Kieras, D. E., Wood, S. D., & Meyer, D. E. (1997). Predictive engineering models based on the EPIC architecture for a multimodal high-performance human-computer interaction task. *Transactions on Computer-Human Interaction, 4*(3), 230–275.

Kosslyn, S. M. (2007). *Clear and to the point: 8 psychological principles for creating compelling Powerpoint presentations*. New York, NY: Oxford University Press.

Laird, J. E., & van Lent, M. (2001). Human-level AI's killer application: Interactive computer games. *AI Magazine, 22*(2), 15–26.

Leveson, N. G., & Turner, C. S. (1993). An investigation of the Therac-25 accidents. *IEEE Computer, 26*(7), 18–41.

Lovett, M. C., Daily, L. Z., & Reder, L. M. (2000). A source activation theory of working memory: Cross-task prediction of performance in ACT-R. *Journal of Cognitive Systems Research, 1*, 99–118.

Masson, M. (1991). Understanding, reporting and preventing human fixation errors. In T. W. v. d. Schaaf, D. A. Lucas & A. Hale (Eds.), *Near miss reporting as a safety tool* (pp. 35–50). Oxford, UK: Butterworth-Heinemann.

Nickerson, R. S., & Adams, M. J. (1979). Long-term memory for a common object. *Cognitive Psychology, 11*, 287–307.

Nielsen, J. (1993). *Usability engineering*. Chestnut Hill, MA: AP Professional Press.

Norman, D. A. (1981). Categorization of action slips. *Psychological Review, 88*, 1–15.

Norman, D. A. (2006). *Emotional design: Why we love (or hate) everyday things*. New York, NY: Basic Books.

Norman, D. A. (2009). *The design of future things*. New York, NY: Basic Books.

Norman, D. A. (2013). *The design of everyday things*. NY: Basic Books.

Payne, S. J. (1995). Naive judgments of stimulus-response compatibility. *Human Factors, 37*, 495–506.

Petroski, H. (1985/1992). *To engineer is human: The role of failure in successful design*. New York, NY: Vintage Books.

Pew, R. W., & Mavor, A. S. (Eds.). (2007). *Human-system integration in the system development process: A new look*. Washington, DC: National Academies Press. http://books.nap.edu/catalog.php?record_id=11893. Accessed 10 March 2014.

Pheasant, S., & Haslegrave, C. M. (2006). *Bodyspace: Anthropometry, ergonomics, and the design of work* (3rd ed.). Boca Raton, FL: Taylor & Francis.

Preece, J., Rogers, Y., & Sharp, H. (2002). *Interaction design*. New York, NY: Wiley.

Reason, J. (1990). *Human error*. Cambridge, UK: Cambridge University Press.

Ritter, F. E., Freed, A. R., & Haskett, O. L. (2005). User information needs: The case of university department web sites. *ACM interactions, 12*(5), 19–27. acs.ist.psu.edu/acs-lab/reports/ritterFH02.pdf

Salvendy, G. (Ed.). (1997). *Handbook of human factors and ergonomics* (2nd ed.). New York, NY: Wiley.

Sheridan, T. B. (1992). *Telerobotics, automation, and human supervisory control*. Cambridge, MA: MIT Press.

Tambe, M., Johnson, W. L., Jones, R. M., Koss, F., Laird, J. E., Rosenbloom, P. S., et al. (1995). Intelligent agents for interactive simulation environments. *AI Magazine, 16*(1), 15–40.

Tractinsky, N. (1997). Aesthetics and apparent usability: Empirically assessing cultural and methodological issues. In *CHI '97* (pp. 115–122). New York, NY: ACM. http://sigchi.org/chi97/proceedings/paper/nt.htm. Accessed 11 March 2014.

Tufte, E. R. (1990). *Envisioning information*. Cheshire, CT: Graphics Press.

Weiland, W., Szczepkowski, M., Urban, G., Mitchell, T., Lyons, D., & Soles, R. (2002). Reusing cognitive models: Leveraging SCOTT technology in an LCAC virtual training environment. In *Proceedings of the 11th Computer Generated Forces Conference*, 02-CGF-115. Orlando, FL: U. of Central Florida.

White, A. W. (2002). *The elements of graphic design: Space, unity, page architecture, and type*. New York, NY: Allworth Press.

Wickens, C. D., Gordon, S. E., & Liu, Y. (1998). *An introduction to human factors engineering*. New York, NY: Addison-Wesley.

Wickens, C. D., & Hollands, J. G. (2000). *Engineering psychology and human performance* (3rd ed.). Upper Saddle River, NJ: Prentice-Hall.