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